# Codici

Somma in codice BCD:

Codificare i seguenti numeri in Cod. BCD ed eseguire l’operazione

47 + 35

I numeri vengono codificati cifra per cifra ed ogni cifra è codificata a 4 bit

0100\_0111 + 0011\_0101

Le operazioni vengono eseguite sommando i bit a 2 a 2 e considerando un eventuale riporto (scritto in rosso). Inoltre se un numero esce dal codice a questo va sommato il valore 6 (0110)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |  | 1 | 1 | 1 |  |  |
| 0 | 1 | 0 | 0 | \_ | 0 | 1 | 1 | 1 | + |
| 0 | 0 | 1 | 1 | \_ | 0 | 1 | 0 | 1 |  |
|  |  |  |  |  | 1 | 1 | 0 | 0 | + |
|  |  |  |  |  | 0 | 1 | 1 | 0 |  |
| 1 | 0 | 0 | 0 | \_ | 0 | 0 | 1 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | 8 |  |  |  |  | 2 |  |  |

Il risultato (1000\_0010) rappresenta correttamente il numero 82.

Somma e Sottrazione in codice ecc3:

Codificare i seguenti numeri in Cod. Ecc3 ed eseguire l’operazione

47 + 35

I numeri vengono codificati cifra per cifra ed ogni cifra è codificata a 4 bit

0111\_1010 + 0110\_1000

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |  | 0 | 0 | 0 |  |  |
| 0 | 1 | 1 | 1 | \_ | 1 | 0 | 1 | 0 | + |
| 0 | 1 | 1 | 0 | \_ | 1 | 0 | 0 | 0 |  |
| 1 | 1 | 1 | 0 | \_ | 0 | 0 | 1 | 0 |  |
| 1 | 1 | 0 | 1 | \_ | 0 | 0 | 1 | 1 |  |
| 1 | 0 | 1 | 1 | \_ | 0 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | 8 |  |  |  |  | 2 |  |  |
|  |  |  |  |  |  |  |  |  |  |

Effettuata la codifica dei singoli “digits” si effettua la somma con tutti i vari riporti evidenziati in rosso.  
Le due cifre ottenute devono però venir corrette aggiungendo 0011 se c’è stato un riporto sull’ultimo bit o sottraendo 0011 se il riporto non c’è stato. Si Noti che “sottrarre 0011” equivale ad “aggiungere 1101” e trascurare un eventuale riporto sulla cifra più significativa.

Codificare i seguenti numeri in Cod. Ecc3 ed eseguire l’operazione

47 - 35

I numeri vengono codificati cifra per cifra ed ogni cifra è codificata a 4 bit

0111\_1010 - 0110\_1000

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | \_ | 1 | 0 | 1 | 0 | - |
| 0 | 1 | 1 | 0 | \_ | 1 | 0 | 0 | 0 |  |

Per eseguire la sottrazione si ricorre al “Complemento a 10” del secondo numero completo che si ottiene complementando i vari “digits” del secondo numero e sommando 1 (ovvero in pratica eseguendo l’operazione:

47 + (100 – 35) = 47 + 64 + 1.

Nell’operazione di somma che segue se il risultato ha un riporto sul bit più significativo della parola intera (ovvero il risultato eccede il valore “cento”) si trascura tale ultimo riporto ottenendo quindi il valore corretto dopo aver effettuato la correzione (di aggiungere 0011 o 1101 a seconda del riporto presente sull’ultimo bit). Se viceversa non c’è un riporto al di fuori della parola intera (ovvero il risultato fosse inferiore a cento) significa che il risultato è negativo ed esso va pertanto ulteriormente complementato a 10)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 |  | 1 | 1 | 1 |  |  |
|  | 0 | 1 | 1 | 1 | \_ | 1 | 0 | 1 | 0 | + |
|  | 1 | 0 | 0 | 1 | \_ | 0 | 1 | 1 | 1 | + |
|  |  |  |  |  |  |  |  |  | 1 |  |
|  | 0 | 0 | 0 | 1 | \_ | 0 | 0 | 1 | 0 |  |
|  | 0 | 0 | 1 | 1 | \_ | 0 | 0 | 1 | 1 |  |
|  | 0 | 1 | 0 | 0 | \_ | 0 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | 1 |  |  |  |  | 2 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

Si noti che l’operazione appena svolta NON è formalmente corretta:

Infatti nell’operazione di “Complemento a 10” l’operazione di “sommare +1” dovrebbe essere fatta all’interno del codice ecc3 nel quale si sta per l’appunto operando. Così facendo l’operazione corretta richiederebbe di sommare 0011\_0100 e successivamente correggere i risultati “sottraendo 0011” da entrambi i termini (si è certi infatti che in questo caso NON ci sarà riporto al digit successivo), il che all’atto pratico coincide con sommare 0000\_0001 (come è stato fatto qui sopra)

