Tipi di codifica

- ASCII --> 2^8 = 256
  - Lettere normali
- Unicode --> 2^16 = 65536
  - Lettere normali + caratteri di qualsiasi tipo (alfabeti)

Codici pesati

C O D I C E	BIT	P E S I
Binario puro	n	Binari
BCD o 8421	4	8-4-2-1
Aiken o 2421	4	2-4-2-1
Quinario	4	5-4-2-1
Due su Cinque	5	6-3-2-1-0
Biquinario	2+5	5-0 4-3-2-1-0

Esempio:

BIT	b_3	b_2	b_1	b_0
PESI	2^3	2^2	2^1	2^0
	8	4	2	1

Codice BCD		
Dec.	**8421**	**Hex**
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
	1010	*A*
	1011	*B*
	1100	*C*
	1101	*D*
	1110	*E*
	1111	*F*

Come si vede, il codice BCD ricalca il binario assoluto e la cifra decimale corrispondente ad una configurazione BCD è data dalla rappresentazione polinomiale:

$$b_3*8 + b_2*4 + b_1*2 + b_0*1 = N_{10}$$

Restando possibili: 8+4+2+1+1=16 rappresentazioni (2^{16}, come c'era da aspettarsi).

CODICE AIKEN O 2421

Il codice Aiken, analogamente al BCD, è un codice a lunghezza fissa di 4 bit, cui però si attribuiscono i pesi: 2-4-2-1, da cui pure il nome di codice 2421.

BIT	b_3	b_2	b_1	b_0	**Codice AIKEN**	
					Dec.	**2421**
PESI	2	4	2	1	0	0000
					1	0001
					2	0010
					3	0011
					4	0100
					5	1011
					6	1100
					7	1101
					8	1110
					9	1111

Siccome tale codifica rende possibili: 2+4+2+1+1=10 configurazioni, essa si rivela utile, in pratica, solo per codificare le 10 cifre decimali, secondo la relazione:

$$b_3*2 + b_2*4 + b_1*2 + b_0*1 = N_{10}$$

Esempio

$$1011_{Aiken} = 1*2 + 0*4 + 1*2 + 1*1 = 5_{10}$$

--> Per arrivare al codice a barre

Codici non pesati

N U M E R I C I

C O D I C E	**BIT**
Gray	n
Eccesso 3	4
Eccesso 3 Riflesso	4
BCD di Petherick	4

CODICE ECCESSO 3

Il codice Eccesso 3 è un codice non pesato a 4 bit derivato dal BCD sommando 3 ad ogni configurazione BCD.

Codice Dec.	**Eccesso 3**
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Il codice è **autocomplementante** e **mancante della configurazione nulla** (e della 1111). Questa particolarità consente di evitare ambiguità (per i circuiti di decodifica) tra l'infor_ mazione corrispondente ad 1 stringa nulla (tipo 0000) e l'assenza di segnale.

La scelta dell'eccesso 3, rispetto, p.es., al 2 o 1, che garantirebbero ancora la condizione di segnale per la cifra 0, è preferita in quanto rende il codice autocomplementante.

CODICE ECCESSO 3 RIFLESSO

Si ottiene dal codice Eccesso 3 tramite operazioni di shift a destra e somme senza riporto, come mostrato nell'esempio:

Dec.	Eccesso 3	Shift a Dx	Somma
3	0110	0011	0110+
			0011=

			0101 <--- Eccesso 3 riflesso

Eccesso 3 Riflesso

Dec.	Ecc.3 R.
0	0010
1	0110
2	0111
3	0101
4	0100
5	1100
6	1101
7	1111
8	1110
9	1010

Come si può notare dalla tabella, il codice, come il Gray, è progressivo.

CODICE BCD DI PETHERICK

Questo codice, non pesato, presenta sia la caratteristica di essere progressivo che quella di non contenere le configurazioni 0000 e 1111.

