



**La Sapienza**

Università degli Studi di Roma

Dipartimento di Informatica e Sistemistica

# CALCOLATORI ELETTRONICI

## Sistemi di numerazione e codici

**Emiliano Trevisani**

**[trevisani@dis.uniroma1.it](mailto:trevisani@dis.uniroma1.it)**

**A.A. 2007/2008**



## □ **Informazione**

- Concetto astratto
- Numerica o alfanumerica
- E' necessario trasformala in un formato compatibile con i meccanismi di processing di un sistema di calcolo  $\Rightarrow$  rappresentazione

## □ **Rappresentazione dell'informazione:**

- $\text{Informazione} \Leftrightarrow \text{Sequenza di simboli}$  [mapping 1:1]
- Il protocollo di rappresentazione deve essere condiviso da tutte le entità coinvolte dallo scambio informativo

*Informazione*  $\xrightarrow{\text{Codifica}}$  *Sequenza di simboli*

*Sequenza di simboli*  $\xrightarrow{\text{Decodifica}}$  *Informazione*



### □ **Informazione numerica**

- Numero: entità astratta
- Numerale: sequenza di simboli che rappresenta un numero in un dato sistema di numerazione [meccanismo di codifica]
- Numerali diversi [sistemi di numerazione diversi] possono rappresentare lo stesso numero: 20 nel sistema decimale, XX nel sistema di numerazione utilizzato nell'antica Roma
- Un numerale è in generale composto da un certo numero ( $n$ ) di simboli scelti in un insieme finito;  $n$  determina l'intervallo di numeri rappresentabili

### □ **Sistemi di numerazione posizionali:**

- Si riferiscono ad una base intera  $b > 1$ ;  $b [0, 1, 2, 3, \dots, b-1]$  sono anche le cifre usate dal sistema di numerazione per comporre numerali; oltre la cifra 9 sono adottati i caratteri A, B, C, ...



### □ Sistemi di numerazione posizionali:

- Il numero rappresentato da un numerale è dato dal valore di un polinomio i cui coefficienti (pesi) sono le potenze intere della base  $b$  positive a sx della virgola, negative a dx
- Indicando con  $c_i$  la cifra nella generica posizione  $i$  di un numerale:
  - $(N)_b = c_s c_{s-1} c_{s-2} \dots c_0, c_{-1} c_{-2} \dots c_{-r}$  rappresentazione di  $N$  nella base  $b$  adottando  $s+r+1$  cifre;  $c_i \in \{0, 1, \dots, b-1\}$ ;
  - Numero rappresentato  $[0 \leq V(c_i) \leq (b-1)]$

$$V(N) = \sum_{i=-r}^s c_i b^i = c_{-r} b^{-r} + \dots + c^0 b^0 + \dots + c^s b^s$$

- Sono detti posizionali in quanto il peso di ciascuna cifra all'interno del numerale dipende dalla posizione della cifra stessa; il sistema di numerazione utilizzato nell'antica Roma non è posizionale
- $b=2 \Rightarrow$  **Sistema di numerazione binario** [ $c_i \in \{0, 1\}$  binary digit  $\rightarrow$  **bit**]



### □ Sistemi di numerazione posizionali:

- Utilizzando la base  $b > 1$  e numerali di lunghezza  $n$  è possibile rappresentare tutti i numeri interi positivi da 0 a  $b^n - 1$  [sono  $b^n$  numeri diversi]

- Dimostrazione. [per induzione]

- $n=1$  banale;

- $n > 1$  ipotesi induttiva:  $\sum_{i=0}^{n-1} (b-1)b^i = b^n - 1$

$$\Rightarrow \sum_{i=0}^n (b-1)b^i = \sum_{i=0}^{n-1} (b-1)b^i + (b-1)b^n = b^n - 1 + (b-1)b^n = b^{n+1} - 1$$

- Si può estendere il risultato precedente riferendosi al numero di “numeri diversi rappresentabili”
- Fissata  $b > 1$  la minima lunghezza  $n$  da adottare per poter rappresentare un numero  $N$  è data da:

$$b^n - 1 \geq N \Rightarrow n = \lceil \log_b N \rceil$$



### □ Sistemi di numerazione posizionali:

- Nei calcolatori elettronici si utilizzano elementi a due soli stati (**dispositivi bistabili**); per questo motivo è opportuno adottare un sistema in base 2 [due simboli ciascuno associato ad uno stato].
- **Conversione di base:** Sia I un numero intero positivo rappresentato nella base b da  $(I)_b$ ; si intende ottenere la rappresentazione del numero in base a:
  - $I = c_0 + a(c_1 + a(c_2 + \dots)) = a \cdot Q + R$  [ $0 \leq R < a$ ] ← Q è il quoziente della divisione di I per a con resto  $R = c_0$
  - La cifra meno rappresentativa ( $c_0$ ) della rappresentazione nella nuova base è il resto della divisione di  $(I)_b$  per  $(a)_b$ ; la divisione è da intendersi nell'aritmetica in base b
  - Si procede analogamente per divisioni successive ottenendo  $c_1, c_2, \dots, c_n$ ; il processo si arresta a  $Q=0$

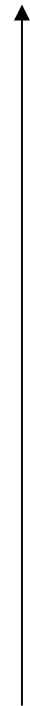
## Sistemi di numerazione posizionali



### □ Sistemi di numerazione posizionali:

- Conversione di base: esempio,  $(325)_{10} \Leftrightarrow (?)_2$  [Decimale  $\rightarrow$  Binario]