Ad Esempio: Codificare i seguenti numeri in Cod. Ecc3 ed eseguire l’operazione

35 – 47

I numeri vengono codificati cifra per cifra ed ogni cifra è codificata a 4 bit

0100\_1000 – 0111\_1010

L’operazione di sottrazione si fa ricorrendo al complemento a 10 ovvero complementando ogni singolo bit della seconda parola (ciò grazie al fatto che il codice di AIKEN è autocomplementante) ed aggiungendo 1 In pratica l’operazione svolta viene ad essere :

35 + (100 – 47) = 35 + 52 + 1

Successivamente si effettua la somma e la correzione dei risultati sommando 0011 se vi è stato un riporto oppure sottraendo 0011 se il riporto non c’è stato. (In alternativa si può sommare 1101 e trascurare il riporto sul bit più significativo).

Poiché nella parola finale non vi è stato riporto alcuno sul digit più significativo , ovvero il risultato risulta composto dallo stesso numero di cifre degli operatori iniziali, si desume che il risultato sia minore di 100 e pertanto negativo, quindi per ottenere i valori corretti dei singoli digits essi vanno nuovamente complementati a 10.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |  | 0 | 0 | 1 |  |  |
| 0 | 1 | 1 | 0 | \_ | 1 | 0 | 0 | 0 | + |
| 1 | 0 | 0 | 0 | \_ | 0 | 1 | 0 | 1 | + |
|  |  |  |  |  |  |  |  | 1 |  |
| 1 | 1 | 1 | 0 | \_ | 1 | 1 | 1 | 0 | + |
| 1 | 1 | 0 | 1 | \_ | 1 | 1 | 0 | 1 |  |
| 1 | 0 | 1 | 1 |  | 1 | 0 | 1 | 1 |  |
| complemento a 10 | | | | | |  |  |  |  |
| 0 | 1 | 0 | 0 |  | 0 | 1 | 0 | 0 |  |
|  |  |  |  |  |  |  |  | 1 |  |
| 0 | 1 | 0 | 0 |  | 0 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | 1 |  |  |  |  | 2 |  |  |

*Nota: si noti che all’interno del codice ad Ecc3 l’operazione di effettuare la sottrazione attraverso il complemento a 10 del secondo termine e quindi attraverso l’operazione di “complementare e sommare +1” dovrebbe prevedere l’operazione di sommare 0100 o nel caso di più cifre 0011\_0011\_0011\_0100 (che è la corretta rappresentazione del numero 1 nel codice ecc3 adottato). A tale operazione però dovrebbe seguire come ad ogni somma la “correzione” che prevede di sottrarre 0011 se non c’è stato riporto o sommare 0011 se il riporto c’è stato. Notasi però che per tutte le cifre comprese tra 1 e 9 il riporto non sarà MAI presente, e quindi sommare 0100 e poi sottrarre 0011 equivale a sommare 0001 direttamente in binario.   
Fa eccezione purtroppo il caso in cui la cifra in questione fosse 0. Nell’operazione di complemento a 10 però si può notare che tale cifra rimmarrà immutata quando posta nelle posizioni meno significative, mentre bisognerà sommare 0001 alla cifra meno significativa diversa da zero.*

Es: Trascrivere in codice ECC3 il numero 50300 e calcolarne il complemento a 10   
*(per poter eseguire un operazione di sottrazione)*

*Metodo 1:*

1. si esegue il complemento a 9 invertendo bit a bit tutte le cifre

2. si somma il valore 1 nella forma 00001 ovvero “0011\_0011\_0011\_0011\_0100” per ottenere il complemento a 10

3. si esegue la correzione sommando 0011 se c’è stato riporto oltre il 4 bit al passo precedente oppure sommando 1101 se non c’è stato riporto (ma ovviamente trascurando il riporto al di fuori dei 4 bit) – evidenziato nel rigo blu