Codice BCD PETHERICK

Dec.	Petherick
0	0010
1	0110
2	0100
3	0100
4	0001
5	1001
6	1101
7	1100
8	1110
9	1010

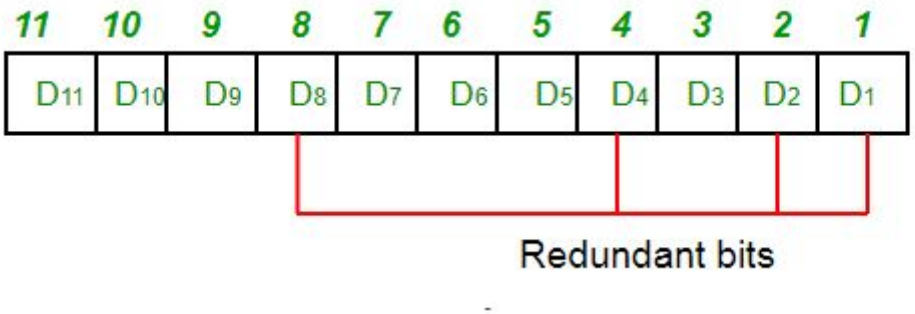
Riferimenti delle due codifiche:
parte5.pdf (scov8.altervista.org)

Dispensa:
https://s16e06bdd4ee29d21.jimcontent.com/download/version/1614442961/module/14667421025/name/Dispensa%20Sistemi%20di%20codifica.pdf