Q	$\div$ 2	R	
325		1	$C_0$
162		0	$C_1$
81		1	$C_2$
40		0	$C_3$
20		0	$C_4$
10		0	$C_5$
5		1	$C_6$
2		0	$C_7$
1		1	$C_8$



$$(325)_{10} \Leftrightarrow (101000101)_2$$



### □ Sistemi di numerazione posizionali:

#### ▪ Sistema ottale

- $b=8; c_i \in \{0,1,\dots,7\}$
- Conversione binario  $\rightarrow$  ottale immediata essendo  $b$  una potenza di 2; si opera raggruppando i bit tre a tre partendo dalla virgola e sostituendo ciascuna terna con l'equivalente decimale rappresentato in ottale. Una traccia della dimostrazione:

$$I = c_s \cdot 2^s + c_{s-1} \cdot 2^{s-1} + \dots + c_0 \cdot 2^0 \Rightarrow \frac{I}{8} = c_s \cdot 2^{s-3} + c_{s-1} \cdot 2^{s-4} + \dots + c_3 + \frac{1}{8}(c_0 + c_1 \cdot 2^1 + c_2 \cdot 2^2) = \dots$$

$$\dots = Q + \frac{R}{8} \Rightarrow R = (c_0 + c_1 \cdot 2 + c_2 \cdot 4)$$

- $R$  è simultaneamente la cifra meno rappresentativa della rappresentazione ottale e l'equivalente decimale dei tre bit meno significativi della rappresentazione binaria
- Es:  $(01110111)_2 \Leftrightarrow (001\ 110\ 111)_2 \Leftrightarrow (167)_8$



### □ Sistemi di numerazione posizionali:

#### ▪ Sistema esadecimale

- $b=16; c_i \in \{0,1,\dots,9,A,B,C,D,E,F\}$
- Conversione binario  $\rightarrow$  esadecimale immediata essendo  $b$  una potenza di  $2$ ; si opera raggruppando i bit quattro a quattro partendo dalla virgola e sostituendo ciascuna quaterna con l'equivalente decimale rappresentato in esadecimale.
- Es:  $(10101001010111)_2 \Leftrightarrow (0010\ 1010\ 0101\ 0111)_2 \Leftrightarrow (2A57)_{16}$
- Notazione:  $0xAB$ ,  $ABh$

## Ordini di grandezza



$2^{10}$	1k	“kilo”
$2^{20}$	1M	“mega”
$2^{30}$	1G	“giga”
$2^{40}$	1T	“tera”

$10^3$	1k	“kilo”
$10^6$	1M	“mega”
$10^9$	1G	“giga”
$10^{12}$	1T	“tera”

- Lo stesso prefisso può riferirsi a valori diversi a in funzione del contesto nel quale è utilizzato
  - 1 kbyte =  $2^{10}$  byte = 1024 byte; [memory size]
  - 1 kbps =  $10^3$  bit/s = 1000 bit/s [transmission bit rate]
  - 1 kHz =  $10^3$  Hz = 1000 Hz [frequency]
- 1 byte = 8 bit

## Rappresentazione dei numeri relativi



- Esistono diverse modalità di rappresentazione
  - **Rappresentazione modulo e segno**
  - **Rappresentazione in complemento a 1**
  - **Rappresentazione in complemento a 2**
  - **Rappresentazione con eccesso  $2^{n-1}$  (polarizzata)**



### □ Rappresentazione modulo e segno [n bit]

- Alloca 1 bit per la rappresentazione esplicita del segno:  $1 \Leftrightarrow -$ ,  $0 \Leftrightarrow +$
- I restanti n-1 bit rappresentano il modulo del numero;
- Intervallo rappresentato:  $[-2^{n-1}+1, 2^{n-1}-1]$  (simmetrico)
- Es:  $n=7 \Rightarrow [-63, +63]$ ;  $(+55)_{10} \Leftrightarrow (0110111)_2$ ;  $(-55)_{10} \Leftrightarrow (1110111)_2$ ;
- Meccanismo semplice ma:
  - Doppia rappresentazione dello 0
  - Addizione e sottrazione non possono essere gestite con un approccio unificato [necessaria gestione esplicita del segno  $\Rightarrow$  circuiti più complessi]



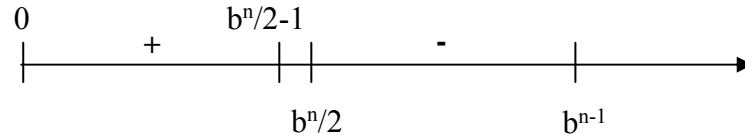
### □ Rappresentazione in complemento a 1 [n bit]

- Per cambiare di segno si complementa il numerale bit a bit [inversione bit a bit]
- I numerali positivi iniziano per 0 i negativi per 1
- Rappresentazione obsoleta
- Intervallo rappresentato:  $[-2^{n-1}+1, 2^{n-1}-1]$
- Es:  $n=7 \Rightarrow [-63, +63]$ ;  $(+55)_{10} \Leftrightarrow (0110111)_2$ ;  $(-55)_{10} \Leftrightarrow (1001000)_2$ ;
- Doppia rappresentazione dello 0



## □ Rappresentazione in complemento a 2 [n bit]

- Caso particolare di rappresentazione in complemento alla base  $b > 1$ 
  - numeri positivi, 0 compreso, sono rappresentati in  $[0, b^n/2 - 1]$
  - numeri negativi rappresentati in  $[b^n/2, b^n - 1]$ ; il generico numero negativo  $N$  viene rappresentato come  $b^n - |N| \Rightarrow$  l'estremo superiore della rappresentazione [sequenza di tutti 1] rappresenta sempre -1



Es. n=3

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

- Intervallo rappresentato:  $[-2^{n-1}, 2^{n-1} - 1]$  (asimmetrico)
- Es:  $n=7 \Rightarrow [-64, +63]$ ;
  - $(+55)_{10} \Leftrightarrow (0110111)_2$ ;
  - $(-55)_{10} \Rightarrow -55 + 128 = 73 \Rightarrow (1001001)_2$ ;



### □ Rappresentazione in complemento a 2 [n bit]

- Algoritmo per la complementazione rapida (in base 2) di una sequenza binaria:
  - Si ispeziona la sequenza a partire dal bit meno significativo fino ad individuare il primo 1
  - La sequenza complementata ha gli stessi bit di quella da originaria fino al bit (compreso) individuato nello step precedente
  - I restanti bit della sequenza complementata si ottengono da quella originaria invertendo bit a bit
  - Es:  $(+55)_{10} \Leftrightarrow (0110111)_2; \Rightarrow (-55)_{10} \Leftrightarrow (\mathbf{1001001})_2;$
- Vantaggi:
  - Rappresentazione unica dello 0
  - Operazioni aritmetiche semplificate;  $[X-Y=X+(-Y)]$