Nota: per semplicità in rosso sono stati evidenziati solamente i riporti relativi al passo 2 di cui sopra solo quando questi arrumevano valore 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | \_ | 0 | 0 | 1 | 1 | \_ | 0 | 1 | 1 | 0 | \_ | 0 | 0 | 1 | 1 | \_ | 0 | 0 | 1 | 1 |  | : | 5 | 0 | 3 | 0 | 0 |
| 1 | 1 | 1 |  |  |  |  |  |  |  |  | 1 | 1 | **1** |  | 1 | 1 | 1 | **1** |  | 1 |  |  |  |  |  |  |  | 1 | 1 |  |
| 0 | 1 | 1 | 1 | \_ | 1 | 1 | 0 | 0 | \_ | 1 | 0 | 0 | 1 | \_ | 1 | 1 | 0 | 0 | \_ | 1 | 1 | 0 | 0 | + | : | 4 | 9 | 6 | 9 | 9 |
| 0 | 0 | 1 | 1 | \_ | 0 | 0 | 1 | 1 | \_ | 0 | 0 | 1 | 1 | \_ | 0 | 0 | 1 | 1 | \_ | 0 | 1 | 0 | 0 |  |  | 0 | 0 | 0 | 0 | 1 |
|  |  |  |  |  | 1 | 1 | 1 |  |  | 1 |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 0 | 1 | 0 |  | 1 | 1 | 1 | 1 |  | 1 | 1 | 0 | 1 |  | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | + |  |  |  |  |  |  |
| 1 | 1 | 0 | 1 |  | 1 | 1 | 0 | 1 |  | 1 | 1 | 0 | 1 |  | 0 | 0 | 1 | 1 |  | 0 | 0 | 1 | 1 |  |  |  |  |  |  |  |
| 0 | 1 | 1 | 1 |  | 1 | 1 | 0 | 0 |  | 1 | 0 | 1 | 0 |  | 0 | 0 | 1 | 1 |  | 0 | 0 | 1 | 1 |  |  | 4 | 9 | 7 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 4 |  |  |  |  | 9 |  |  |  |  | 7 |  |  |  |  | 0 |  |  |  |  | 0 |  |  |  |  |  |  |  |  |

*Metodo 2:*

1. si riconosce che in posizione meno significativa sono presenti degli zeri. Questi vengono riportati come sono

2. si complementano tutte le altre cifre

3. alla cifra meno significativa (diversa da zero) si somma 0001 (in forma binaria)

4. Non vi sarà alcun riporto

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | \_ | 0 | 0 | 1 | 1 | \_ | 0 | 1 | 1 | 0 | \_ | 0 | 0 | 1 | 1 | \_ | 0 | 0 | 1 | 1 |  | : | 5 | 0 | 3 | 0 | 0 |
| 0 | 1 | 1 | 1 | \_ | 1 | 1 | 0 | 0 | \_ | 1 | 0 | 0 | 1 |  |  |  |  |  |  |  |  |  |  |  | : | 4 | 9 | 6 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |  |
| 0 | 1 | 1 | 1 |  | 1 | 1 | 0 | 0 |  | 1 | 0 | 1 | 0 |  | 0 | 0 | 1 | 1 |  | 0 | 0 | 1 | 1 |  |  | 4 | 9 | 7 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 4 |  |  |  |  | 9 |  |  |  |  | 7 |  |  |  |  | 0 |  |  |  |  | 0 |  |  |  |  |  |  |  |  |

Somma e Sottrazione in codice di AIKEN:

Codificare i seguenti numeri in Cod. di AIKEN ed eseguire l’operazione

47 + 35

I numeri vengono codificati cifra per cifra ed ogni cifra è codificata a 4 bit

0100\_1101 + 0011\_1011

Si esegue la somma in BCD e se i risultati ottenuti NON sono tra i valori consentiti dal codice si esegue una “correzione” ovvero si somma 6 se non c’è stato riporto oppure si sottrae 6 se c’è stato riporto verso il digit successivo. Si noti che sottrarre 6 equivale a sommare 10 ed a trascurare un eventuale riporto finale.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | **1** |  | 1 | 1 | 1 |  |
| 0 | 1 | 0 | 0 | \_ | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | \_ | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | \_ | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | \_ | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | \_ | 0 | 0 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
|  |  | 8 |  |  |  |  | 2 |  |

Codificare i seguenti numeri in Cod. AIKEN ed eseguire l’operazione

47 – 35

I numeri vengono codificati cifra per cifra ed ogni cifra è codificata a 4 bit

0100\_1101 - 0011\_1011

L’operazione di sottrazione si fa ricorrendo al complemento a 10 ovvero complementando ogni singolo bit della seconda parola (ciò grazie al fatto che il codice di AIKEN è autocomplementante) ed aggiungendo 1 In pratica l’operazione svolta viene ad essere :

47 + (100 – 35) = 47 + 64 + 1

1. si scrivano le parole nel codice di aiken

2. si complementi la seconda parola bit a bit (complemento a 9)

3. si sommi 1 (complemento a 10)

4. si sommi alla prima parola (le operazioni 3 e 4 possono avvenire congiuntamente)