Rilevazione e correzione di errori

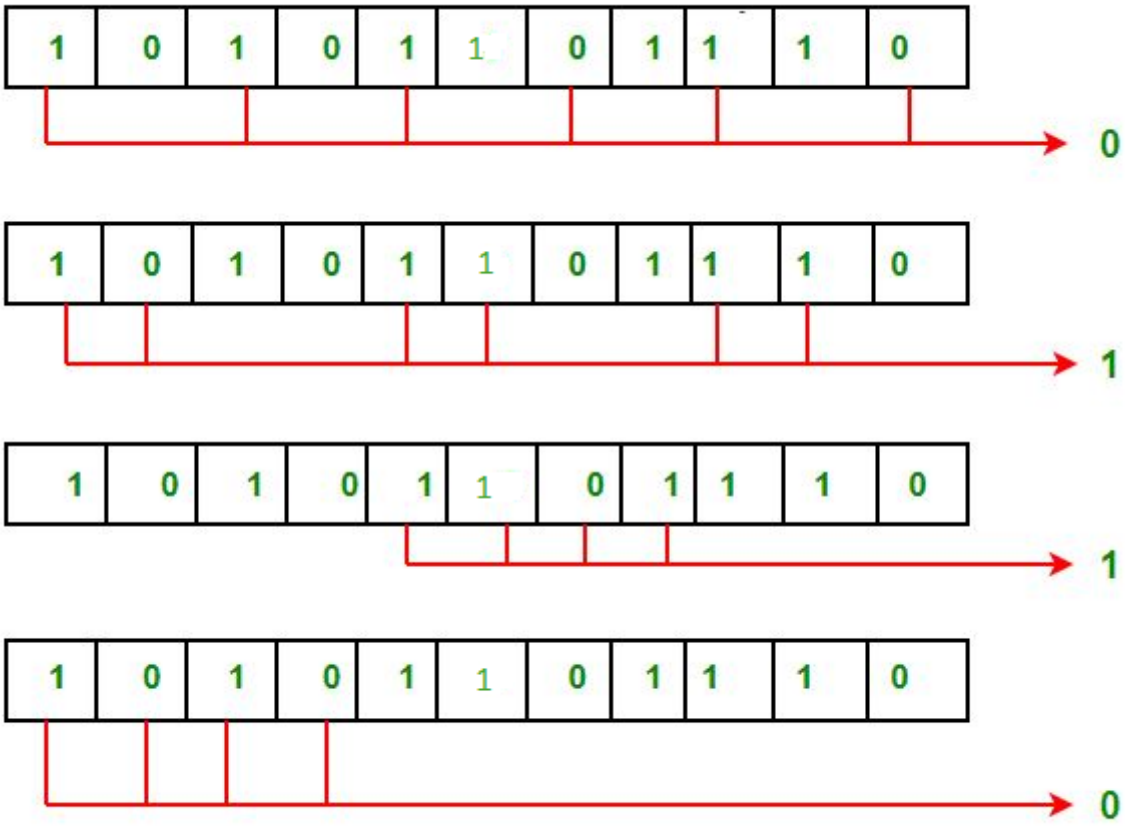
Codifica di Hamming

- Aggiunta dei bit ridondanti (la loro somma deve essere pari - esempio, n. di bit ad 1 deve essere pari per fare)



Ad ogni posizione controlliamo se la somma del numero di bit a potenze adi 2 (D_2, D_4, D_8) somma esattamente ad 1

Arrivata la trasmissione:



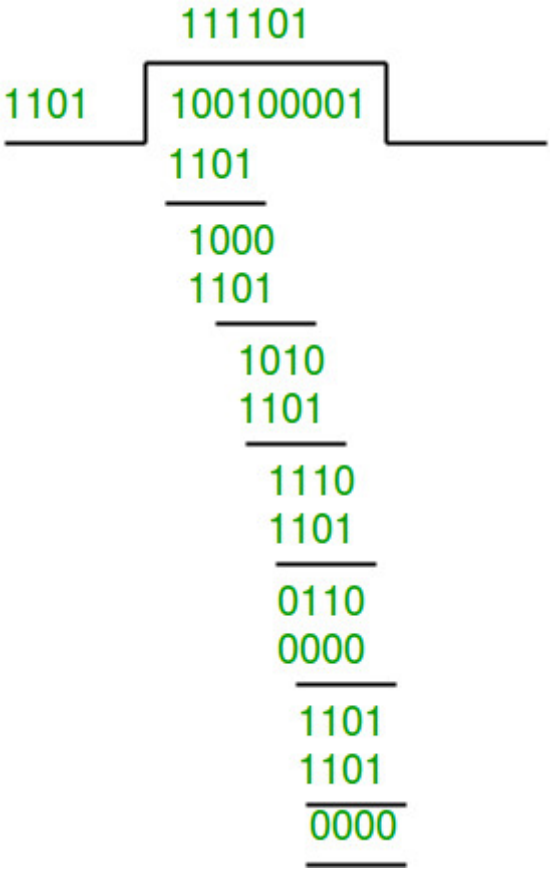
0 = no errori (bit pari)
1 = sì errori (bit dispari --> in quella posizione c'è stato un errore)

Tramite XOR usando i bit finali si verifica se si riottiene la codifica iniziale.

CRC - Cyclic redundancy check

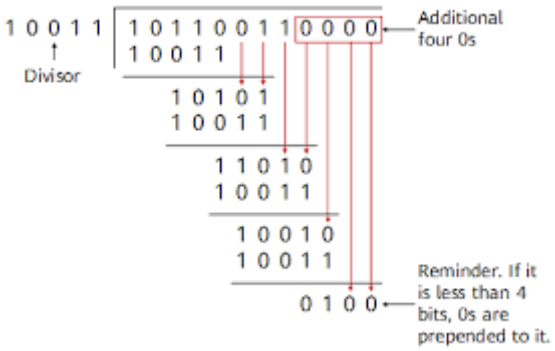
- Somma di bit di polinomi
- Se resto 0 alla fine (usando la XOR), allora nessun errore in trasmissione
  - Questo tramite divisione "modulo 2"

Assumendo il dato come:



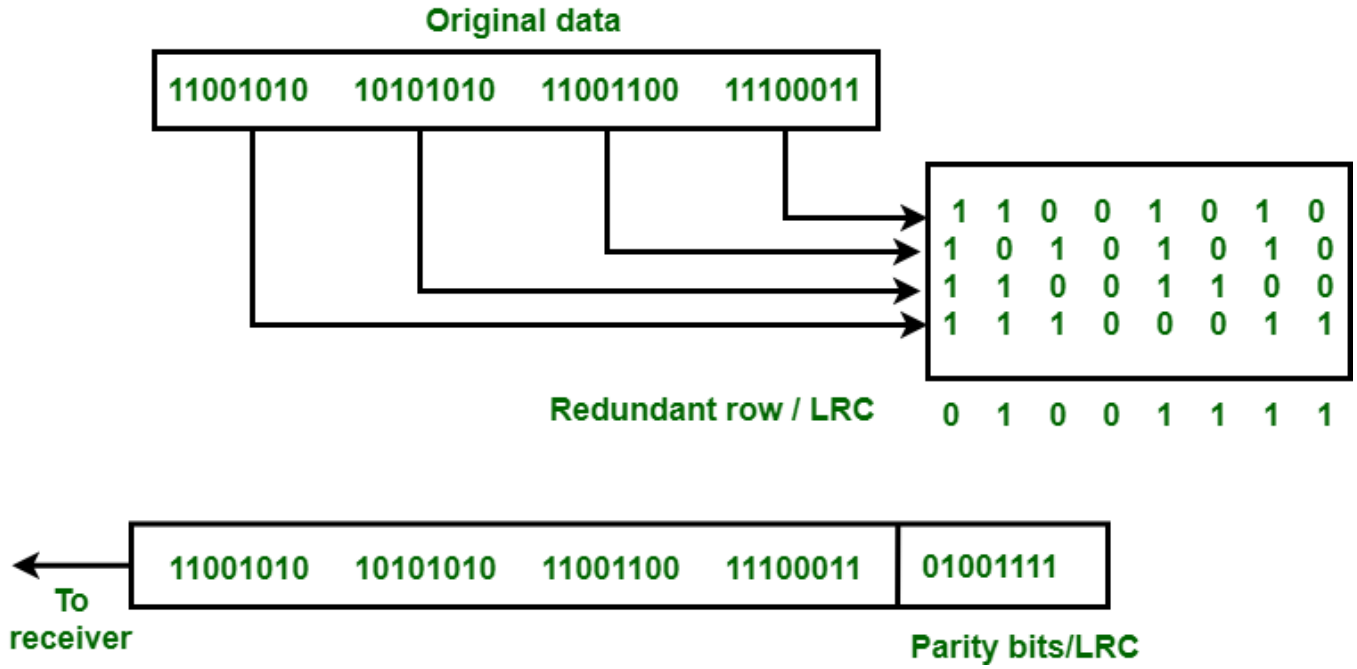
Se resto 0 = allora tutto bene

Altro esempio (stesso concetto, ma con bit aggiunti):