### □ Rappresentazione con eccesso a $2^{n-1}$ [n bit]

- Utilizza un offset per la rappresentazione;  $2^{n-1}$  costante di polarizzazione
- Un numero  $x$  viene rappresentato come  $x+2^{n-1}$
- I numeri dell'intervallo  $[-2^{n-1}, 2^{n-1}-1]$  sono rappresentati in  $[0, 2^n-1]$
- Rappresentazione unica dello 0; utilizzata in IEEE 754
- Es:  $n=4$ ;  $(-3)_{10} \Rightarrow -3+8=5 \Rightarrow (0101)_2$ ;  $(4)_{10} \Rightarrow 4+8=12 \Rightarrow (1100)_2$
- Algoritmo per la rappresentazione rapida con eccesso di una sequenza binaria
  - Si effettua un complementazione in base 2 preliminare
  - Viene invertito il bit più significativo della sequenza ottenuta
  - Es:  $n=4$ ;  $(3)_{10} \Leftrightarrow (0011)_2 \Rightarrow (-3)_{10} \Leftrightarrow (\mathbf{1101})_{CP2} \Leftrightarrow (\mathbf{0101})_{E8}$

## Operazioni aritmetiche



### □ Addizione

- $0+0=0$ ;
- $0+1=1$ ;
- $1+0=0$ ;
- $1+1=0$  (**1**) **riporto**
- Aritmetica non segnata: la condizione di trabocco nella somma (overflow) **coincide** con l'occorrenza di un riporto non nullo dalle 2 cifre più significative
- Esempio:
  - $n=5$ ; aritmetica non segnata
  - Oss.  $(39)_{10}$  non è rappresentabile con 5 bit anche in un aritmetica non segnata

(1)	(1)	(0)	(0)	(0)		← riporti
	0	1	0	1	1	+ (11) <sub>10</sub>
	1	1	1	0	0	= (28) <sub>10</sub>
	0	0	1	1	1	(39) <sub>10</sub>



### □ Addizione

- Aritmetica segnata: la condizione di trabocco nella somma (overflow) **non coincide** con l'occorrenza un riporto non nullo dalle 2 cifre più significative
- Si verifica una condizione di overflow **se e solo se** i due addendi hanno lo stesso segno ed il risultato ha segno diverso; si può dimostrare che questo equivale alla condizione di riporti diversi dalle 2 cifre più significative
- Il problema dell'overflow è una conseguenza dell'aver adottato numerali di lunghezza finita



## □ Addizione

- Esempi: n=8; aritmetica segnata

(1)	(1)	(1)	(0)	(1)	(1)	(1)	(0)		← riporti
	1	1	1	0	0	1	1	0	+ (-26) <sub>10</sub>
	1	1	1	0	1	0	1	1	= (-21) <sub>10</sub>
	1	1	0	1	0	0	0	1	(-47) <sub>10</sub>

## Riporto dalla cifra più significativa ma no overflow

(0)	(1)	(0)	(0)	(0)	(0)	(1)	(1)		← riporti
	0	1	1	0	0	0	0	1	+ (+97) <sub>10</sub>
	0	1	0	0	1	0	1	1	= (+75) <sub>10</sub>
	1	0	1	0	1	1	0	0	(-84) <sub>10</sub>

## No riporto dalla cifra più significativa ma overflow

## Rappresentazione dei numeri reali



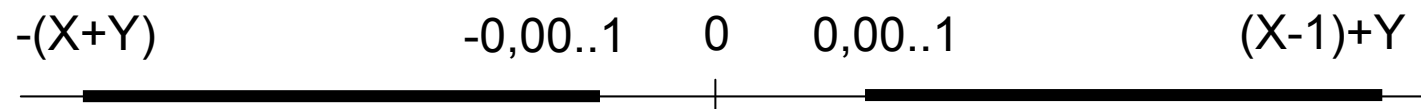
- Dato il problema: rappresentare un qualunque numero **reale** attraverso una sequenza **finita** di cifre
  - Approccio: si rappresenta un numero razionale che in generale approssima il numero dato
  - Utilizzeremo numerali di lunghezza finita: insieme dei numeri reali rappresentabili discontinuo ed ha cardinalità finita
- A seconda della gestione statica o dinamica della parte decimale nella rappresentazione:
  - Rappresentazione in virgola fissa
  - Rappresentazione in virgola mobile



### □ Rappresentazione in virgola fissa [fixed point]

- La parte intera e la parte decimale del numero sono rappresentate separatamente (n cifre, m cifre):
- $(I,F)_B$ ; I in CP alla base B
- Utilizzo non efficiente della capacità rappresentativa
- Risoluzione della rappresentazione:  $B^{-m}$
- Intervallo di rappresentabilità:

- siano  $X = B^{n-1}$  ed  $Y = B^{-1} + B^{-2} + \dots + B^{-m} = 1 - B^{-m}$



- Non adatta per calcoli tecnico – scientifici
- Es:  $n=4, m=5; (-7,8125)_{10} \Rightarrow (1001,1101)_2$



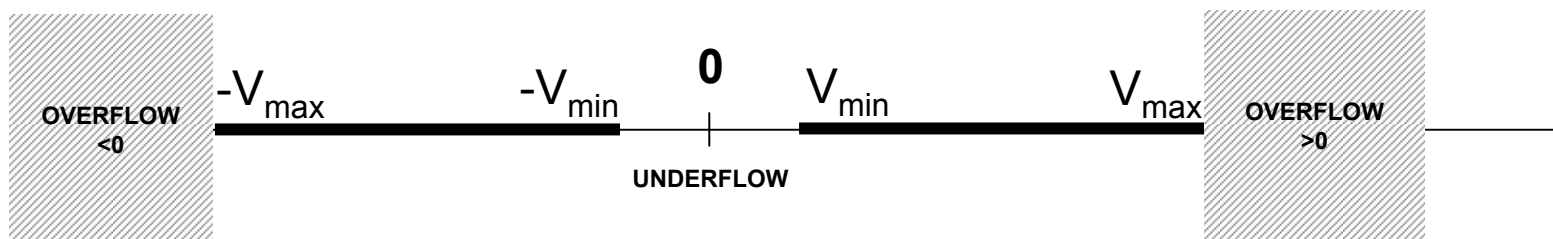
### □ Rappresentazione in virgola mobile [floating point]