5. se vi è riporto oltre la cifra più signifcativa questo va trascurato (il risultato eccede 100 a causa dell’operazione di complemento a 10). Se non vi fosse riporto ciò indicherebbe che il risultato è negativo ed andrebbe complementato ulteriormente  
6. se le cifre non sono presenti nel codice si apportino le dovute correzioni (sommare 0110 oppure 1010)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
| 0 | 1 | 0 | 0 | \_ | 1 | 1 | 0 | 1 | - |
| 0 | 0 | 1 | 1 | \_ | 1 | 0 | 1 | 1 |  |
| 1 |  |  | 1 |  | 1 |  | 1 |  |  |
| 0 | 1 | 0 | 0 | \_ | 1 | 1 | 0 | 1 | + |
| 1 | 1 | 0 | 0 | \_ | 0 | 1 | 0 | 0 | + |
|  |  |  |  |  |  |  |  | 1 |  |
| 0 | 0 | 0 | 1 |  | 0 | 0 | 1 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | 1 |  |  |  |  | 2 |  |  |

Codificare i seguenti numeri in Cod. AIKEN ed eseguire l’operazione

35 – 47

I numeri vengono codificati cifra per cifra ed ogni cifra è codificata a 4 bit

0011\_1011 - 0100\_1101

L’operazione di sottrazione si fa ricorrendo al complemento a 10 ovvero complementando ogni singolo bit della seconda parola (ciò grazie al fatto che il codice di AIKEN è autocomplementante) ed aggiungendo 1 In pratica l’operazione svolta viene ad essere :

35 + (100 – 47) = 35 + 52 + 1 = 88 (= 100 – 12)

Il fatto che il risultato risulterà minore di 100 sta ad indicare che la cifra ottenuta è associata ad un numero negativo e pertanto andrà nuovamente complementata.

1. si scrivono le cifre nel codice di aiken

2. Si complementa bit a bit la seconda cifra ( omplemento a 9)

3. Si somma 1 (Complemento a 10)

4. si sommano le prime 2 cifre (l’operazione 3 e 4 possono avvenire congiuntamente)

5. Si nota che non vi è riporto al di fuori della parola (ovvero il risultato è minore di 100. Questo indica che il risultato in realtà rappresenta un numero negativo.  
6. per avere le cifre corrette occorre complementare nuovamente il risultato e sommare 1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0 | 0 | 1 | 1 | \_ | 1 | 0 | 1 | 1 | - |
|  |  | 0 | 1 | 0 | 0 | \_ | 1 | 1 | 0 | 1 |  |
| rip |  |  | 1 | 1 |  |  |  | 1 | 1 |  |  |
|  |  | 0 | 0 | 1 | 1 | \_ | 1 | 0 | 1 | 1 | + |
|  |  | 1 | 0 | 1 | 1 | \_ | 0 | 0 | 1 | 0 | + |
|  |  |  |  |  |  |  |  |  |  | 1 |  |
| rip |  |  |  |  |  |  | 1 | 1 |  |  |  |
|  |  | 1 | 1 | 1 | 0 | \_ | 1 | 1 | 1 | 0 |  |
|  |  | 0 | 0 | 0 | 1 | \_ | 0 | 0 | 0 | 1 |  |
|  |  |  |  |  |  |  |  |  |  | 1 |  |
|  |  | 0 | 0 | 0 | 1 | \_ | 0 | 0 | 1 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  | 2 |  |  |

Conversione da Binario a BCD:

Codifificare in Binario il numero decimale 735 e successivamente convertire il numero binario in BCD

* Si divida il numero originale per 2, si calcoli il resto e la parte intera della divisione.
* Si iteri successivamente il procedimento partendo dal risultato della divisione testè eseguita
* La successione dei resti fornirà il numero binario ordinato dal LSB (Less Significant Bit) al MSB (Most Significant Bit)



Primo passo: il numero 735 in binario è rappresentato dalla sequenza 1011011111

A riprova si può effettuare il procedimento inverso:



Partendo dal MSB , passo passo, raddoppia e si somma il valore del bit seguente

Secondo passo: Partendo dal numero espresso in binario si effettuano shift successivi e si raggruppano i bit in “nibble” composti da 4 bits. Se il valore è maggiore o uguale a 0101 si somma 0011 e si prosegue fintanto che l’intera parola non sia shiftata e tutti i bit siano entrati



Probabilità di errore NON rilevato:

Supponendo di impiegare il seguente codice composto da 8 parole (A,B, … H) di 7 bit per codificare 8 elementi distinti. Calcolare quale sia la probabilità di un errore non rilevato durante la trasmissione di ciascuna delle parole, supponendo che il “bit error rate” sia pari al 10 %,, e se vi si aggiungesse un ulteriore bit quale controllore di parità ?