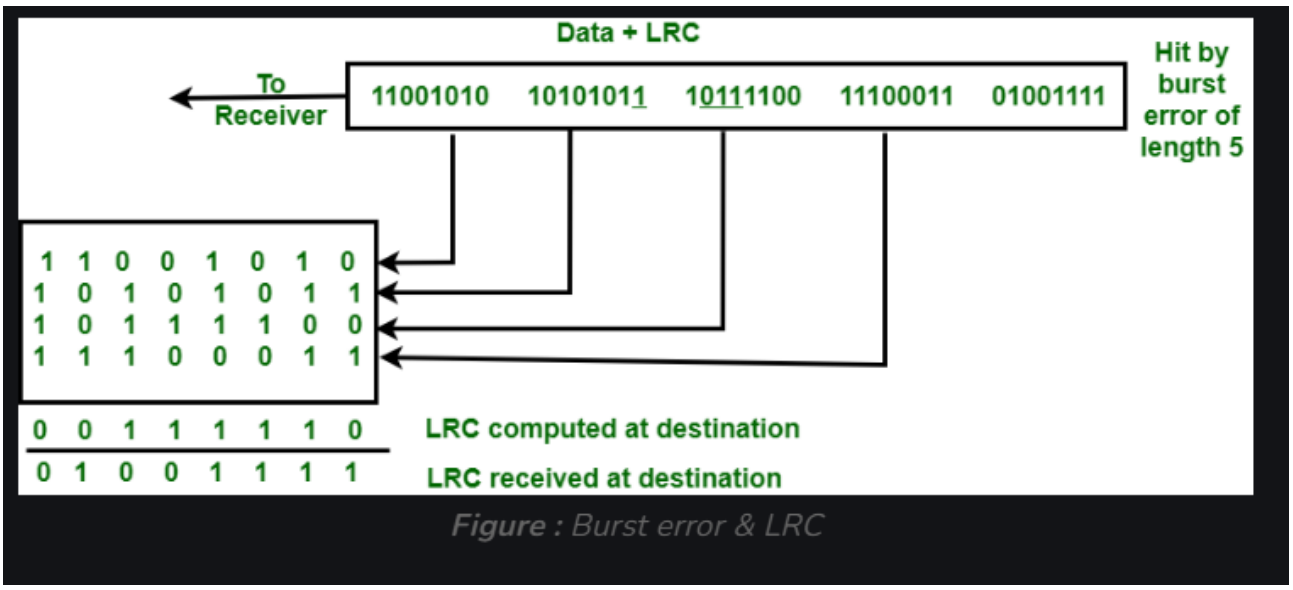


LRC (Longitudinal Redundancy Check)

- Prendo il dato come blocchi/matrice di insiemi di bit
- Aggiungo un bit ridondante per riga
- Questo lo aggiunge ad ogni blocco
- Trasmetto la matrice col dato



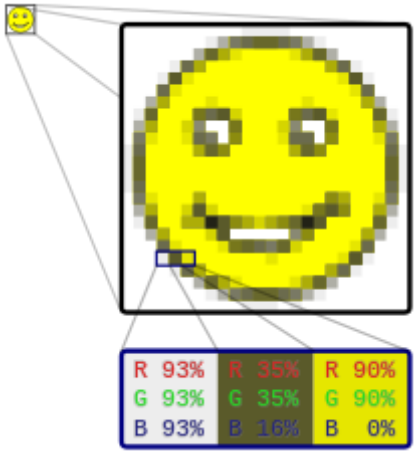
- Trova errori in burst (= 5 bit errati = trova bit errati in blocco)



Codifica multimediale

Immagini

- Raster
  - Ad ogni bit associata un informazione
  - Salvo 256 colori --> 2^8
    - Palette di colori
  - Uso Hamming e metodi "semplici" per correggere gli errori



- Risoluzione
  - Numero di pixel moltiplicato per il numero di colonna
- Profondità
  - Colori e numero di bit per colore

A seconda del tipo di colore, usiamo una codifica diversa:

Red/Green/Blu
Cyan/Magenta/Black/Yellow

Trasparenza = Dipende dal tipo di codifica immagine usato

- I bit "riflettono" il colore dato (es. png)

Immagini vettoriali (svg)

- No perdita di qualità ingrandendo
- Piccola taglia nel file finale
- Compressione efficiente

Compressione con perdita (lossy)

- JPG/MP3

Compressione senza perdita (lossless)

- PNG/SVG/MP4

Densità = Numero di pixel per inch (pollice)

- DPI (Dots) indicano quanti punti sono necessari se messi uno accanto all'altro dalla stampante per ogni pollice
- PPI (Pixels) invece sono relativi al mondo digitale e indicano i pixel e la loro riproduzione

Multimedia

Multimedia = Insieme di informazioni in un formato (audio/video, etc.)