- Si utilizza la notazione scientifica:  $V(N) = m \cdot B^e$   $m$  indica la mantissa, e l'esponente,  $B$  la base della rappresentazione
- $m \in Q, e \in Z \Rightarrow V(N) \in Q$
- Normalizzazione della mantissa necessaria
  - Univocità della rappresentazione
  - Efficienza della rappresentazione
  - $B^{-1} \leq m < 1$  [ $\Leftrightarrow m=0,xxx\dots$ ] oppure  $1 \leq m < B$  [ $\Leftrightarrow m=1,xxx\dots$ ]
  - $(325,33)_{10} \Leftrightarrow 0,32533 \cdot 10^3$
  - $(10101,110)_2 \Leftrightarrow 0,10101110 \cdot 2^{101}$
- Vantaggi rispetto a virgola fissa
  - Consente di rappresentare intervalli più grandi
  - Maggiore risoluzione



## □ Rappresentazione in virgola mobile [floating point]

- Diverse possibilità per la rappresentazione dei parametri  $m, e \Rightarrow$  diversi formati
- Espressione di riferimento per  $B=2$ :  $V(N) = (-1)^s \cdot \underline{m} \cdot 2^e$  dove  $s$  indica il segno della mantissa ( $s=0$  se  $m \geq 0$ ,  $s=1$  se  $m < 0$ ) ed  $\underline{m}$  indica la mantissa normalizzata
- Intervallo di rappresentazione:





### □ Rappresentazione in virgola mobile [floating point]

- Se si adottano  $m=0,xxx..$  rappresentato in modulo con 23 bit, e rappresentato con 8 bit in CP2, s rappresentato con 1 bit
  - $V(N) = (-1)^s \cdot 0,m \cdot 2^e$ ;  $V_{\min} = 0.1 \cdot 2^{-128}$ ,  $V_{\max} = 0.11...1 \cdot 2^{+127}$
  - Si osservi che  $m=0,1xx..$
  - **Esercizio: Calcolare la cardinalità dell'insieme di numeri razionali rappresentabili**
- ANSI / IEEE Std 754 -1985
  - come sopra ma con  $m=1,xx..$  ed  $e$  rappresentato in **eccesso  $2^{n-1}=128$**  [precisione singola]; la parte intera di  $m$  viene omessa dalla rappresentazione
  - Range [in decimale]:  $V_{\min} \cong 10^{-45}$   $V_{\max} \cong 10^{38}$  ; precisione circa 7 cifre decimali; alcune configurazioni per  $m, e$  corrispondono a valori speciali; 52 bit per  $m$  ed 11 per  $e$   
⇒ precisione doppia



### □ Rappresentazione in virgola mobile [floating point]

- Siano  $N_1, N_2$  due numeri consecutivi rappresentabili ed indicando con  $r$  il numero di cifre della mantissa:

$$N_1 = 2^e \cdot 0, m \quad N_2 = 2^e \cdot 0, (m \pm 2^{-r})$$

- Approssimazione di  $r \in \mathbb{R}$  con  $r' \in \mathbb{Q}$ :

- Errore assoluto:  $N_1 - N_2 = f(m, r)$

- Errore relativo:  $\frac{N_1 - N_2}{N_2} = \pm \frac{2^{-r}}{0, m} \cong 2^{-r} = f(r)$

- L'ordine di grandezza dell'errore assoluto dipende da  $m, r$  mentre quello dell'errore relativo solo da  $r$



### □ Rappresentazione in virgola mobile [floating point]

- Esempio: Fornire l'equivalente decimale del numerale **1-01111010-1011010...0** in formato IEEE 754.
  - Segno numero:  $<0$
  - Esponente:  $(01111010)_2 = (122)_{10} \Rightarrow e = 122 - \mathbf{128} = -6$
  - Mantissa:  $(0.101101)_2 = 0,703125 \Rightarrow m = 1,703125$
  - Numero:  $-1,703125 \cdot 2^{-6} = -0,026611328125$

## Operazioni sui numeri in virgola mobile



- Dati:  $\langle s_1, m_1, e_1 \rangle$  ,  $\langle s_2, m_2, e_2 \rangle$ 
  - Moltiplicazione / Divisione
    - Si opera nel modo ordinario sfruttando le proprietà delle potenze
    - Il risultato potrebbe dover essere normalizzato  $\Rightarrow$  shift della mantissa ed incremento / decremento dell'esponente
  - Somma / Sottrazione
    - Si opera nel modo ordinario avendo cura di allineare i due esponenti
    - L'esponente minore viene ricondotto a quello maggiore operando uno shift a dx della mantissa [perché non il contrario]
    - Il risultato potrebbe dover essere normalizzato