Codice:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| B | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| C | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| D | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| E | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| F | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| G | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| H | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

Innanzitutto si calcolino le distanze reciproche tra tutte le parole

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H |
| A | 0 | 3 | 4 | 2 | 4 | 3 | 3 | 3 |
| B | 3 | 0 | 5 | 3 | 5 | 4 | 2 | 2 |
| C | 4 | 5 | 0 | 4 | 4 | 3 | 5 | 5 |
| D | 2 | 3 | 4 | 0 | 2 | 5 | 3 | 3 |
| E | 4 | 5 | 4 | 2 | 0 | 5 | 3 | 5 |
| F | 3 | 4 | 3 | 5 | 5 | 0 | 2 | 4 |
| G | 3 | 2 | 5 | 3 | 3 | 2 | 0 | 4 |
| H | 3 | 2 | 5 | 3 | 5 | 4 | 4 | 0 |

Si nota che la distanza di Hamming ed il numero di parole di codice poste a tale distanza, per ciascuna parola è la seguente sono le seguenti, e di conseguenza la probabilità di un errore Non rilevato è riportata nell’ultima colonna

|  |  |  |  |
| --- | --- | --- | --- |
|  | Dist.Hamm | N | Pr |
| A | 2 | 1 | 0,01 |
| B | 2 | 2 | 0,02 |
| C | 3 | 1 | 0,001 |
| D | 2 | 2 | 0,02 |
| E | 2 | 1 | 0,01 |
| F | 2 | 1 | 0,01 |
| G | 2 | 2 | 0,02 |
| H | 2 | 1 | 0,01 |

Si può ad esempio notare che la parola C avendo una distanza di Hamming pari a 3 rispetto tutte le altre parole risulti molto più robusta ed è più difficile che venga “confusa” in fase di trasmissione.

Se poi si scegliesse di inserire un ulteriore bit quale controllore di parità

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| A | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| B | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| C | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| D | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| E | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| F | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| G | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| H | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |

Si potrebbe notare che le uniche a beneficiarne sono le distanze dispari che vengono incrementate di una unità, mentre le distanze pari tra le parole rimangono invariate, ritrovando pertanto una tabella delle distanze come qui riportato

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H |
| A | 0 | 4 | 4 | 2 | 4 | 4 | 4 | 4 |
| B | 4 | 0 | 6 | 4 | 6 | 4 | 2 | 2 |
| C | 4 | 6 | 0 | 4 | 4 | 4 | 6 | 6 |
| D | 2 | 4 | 4 | 0 | 2 | 6 | 4 | 4 |
| E | 4 | 6 | 4 | 2 | 0 | 5 | 4 | 6 |
| F | 4 | 4 | 4 | 6 | 6 | 0 | 2 | 4 |
| G | 4 | 2 | 6 | 4 | 4 | 2 | 0 | 4 |
| H | 4 | 2 | 6 | 4 | 6 | 4 | 4 | 0 |

Il che porta a modificare le distanze di Hamming solo di alcune parole, insieme al numero di parole poste a tale distanza, aumentando pertanto la robustezza del codice, ma esclusivamente per la parola “C”

|  |  |  |  |
| --- | --- | --- | --- |
|  | Dist.Hann | N | Pr |
| A | 2 | 1 | 0,01 |
| B | 2 | 2 | 0,02 |
| C | 4 | 4 | 0,0004 |
| D | 2 | 2 | 0,02 |
| E | 2 | 1 | 0,01 |
| F | 2 | 1 | 0,01 |
| G | 2 | 2 | 0,02 |
| H | 2 | 1 | 0,01 |

Un codice completo a sei bit, a peso costante pari a 3 da quante parole è composto ? Qual è la probabilità di errore non rilevato se la trasmissione avviene in un canale la cui probabilità di errore per singolo bit è dell 1 % ?

Analizziamo innanzitutto come è fatta la parola VALIDA di codice: essa sarà composta evidentemente da 3 bit posti a uno e pertanto i rimanenti 3 bit ovviamente posti a 0.

Es:



Il codice completo sarà composto quindi da 20 parole distinte che corrispondono a quante siano le diverse combinazioni con cui posso scegliere i tre bit da essere posti a 1 sui 6 bit disponibili.

Per ciascuna parola la distanza di Hamming dalle altre parole è pari a 2.

Per passare da una parola di codice ad un’altra sempre appartenete al codice a distanza 2 bisogna invertire tanto un bit posto a 0 quanto un bit posto a 1.

Es.



In quanti modi diversi si può effettuare tale scelta ? Vi sono 3 modi per scegliere il bit pari a 1 da invertire e altrettanti per scegliere il bit a zero da invertire. Pertanto si desume che per qualsiasi parola di codice vi sono 9 parole di codice a distanza di Hamming. Pertanto ciascuna parola ha la medesima probabilità di errore non rilevabile che è

Un codice completo a sei bit, è realizzato con un peso costante pari a 3, ma col vincolo che il primo e l’ultimo BIT devono sempre assumere un valore diverso l’uno dall’altro. Da quante parole è composto tale codice? Qual è la probabilità di errore non rilevato se la trasmissione avviene in un canale la cui probabilità di errore per singolo bit è dell 1 % ?