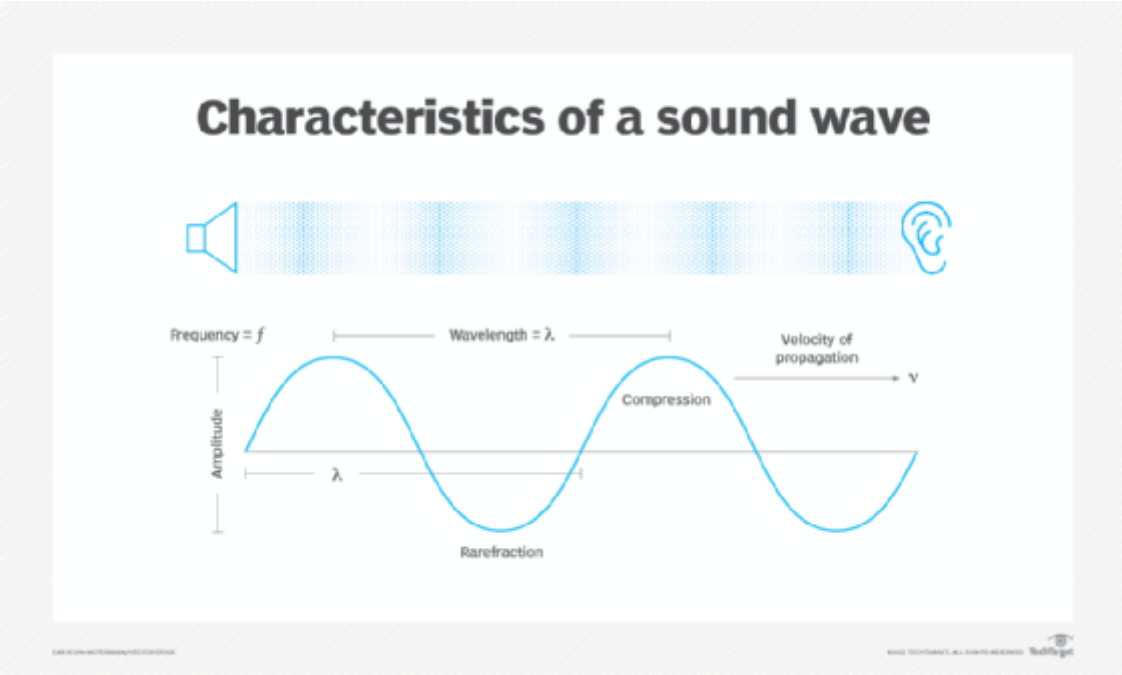
Frame / Fotogrammi = Insieme di immagini

- FPS = Frame per Second

Formati:

- AVI
  - Formato più vecchio usato in codifica di video
- MPEG
  - Compressione con perdita
  - Serve sia per video (4) che per audio (3)
- MOV
  - Sistemi Apple = MP4

Rappresentazione del suono:



- Rappresentabile in un tempo finito
  - Periodo
  - Frequenza = Quante oscillazioni ci sono
- Compressione sonora
  - = Campionamento
  - Prendo una serie di bit e codifico in un formato
  - Bitrate = Numero di bit per secondo
- Quantizzazione
  - Passaggio da continuo a discreto
  - = Prendo solo alcuni punti e uso quelli per rappresentare il segnale

MIDI

- Salvataggio tracce musicali per composizione colonne sonore/strumenti di vario genere
- Tastiere ma anche tutti gli altri strumenti

MP3

- Codifica audio a poco peso
- = Grosso salvataggio di dati in fase di quantizzazione

Compressione dei dati

Lossy = Con perdita

- Metodi di calcolo nella compressione del dato = motivo di perdita dati
- Facendo tante volte la compressione, si "vedono" le imperfezioni nell'immagine

Lossless = Senza perdita

- I file tendenzialmente sono più pesanti
- Puoi "rifare" la compressione

Esempio compressione lossy: Huffman

- Codifica ottima (con perdita)
- Usando pochi bit riesco facilmente a trovare l'errore
- Ordino i bit per frequenza
    - I rami più bassi dell'albero sono i meno frequenti
- La somma di tutti i nodi = tutti i bit venendo su = hanno lo stesso valore
- Contro: essere usato solo per file piccoli