## Codici



□ Siano  $C' = \{e_1, e_2, \dots, e_N\}$  e  $C = \{0, 1\}^n$

$$f : C' \xrightarrow{\text{Codifica}} \{0, 1\}^n$$

□ Codifica binaria dell' insieme  $C'$

- $f$  iniettiva  $\Leftrightarrow$  codice non ambiguo
- $m = \min$  valore di  $n$  che consente codifica non ambigua
- per codici non ambigui possiamo definire l'operazione inversa rispetto alla codifica: decodifica
- Codici **irridondanti** ( $n=m$ ), **ridondanti** ( $n>m$ ), **ambigui** ( $n<m$ )
- Oss: un codice può risultare ambiguo anche se  $n \geq m$ ; codici ambigui non sono d'interesse
- **Distanza di Hamming** di un codice ( **$h$** ): numero minimo di bit ("distanza") per i quali differiscono 2 qualsiasi parole di codice al variare (in tutti i modi possibile) di queste ultime
- Oss: codici irridondanti  $\Rightarrow h=1$  (in generale non è vero il viceversa)

$$2^n \geq N \Rightarrow m = \lceil \log_2 N \rceil$$



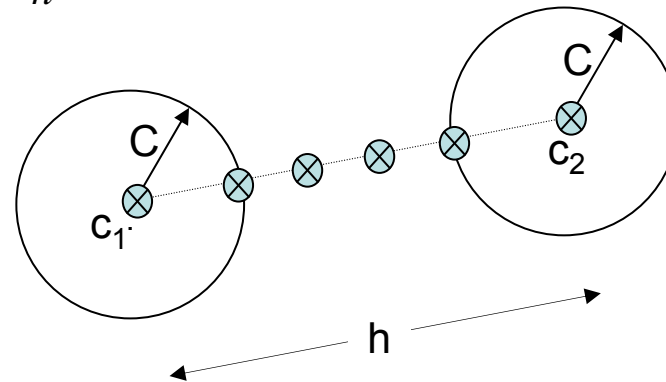
- Oss: codici ridondanti  $\Rightarrow h \geq 1$  , codici ambigui  $\Rightarrow h = 0$
- Esempi  $N=3$ ;  $n=3$ 
  - Codice ridondante  $h=1$ 
    - $e_1 \rightarrow 001$
    - $e_2 \rightarrow 010$
    - $e_3 \rightarrow 101$
  
  - Codice ridondante  $h=2$ 
    - $e_1 \rightarrow 001$
    - $e_2 \rightarrow 010$
    - $e_3 \rightarrow 111$
- **Esercizio: Quante sono le possibili codifiche binarie f ambigue e non ambigue dell'insieme  $C'$ ? [ $f: C' \rightarrow \{0,1\}^n$  ;  $|C'|=N$ ]**



- Codici ridondanti con  $h > 1$  hanno capacità di rilevazione / correzione d'errore

- **Capacità di rilevazione R:**  $C_1 \underbrace{x x}_{h} C_2$ 
  - $R = h - 1$

- **Capacità di correzione C:**
  - $h - 2C \geq 1 \Rightarrow C \leq \left\lfloor \frac{h - 1}{2} \right\rfloor$



- Valore min di  $h$  per avere  $C > 0 \Rightarrow h = 3$
- Le due “sfere” entro le quali vengono corretti errori (raggio  $C$ ) devono distare almeno 1 bit per evitare ambiguità in fase di decisione
- Decisione a massima verosimiglianza MLD

## Codici



### ❑ Codice BCD [Binary Coded Decimal]

- Codifica con 4 cifre binarie i numeri da 0 a 9
- Codice irridondante [ $n=m=4$ ]
- Es.  $(570)_{10} \Rightarrow (0101\ 0111\ 0000)_{BCD}$

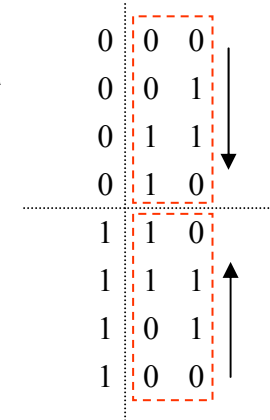
### ❑ Codici di GRAY

- Rappresenta numeri binari di  $n$  bit
- Riorganizza le sequenze in modo che sequenze successive differiscano per **un solo bit**
- Codice  $n=2$ : **00,01,11,10**; per  $n>2$ : tecnica “speculare” a partire dal codice  $n-1$

### ❑ Codici ASCII

- Codifica caratteri alfanumerici con 7 o 8bit
- 8° bit: parità o estensione set caratteri

Cifra	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001





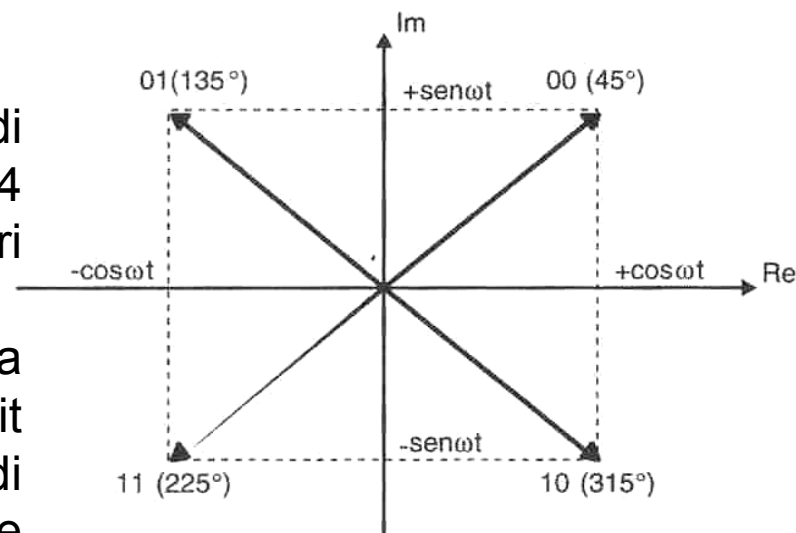
## □ Codice ASCII 7 bit

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>	64	40	100	&#64;	<b>@</b>	96	60	140	&#96;	<b>`</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>	65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>	66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>	67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>	68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>	69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>	70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>	71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>	72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>	73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>	74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>	75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>	76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>	77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>	78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>	79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>	80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>	81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>	82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>	83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>	84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>	85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	<b>6</b>	86	56	126	&#86;	<b>V</b>	118	76	166	&#118;	<b>v</b>
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	<b>7</b>	87	57	127	&#87;	<b>W</b>	119	77	167	&#119;	<b>w</b>
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	<b>8</b>	88	58	130	&#88;	<b>X</b>	120	78	170	&#120;	<b>x</b>
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	<b>9</b>	89	59	131	&#89;	<b>Y</b>	121	79	171	&#121;	<b>y</b>
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	<b>:</b>	90	5A	132	&#90;	<b>Z</b>	122	7A	172	&#122;	<b>z</b>
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	<b>;</b>	91	5B	133	&#91;	<b>[</b>	123	7B	173	&#123;	<b>{</b>
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<b>&lt;</b>	92	5C	134	&#92;	<b>\</b>	124	7C	174	&#124;	<b> </b>
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	<b>=</b>	93	5D	135	&#93;	<b>]</b>	125	7D	175	&#125;	<b>}</b>
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	<b>&gt;</b>	94	5E	136	&#94;	<b>^</b>	126	7E	176	&#126;	<b>~</b>
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	<b>?</b>	95	5F	137	&#95;	<b>_</b>	127	7F	177	&#127;	<b>DEL</b>