Analizziamo innanzitutto come è fatta la parola VALIDA di codice: essa sarà composta evidentemente da 3 bit posti a uno e pertanto i rimanenti 3 bit posti a 0, ma il primo e l’ultimo devono essere sempre diversi tra loro

Es:



Vi sono pertanto 2 tipologie di parole: quelle il cui primo bit è 1 (e l’ultimo è 0)

E di queste ve ne sono tante quanti sono i modi per combinare i 4 bit centrali della parola:

E poi ve ne sono altrettante dove il primo bit è 0 (e l’ultimo è 1)

Pertanto il codice completo è composto da 12 parole distinte.

Un altro modo per arrivare al medesimo risultato è quello di considerare il codice di 6 bit a peso 3 considerato nell’esercizio precedente composto da 20 parole e scartare da queste tutte quelle che hanno in prima ed in ultima posizione due zeri oppure due uni e di queste ve ne sono 4+4 =8 quindi 20-8 =12.

Perché vi sia un errore non rilevabile una parola deve diventare un’altra di codice e questo deve avvenire attraverso 2 errori: uno 0 viene ricevuto come 1 ed un 1 viene ricevuto come 0, ma se l’errore colpisce il primo bit deve colpire anche l’ultimo per non essere rilevabile.

Quindi ogni parola ha a distanza 2 esattamente 5 parole di codice:

una è quella in cui vengono invertiti il primo e l’ultimo bit, mentre altre 4 sono quelle in cui uno dei due zeri centrali viene trasformato in 1 ed al contempo uno degli uni centrali viene trasformato in zero.

Pertanto la probabilità di errore NON rilevabile è uguale per ciascuna prola di codice e vale:

Codice di Hamming con H=4:

In un canale a 12 bit vengono trasmesse le seguenti parole utilizzando il codice di Hmming.   
Valutare se la parola ricevuta sia corretta oppure se via stato un errore di trasmissione. In tal caso, ove possibile suggerire la parola di codice che più verosimilmente sia stata trasmessa.

Parola 1: 0xFAA

Parola 2: 0x9A1

Parola 3: 0x167

Parola 4: 0xDB5

Parola 5: 0x747

Parola 1: 0xFAA

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | check |  |
| parola esagesimale |  |  | F |  |  |  | A |  |  |  | A |  |  |  |
| parola binaria | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | ok |  |
| Contr. 1 |  | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | ok | 0 |
| Contr. 2 |  |  | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | ok | 0 |
| Contr. 3 |  |  |  |  | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | ok | 0 |
| Contr. 4 |  |  |  |  |  |  |  |  | 1 | 0 | 1 | 0 | ok | 0 |

Si trascriva la parola in binario: 1111\_1010\_1010. I bit di controllo sono evidenziati in verde e i bit sui quali essi effettuano il controllo di parità sono evidenziati in giallo.

NON si nota alcun errore quindi con ottima probabilità la parola ricevuta è corretta. Si noti infatti che la parola di codice ad essa più prossima è posta a distanza almeno 4 ed ipotizzare un errore quadruplo è estremamente più improbabile della probabilità di non avere avuto alcun errore errore.

Esempio: Supponendo di avere una probabilità di errore su ogni singolo bit pari a 10% l’errore quadruplo si presenta con una probabilità 0.01% (10% ^ 4) mentre la probabilità di ricevere la parola corretta ha una probabilità del 28,2% (90% ^ 12).

Parola 2: 0x9A1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | check |  |
| parola esagesimale |  |  | 9 |  |  |  | A |  |  |  | 1 |  |  |  |
| parola binaria | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | no |  |
| Contr. 1 |  | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | ok | 0 |
| Contr. 2 |  |  | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | no | 1 |
| Contr. 3 |  |  |  |  | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | ok | 0 |
| Contr. 4 |  |  |  |  |  |  |  |  | 0 | 0 | 0 | 1 | no | 1 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | =10 |

Si trascriva la parola in binario: 1001\_1010\_0001. I bit di controllo sono evidenziati in verde e i bit sui quali essi effettuano il controllo di parità sono evidenziati in giallo. Dal check si nota che è stato rilevato un errore sia nel controllo globale sia nei gruppi di controllo 2 e 4.

