Lezione 6 Codici ridondanti e non 3

venerdì 6 ottobre 2023 12:58

Vediamo la correzione dell'esercizio per casa : dati due numeri a 16 bit fare il complemento a 2, fare la somma ed trasformare la somma in ottale :

A=2176	B=-2040

Cominciamo con il trasformare i numeri in binario in valore (il secondo):

2176	0
1088	0
544	0
272	0
136	0
66	1
33	0
16	1
8	0
4	0
2	0
0	1

Dato che la rappresentazione della parola è a 12 bit, ne dobbiamo aggiungere 4 :

0000	1000	1010	0000

Analogamente per b, ma considerando il modulo :

2040	0
1020	0
510	0
255	0
127	0
63	1
31	1
15	1
7	1
3	1
1	1
0	1

Dato che abbiamo calcolato il valore in modulo, dobbiamo trasformarlo nel complemento a 2 : prima complemento tutti i bit a partire dal primo "1" e poi aggiungiamo "1" :

0000	0111	1111	1000

Complemento a 2 :

1111	1000	0000	0111
			+1
1111	1000	0000	1000

Quindi è la rappresentazione in complemento a 2 del numero -2040 !

Facciamo ora la somma : **se signed non c'è overflow** , in quanto i numeri sono opposti , altrimenti :

0000	0111	1111	1000	
				+
1111	1000	0000	1000	=
10000	0000	0000	0000	

Infine vediamo la parola in ottale (raggruppo i bit in gruppi da 3):

1000000000000 -> 001 000 000 000 000 -> 1 0 0 0 0

Esempio :

1. Codificare 0110000
2. N=7 per trovare k si va a tentativi:
   a. K=3 non rispetta disequazione : 8<=7 FALSO
   b. K=4 rispetta disequazione : 8<=16-4 Vero -> prendiamo questo valore di k
   c. M=n+k -> 7+4=11
3. Si scrive il messaggio nel seguente modo includendo i bit di parità :
   a. ****0*110*000 [^], dove gli "*" sono i bit di parità nelle posizioni delle potenze di 2,** assumendo la cifra meno significativa a sinistra
4. Vediamo la rappresentazione in binario delle configurazioni:

1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011

Dopo mi fermo perché uso 11 bit massimo!

5. Dato che abbiamo 4 bit di parità dobbiamo trovare la parità di tutte le k parole : per ogni bit dobbiamo vedere il bit n-esimo settato ad "1" se presente :
   a. P1 = 1,3,5,7,9,11
      i. Di questa configurazione prendiamo i valori nella [^] corrispondenti , ignorando il primo numero in cui compare :
      ii. Nel nostro caso : 0+1+0+0+0
   b. P2= 2,3,6,7,10,11

i. Nel nostro caso :0+1+0+0+0

c. P4 = 4,5,6,7
   i. Nel nostro caso : 1+1+0
d. P8 = 8,9,10,11 non vado oltre perché ho solo 11 bit
   i. Nel nostro caso : 0+0+0
e. Di ognuna di queste configurazioni facciamo or tra gli addendi : se totale è pari il bit di parità vale 0 , altrimenti 1
   i. In questo esempio ho come sequenza 0110
f. Assemblo il messaggio finale inserendo al posto di "*" i bit corrispondenti :
   i. 11001100000

6. Poniamo caso che il messaggio ricevuto sia : 11001100100 (inclusi i bit di parità) , come facciamo a vedere la posizione dove c'è stato il bit flip ? Ripetiamo il procedimento , ma con la nuova parola :

Esempio :

1. decodificare 11001100100
2. N=8 per trovare k si va a tentativi:
   a. K=3 non rispetta disequazione : 9<=7 FALSO
   b. K=4 rispetta disequazione : 9<=16-4 Vero -> prendiamo questo valore di k
   c. M=n+k -> 7+4=11
3. Vediamo la rappresentazione in binario delle configurazioni:

1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100

Dopo mi fermo perché uso 12 bit massimo!

4. Dato che abbiamo 4 bit di parità dobbiamo trovare la parità di tutte le k parole : per ogni bit dobbiamo vedere il bit n-esimo settato ad "1" se presente :
   a. P1 = 1,3,5,7,9,11
      i. Di questa configurazione prendiamo i valori nella [^] corrispondenti , ignorando il primo numero in cui compare :
      ii. Nel nostro caso : 0+0+0+1
   b. P2= 2,3,6,7,10,11
      i. Nel nostro caso : 0+1+0+0+0
   c. P4 = 4,5,6,7
      i. Nel nostro caso : 1+1+0
   d. P8 = 8,9,10,11 non vado oltre perché ho solo 11 bit
      i. Nel nostro caso : 1+0+0
   e. Di ognuna di queste configurazioni facciamo or tra gli addendi : se totale è pari il bit di parità vale 0 , altrimenti 1
   f. **Aggiungo ulteriore parità** :
      i. In questo esempio ho come sequenza 1001
   g. Assemblo il messaggio finale inserendo al posto di "*" i bit corrispondenti :
      i. 11001100000
   h. Lo confronto con il messaggio di partenza :

    i. 11001100100
i. Converto in decimale il valore della seconda parità :
    i. (1001)base 2 -> (9) base 10, quindi il bit flippato è in questa posizione.