## □ Oss: Codici di GRAY

- Molto utilizzato nei formati di modulazione numerica [es. 4 PSK (Phase Shift Keying) 4 valori di fase]
- Obiettivo: simboli adiacenti nella codifica differiscono per 1 solo bit  $\Rightarrow$  maggiore probabilità di rilevazione / correzione d'errore usando criteri MLD



## □ Codice di parità

- Si ottiene a partire da un codice irridondante aggiungendo un bit di controllo [bit di parità]; codice ridondante  $h=2 \Rightarrow$  consente di rilevare la presenza di 1 errore nella sequenza
- Utilizza operatore somma diretta [o somma modulo 2]  $\oplus$  :  $0 \oplus 0 = 0$ ;  $0 \oplus 1 = 1$ ;  $1 \oplus 0 = 1$ ;  $1 \oplus 1 = 0$ ; “controllo di parità degli 1”
- Codice parità  $n=2$ ;  $00, 01, 10, 11 \Rightarrow 000, 011, 101, 110$



### □ Codici di Hamming

- Codici ridondanti con capacità di correzione ( $h>2$ )
- Famiglia di codici; consideriamo caso  $h=3$
- Parola di codice:  $m = n+k$  bit dove  $n$  indicano i bit del codice irridondante di partenza e  $k$  i **bit di controllo**
- $2^k$  possibili configurazioni dei bit di controllo (**sindromi**)
- Vale la relazione:  **$n+k+1 \leq 2^k$**
- Sia  $c_1c_2c_3c_4\dots c_m$  la generica parola di codice:
  - si esprimono in binario gli indici  $1,2,\dots,m$
  - i bit di controllo si trovano in posizione  $2^i$  ( $i=0,1,2,\dots,k-1$ )
  - si considerano i bit il cui indice (in binario) contiene 1 in posizione  $2^i$  ( $i=0,1,2,\dots,k-1$ ): il bit di controllo in posizione  $2^i$  controlla la parità di questi bit
- In ricezione si calcola un numero di controllo  $N_c$  di  $k$  bit il cui bit  $j$  controlla la parità dei bit della parola ricevuta nelle posizioni di indice  $2^j$  ( $j=0,1,2,\dots,k-1$ ) [analogamente al punto precedente]
- Se  $N_c=0$  [la parola di codice ricevuta è ritenuta corretta]
- Se  $N_c=E \neq 0$  [si assume l'inversione del bit in posizione  $E$ ]

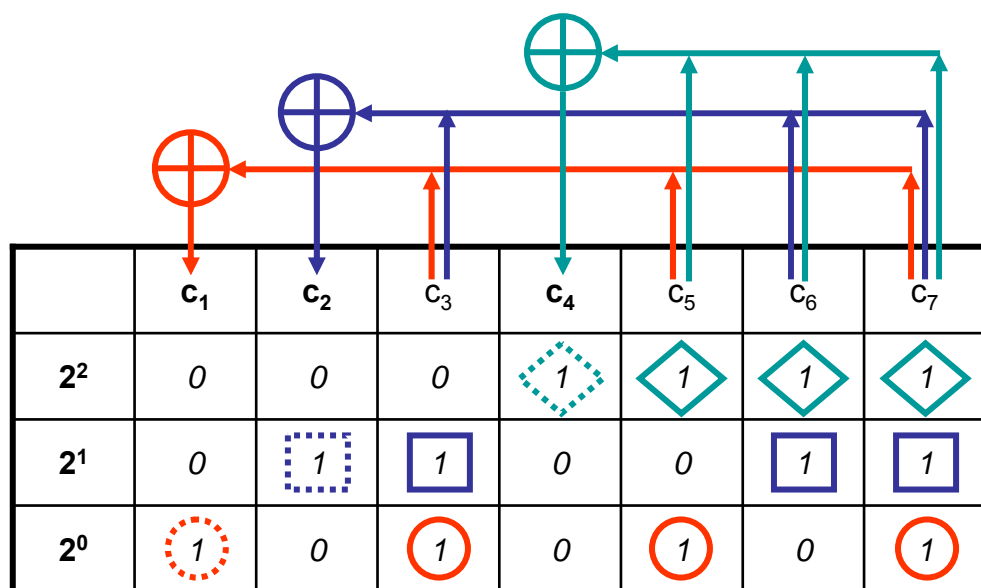
# Codici



## □ Codici di Hamming:

- Esempio: Trasformare codice BCD in un codice di Hamming a distanza 3;
- $n=4$ ; da  $n+k+1 \leq 2^k \Rightarrow k=3 \Rightarrow m=7$

Cifra	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001



⇒

$$\begin{aligned}
 & C_1 \ C_2 \ C_3 \ C_4 \ C_5 \ C_6 \ C_7 \\
 & C_1 = C_3 \oplus C_5 \oplus C_7 \\
 & C_2 = C_3 \oplus C_6 \oplus C_7 \\
 & C_4 = C_5 \oplus C_6 \oplus C_7
 \end{aligned}$$