La presenza dell’errore sull’intera parola indica che l’errore è di sicuramente di molteplicità dispari. Esso potrebbe essere pertanto un errore singolo oppure un errore triplo o magari anche superiore , ma questi ultimi risultano essere estremamente meno probabili, quindi si può cercare se vi sia una parola a distanza uno dalla parola ricevuta che plausibilmente è la parola originariamente trasmessa.

Il bit errato deve appartenere sia al gruppo di controllo 2 che al gruppo di controllo 4 e quindi può facilmente essere individuato nel bit numero 10 che è l’unico in comune solo tra i due gruppi di controllo. L’operazione può essere svolta in binario assegnando per ogni controllo errato il valore 1 e per ogni controllo corretto il valore 0. La parola finale 1010 letta in binario restituisce il valore 10 che identifica il bit da modificare per ottenere la parola di codice più vicina (a distanza unitaria) e che presumibilmente è quella inviata.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | check |
| parola esagesimale |  |  | 9 |  |  |  | A |  |  |  | 2 |  |  |
| parola binaria | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | ok |
| Contr. 1 |  | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | ok |
| Contr. 2 |  |  | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | ok |
| Contr. 3 |  |  |  |  | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | ok |
| Contr. 4 |  |  |  |  |  |  |  |  | 0 | 0 | 1 | 1 | ok |

Una rapida riprova indica che invertendo il bit 10 si ottiene una parola (0x9A2) appartenente al codiceed a distanza unitaria dalla parola ricevuta.

Parola 3: 0x167

Si trascriva la parola in binario: 0001\_0110\_0111. I bit di controllo sono evidenziati in verde e i bit sui quali essi effettuano il controllo di parità sono evidenziati in giallo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | check |  |
| parola esagesimale |  |  | 1 |  |  |  | 6 |  |  |  | 7 |  |  |  |
| parola binaria | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | no |  |
| Contr. 1 |  | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | ok | 0 |
| Contr. 2 |  |  | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | ok | 0 |
| Contr. 3 |  |  |  |  | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | ok | 0 |
| Contr. 4 |  |  |  |  |  |  |  |  | 1 | 1 | 1 | 1 | ok | 0 |

Dal check si nota che vi deve essere stato è un errore dispari sull’intera parola infatti il numero dei bit posti a 1 risulta essere dispari, ma nei vari gruppi di controllo non si riscontra alcun errore. Per trovare la parola a distanza unitaria dalla quale con ottime probabilità è derivata la parola ricevuta, bisogna individuare quel bit che pur appartenendo alla parola non appartenga a nessuno dei gruppi di controllo. Questo bit è il bit di parità globale stesso (ovvero il bit in posizione 0000=0), quindi la parola che con ottime probabilità è stata inviata è 0x967.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | check |  |
| parola esagesimale |  |  | 9 |  |  |  | 6 |  |  |  | 7 |  |  |  |
| parola binaria | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | ok |  |
| Contr. 1 |  | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | ok | 0 |
| Contr. 2 |  |  | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | ok | 0 |
| Contr. 3 |  |  |  |  | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | ok | 0 |
| Contr. 4 |  |  |  |  |  |  |  |  | 1 | 1 | 1 | 1 | ok | 0 |

Parola 4 :0xDB5

Si trascriva la parola in binario: 1101\_1011\_0101. I bit di controllo sono evidenziati in verde e i bit sui quali essi effettuano il controllo di parità sono evidenziati in giallo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | check |
| parola esagesimale |  |  | D |  |  |  | B |  |  |  | 5 |  |  |
| parola binaria | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 1 |  | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | no |
| Contr. 2 |  |  | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 3 |  |  |  |  | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | no |
| Contr. 4 |  |  |  |  |  |  |  |  | 0 | 1 | 0 | 1 | ok |

Si può constatare che attraverso il controllo di parità globale non si evidenzia alcun errore. Da ciò si desume che: o non c’è stato alcun errore oppure questo è stato di molteplicità pari. Se non vi fosse errore i controlli per gruppi non dovrebbero restituire alcun errore, ma così non è, pertanto la parola ha subito un errore probabilmente doppio e quindi non si può risalire alla parola dalla quale essa potrebbe derivare, in quanto vi sono molteplici parole a distanza 2 che potrebbero essere considerate possibili candidate, ad esempio: 0x5F5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |  |
| parola esagesimale |  |  | 5 |  |  |  | F |  |  |  | 5 |  |  |
| parola binaria | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 1 |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 2 |  |  | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 3 |  |  |  |  | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 4 |  |  |  |  |  |  |  |  | 0 | 1 | 0 | 1 | ok |

Ma anche 0x935

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |  |
| parola esagesimale |  |  | 9 |  |  |  | 3 |  |  |  | 5 |  |  |
| parola binaria | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 1 |  | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 2 |  |  | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 3 |  |  |  |  | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 4 |  |  |  |  |  |  |  |  | 0 | 1 | 0 | 1 | ok |

Oppure 0xFA5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |  |
| parola esagesimale |  |  | F |  |  |  | A |  |  |  | 5 |  |  |
| parola binaria | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | ok |
| Contr. 1 |  | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | ok |
| Contr. 2 |  |  | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | ok |
| Contr. 3 |  |  |  |  | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | ok |
| Contr. 4 |  |  |  |  |  |  |  |  | 0 | 1 | 0 | 1 | ok |

Oppure 0xD95

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |  |
| parola esagesimale |  |  | D |  |  |  | 9 |  |  |  | 5 |  |  |
| parola binaria | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 1 |  | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 2 |  |  | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 3 |  |  |  |  | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | ok |
| Contr. 4 |  |  |  |  |  |  |  |  | 0 | 1 | 0 | 1 | ok |

Se il codice fosse a 16 bit (codice ottimo) allora per ogni bit della parola ricevuta si potrebbe trovare un corrispondente bit che, se modificati entrambi, identificherebbero una parola di codice a distanza 2 dalla parola data.

Parola 5 :0x747

Si trascriva la parola in binario: 1101\_1011\_0101. I bit di controllo sono evidenziati in verde e i bit sui quali essi effettuano il controllo di parità sono evidenziati in giallo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | check |  |
| parola esagesimale |  |  | 7 |  |  |  | 4 |  |  |  | 7 |  |  |  |
| parola binaria | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | no |  |
| Contr. 1 |  | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | ok | 0 |
| Contr. 2 |  |  | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | ok | 0 |
| Contr. 3 |  |  |  |  | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | no | 1 |
| Contr. 4 |  |  |  |  |  |  |  |  | 0 | 1 | 1 | 1 | no | 1 |

Dall’analisi del controllo di parità per gruppi si desume che :

Il controllo globale indica la presenza di un errore di molteplicità dispari e sappiamo che tra questi il più probabile è l’errore unitario.

I controlli di parità a gruppi individuano che l’errore dovrebbe essere di un bit presente sia nel gruppo di controllo 4 che nel gruppo di controllo 3. Ma purtroppo questi due gruppi non hanno alcun bit in comune. Ovvero applicando la regola della codifica binaria degli errori il bit sbagliato dovrebbe trovarsi in posizione 1100 =12, ma i bit della parola sono numerati da 0 a 11 ed il bit 12 NON ESISTE. Da ciò si può desumere che questo è un raro caso in cui giocoforza l’errore seppur raro è stato triplo ed ovviamente non vi è modo di risalire alla parola originale.

Si noti che nel caso di un codice ottimo (ad esempio a 16 bit questo caso non potrebbe mai verificarsi) in quanto tutti i gruppi di controllo hanno almeno un bit in condivisione, quindi in tal caso un eventuale errore triplo non potrebbe in alcun modo essere distinguibile da un errore singolo.

Va inoltre sottolineato che il riuscire ad identificare con certezza un errore triplo è un caso assolutamente fortuito che capita solo in particolari condizioni:

Consideriamo ad esempio di aver trasmesso la parola corretta: 0xFAA

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | check |
| parola esagesimale |  |  | F |  |  |  | A |  |  |  | A |  |  |
| parola binaria | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | ok |
| Contr. 1 |  | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | ok |
| Contr. 2 |  |  | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | ok |
| Contr. 3 |  |  |  |  | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | ok |
| Contr. 4 |  |  |  |  |  |  |  |  | 1 | 0 | 1 | 0 | ok |

E supponiamo che essa abbia subito un errore triplo trasformandola nella parola 0xB6A

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | check |
| parola esagesimale |  |  | B |  |  |  | 6 |  |  |  | A |  |  |
| parola binaria | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| contr. Globale | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | no |
| Contr. 1 |  | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | ok |
| Contr. 2 |  |  | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | ok |
| Contr. 3 |  |  |  |  | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | ok |
| Contr. 4 |  |  |  |  |  |  |  |  | 1 | 0 | 1 | 0 | ok |

Attraverso il check dei gruppi di controllo si sarebbe portati a riconoscervi un errore singolo che abbia colpito il bit di parità globale suggerendo così di correggerla nella parola 0x36A che effettivamente è la parola di codice più vicina alla parola ricevuta. Ovvero in questo caso, a differenza della situazione assolutamente fortuita avvenuta nel caso precedente NON vi è modo di distinguere un errore triplo avvenuto sulla parola 0xFAA da un errore singolo avvenuto sulla parola 0x36A, però parlando in termini probabilistici si può affermare con sicurezza che l’errore triplo sia assolutamente molto più improbabile dell’errore singolo.