## ❑ Codici di Hamming: Esempio (continua)

$$C_1 \ C_2 \ C_3 \ C_4 \ C_5 \ C_6 \ C_7$$

$$C_1 = C_3 \oplus C_5 \oplus C_7$$

$$C_2 = C_3 \oplus C_6 \oplus C_7$$

$$C_4 = C_5 \oplus C_6 \oplus C_7$$

<b>C<sub>1</sub></b>	C <sub>3</sub>	C <sub>5</sub>	C <sub>7</sub>
0	0	0	0
1	0	0	1
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1

<b>C<sub>2</sub></b>	C <sub>3</sub>	C <sub>6</sub>	C <sub>7</sub>
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1

<b>C<sub>4</sub></b>	C <sub>5</sub>	C <sub>6</sub>	C <sub>7</sub>
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1
0	1	1	0
1	1	1	1
0	0	0	0
1	0	0	1



Cifra	BCD	Hamming h=3
0	0000	<b>0000000</b>
1	0001	<b>1101001</b>
2	0010	<b>0101010</b>
3	0011	<b>1000011</b>
4	0100	<b>1001100</b>
5	0101	<b>0100101</b>
6	0110	<b>1100110</b>
7	0111	<b>0001111</b>
8	1000	<b>1110000</b>
9	1001	<b>0011001</b>



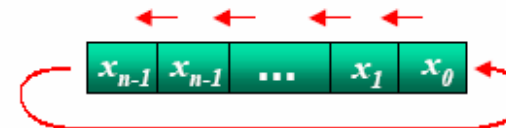
- **Codici di Hamming: Esempio (continua):**
  - Sequenza ricevuta: **0101010**  $\Rightarrow N_c=000 \Rightarrow$  **si ritiene** che la trasmissione sia avvenuta senza errori  $\Rightarrow$  Messaggio interpretato: 0010
  - Sequenza ricevuta: **0101011**  $\Rightarrow N_c=111 \Rightarrow$  **si ritiene** che sia avvenuto un errore durante la trasmissione che abbia provocato l'inversione del bit  $c_7 \Rightarrow$  Messaggio interpretato: 0010
  - **Oss:  $N_c=0 \Leftrightarrow$  assenza di errori?** (MLD....)
  - Questi codici sono adatti a scenari nei quali gli errori sono distribuiti casualmente sul flusso binario
    - non adatti alla trasmissione dati [errori a "burst"] a meno di introdurre interleaving
    - possono essere utilizzati nelle memorie

$$\begin{array}{cccccccc}
 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
 N_{c1} & = & c_1 \oplus c_3 \oplus c_5 & \oplus & c_7 \\
 N_{c2} & = & c_2 \oplus c_3 \oplus c_6 & \oplus & c_7 \\
 N_{c3} & = & c_4 \oplus c_5 \oplus c_6 & \oplus & c_7
 \end{array}$$



## □ Codici ciclici o polinomiali:

- Interpretano il messaggio da trasmettere ( $n$  bit) come un polinomio  $X(\lambda)$  di grado  $n-1$  [si lavora in aritmetica modulo 2 ( $\oplus$ ); campi di Galois]
- Sono così definiti [ciclici] in quanto ogni scorrimento ciclico di una generica parola di codice genera un'altra parola di codice
- Hanno capacità di correzione / rilevazione d'errore; in particolare sono in grado di rilevare un gran numero di errori introducendo una ridondanza relativamente bassa
- I codici si differenziano per il polinomio generatore  $G(\lambda)$  di grado  $k$  che li caratterizza [polinomio generatore del codice]
- La parola di codice da trasmettere ha  $n+k$  bit dove  $k$  sono i bit di controllo ed è ottenuta a partire da  $X(\lambda)$  e  $G(\lambda)$
- Es.  $G(\lambda) = \lambda^{16} + \lambda^{12} + \lambda^{15} + 1$  [rileva fino a 16 errori a raffica...]
- Codici a ridondanza ciclica [CRC]

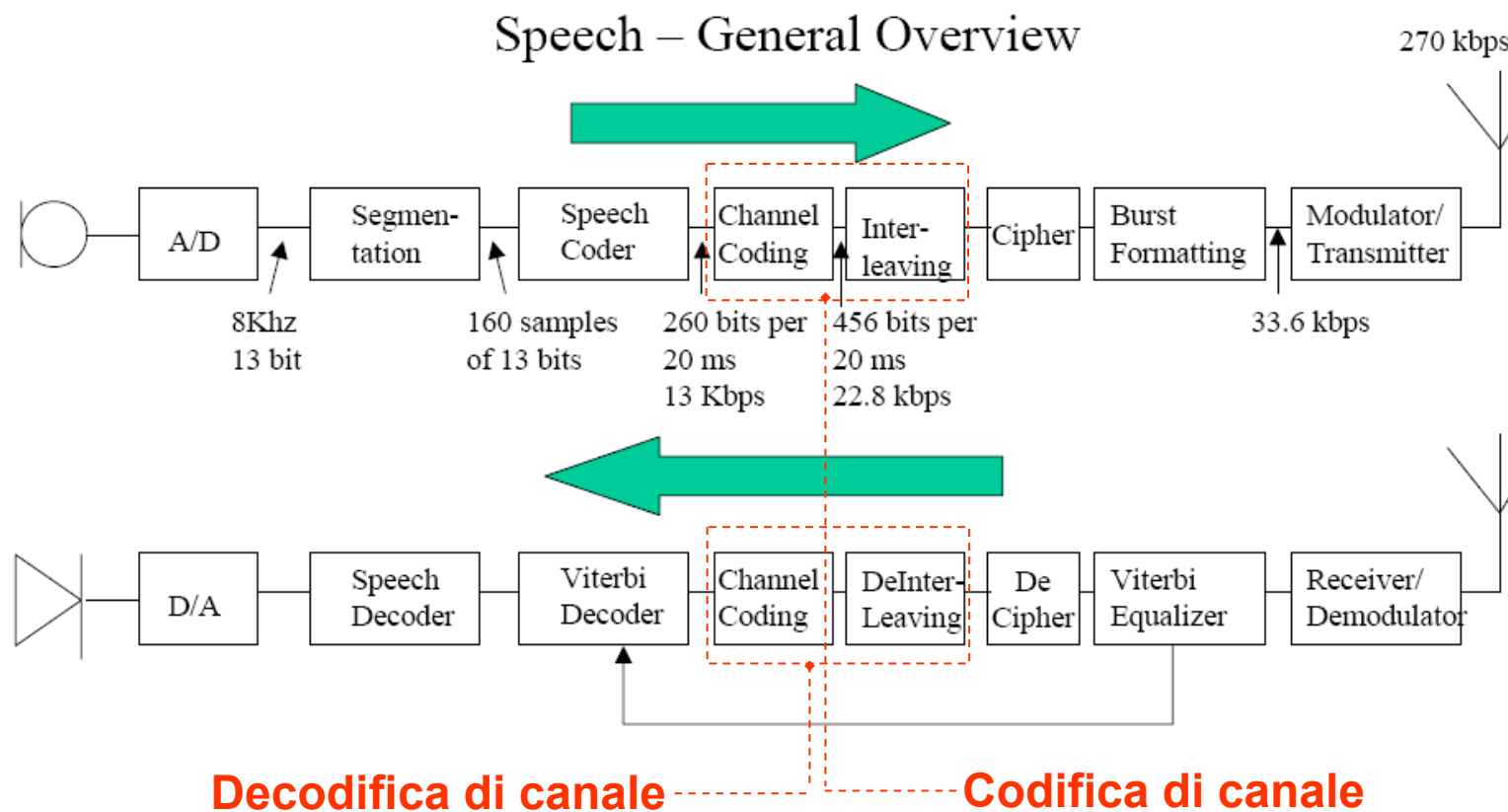




### □ **Codici convoluzionali:**

- A differenza dei precedenti non sono codici a blocchi; sono anche detti codici con memoria
- La sequenza binaria corrente in uscita dal codificatore dipende dalla sequenza binaria corrente in **input** e dallo **stato** corrente del codificatore
- Il generico bit  $m_i$  del messaggio in input al codificatore influenza l'output corrente e "futuro" dello stesso entro un certo numero di bit
- Il codificatore è una rete sequenziale
- Sono in grado di rilevare / correggere errori
- Es. Codifica di canale GSM

- Esempio: GSM transmission layer:



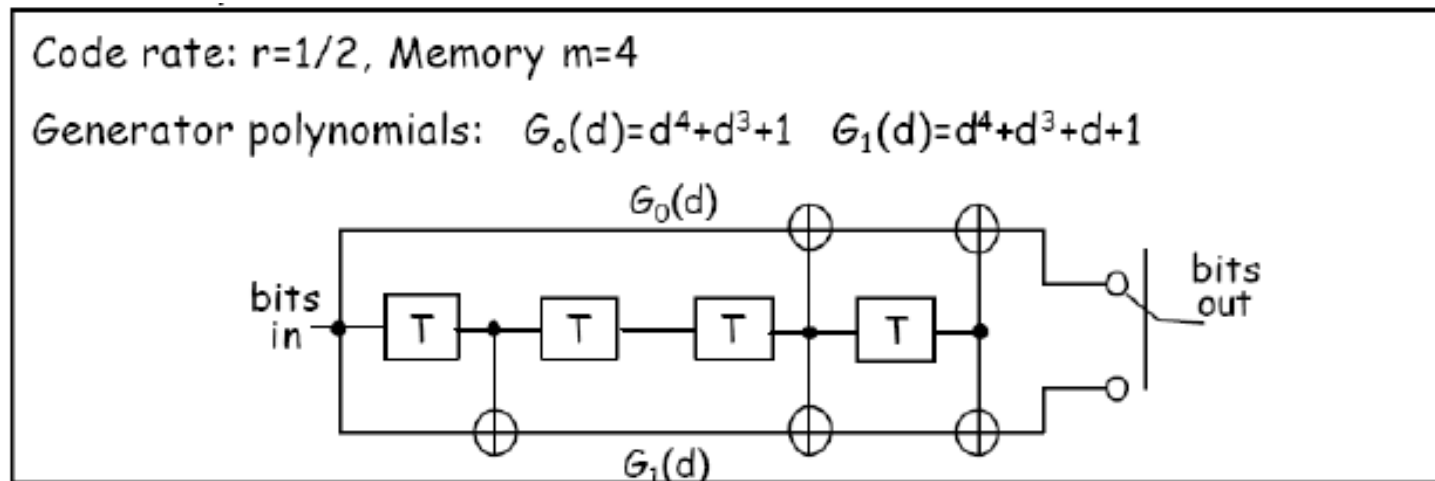


- Esempio: GSM speech encoding:
  - Campionamento a 8kHz [8000 campioni/sec; 160 campioni ogni 20 ms]
  - Ogni campione è quantizzato con codifica lineare a 13 bit  $\Rightarrow$   $13 \cdot 160 = 2080$  bit / 20ms
  - Codifica lineare a 3 stadi
    - RPE – LTP - LPC: Regular Pulse Excitation - Long Term Prediction - Linear Predictive Coder
    - I 2080 bit sono compressi in 260; 260 bit / 20ms  $\Rightarrow$  13kbps
  - Codifica di canale:
    - Outer code: codifica a blocchi
      - canali voce + alcuni di segnalazione: codice di parità
      - restanti canali di segnalazione: codice ciclico per rilevazione e correzione d'errore;  $G(\lambda) = (\lambda^{23} + 1)(\lambda^{17} + \lambda^3 + 1)$ ; ridondanza 40 bit
    - Inner code: codifica convoluzionale [correzione d'errore]
    - Interleaving: introduce robustezza rispetto ad errori a “burst”

## Codici



- ❑ Codificatore convoluzionale GSM:



## □ Interleaving:

- Tecnica che prevede la distribuzione non seriale delle parole di codice nel flusso binario utilizzando:
  - time spreading
  - merging
- L'occorrenza di errori a "burst" si traduce in una distribuzione uniforme degli errori su diverse parole di codice
- Aumenta il tempo di trasmissione di una parola di codice
- Molto utilizzata nei protocolli FEC [Forward Error Correction]

