Hamming Encoding/Decoding Circuit

by Antonio Bernardini

Table of contents

1 Homework

In questo esercizio viene chiesto di progettare e realizzare un circuito per la codifica e decodifica di una parola a 8 bit utilizzando il **codice di Hamming** con distanza minima di Hamming $h = 3$. Il circuito dovrà essere in grado di gestire sia la fase di codifica che quella di decodifica, a seconda dell'operazione richiesta.

Il circuito ha 13 ingressi e 13 uscite, che vengono utilizzate in maniera differente a seconda che esso operi in modalità di codifica o di decodifica. Gli ingressi sono chiamati $d, p_1, p_2, d_1, p_4, d_2, d_3, d_4, p_8, d_5, d_6, d_7, d_8$. Le uscite sono chiamate $e, y_1, \ldots, y_{12}$. L'ingresso d (decode) indica se si opera in modalità di codifica ($d = 0$) o di decodifica ($d = 1$). Il funzionamento delle due modalità è spiegato qui sotto.

1.1 Modalità di codifica

Quando $d = 0$, il circuito opera in modalità di codifica. In questo caso, i bit di input p_1, p_2, p_4, p_8 sono *don't care conditions* e possono assumere qualsiasi valore. I bit di dati $d_1, \ldots, d_8$ portano il valore che si intende codificare.

Il circuito calcola quindi i valori dei bit di parità secondo un codice di Hamming $(12, 8)$ e riporta in output i seguenti valori:

Bit di uscita	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	e
Valore	primo bit di parità	secondo bit di parità	d_1	terzo bit di parità	d_2	d_3	d_4	quarto bit di parità	d_5	d_6	d_7	d_8	0

Table 1. Tabella dei valori per la codifica

Ossia, vengono riportati in output i bit dati in ingresso nelle posizioni corrette per la codifica di Hamming adottata e vengono generati i bit di parità (anch'essi posizionati nelle posizioni corrette).

Si noti che, in modalità di codifica, l'uscita e deve essere sempre forzata a zero.

1.2 Modalità di decodifica

Quando $d = 1$, il circuito riceve in input una parola codificate secondo il codice di Hamming $(12, 8)$ e fornisce in output il valore decodificato (ed eventualmente corretto). L'output sarà il seguente:

Bit di uscita	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	e
Valore	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	0	0	0	0	?

Table 2. Tabella dei valori per la decodifica

Dove con ? si intende il codice di errore: 0 se non è stato rilevato alcun errore di trasmissione, 1 se è stato rilevato almeno un errore.

Nel caso sia stato rilevato un errore, il circuito tenta la correzione: viene utilizzata la sindrome per determinare quale bit è necessario invertire (e si effettua l'inversione). Si noti che il risultato è sensato solo se si è verificato un solo errore.

Si noti che, in modalità di decodifica, i bit di output $y_9, \ldots, y_{12}$ sono sempre forzati a zero.

1.3 Vincoli

- Utilizzare esclusivamente **porte logiche** di base e **componenti elementari** (encoder, decoder, ...) per la progettazione del circuito.

- Non è consentito utilizzare componenti predefiniti per la correzione degli errori o circuiti già progettati per il codice di Hamming.

2 Soluzione

2.1 Introduzione

Data la presenza di vincoli non possiamo utilizzare un'unica ROM (Read Only Memory) per rappresentare la logica combinatoria. Pertanto doppiamo ragionare in modo molto basilare. Infatti per capire come implementare un circuito per la codifica/decodifica di un codice di Hamming ad 8 bit occorre avere bene in mente quali sono le operazioni logiche che si svolgono durante la codifica, la decodifica e anche durante la correzione degli errori. L'idea è procedere step-by-step ricordando come di effettua la codifica/decodica e correzione degli errori "a mano" per poi automatizzare tale abilità con un circuito elettronico digitale.

2.2 Codifica

Immaginiamo di dover codificare il messaggio 10011010, utilizzando il codice di Hamming. Procediamo come segue. Per prima cosa sappiamo che il messaggio da codificare presenta $n = 8$ bit e che il codice di Hamming è un metodo per la costruzione di codici a *distanza di Hamming* $h \geqslant 3$. Pertanto indichiamo con $m = n + k$ cifre il numero di bit del messaggio codificato e calcoliamo il valore di k, utilizzando la seguente relazione:

$$n \leqslant 2^k - k - 1$$

da cui segue che $9 \leqslant 2^k - k$ per cui $k = 4$. Quindi otteniamo che il messaggio codificato avrà una lunghezza pari a $m = n + k = 12$. Cominciamo il nostro algoritmo di codifica sapendo che dovremmo aggiungere $k = 4$ cifre, ossia 4 bit di parità, che indicheremo rispettivamente con p_1, p_2, p_4 e p_8, dove il pedice i-esimo indica la posizione del bit di parità nel messaggio codificato. Quindi costruiamo la seguente tabella:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		p_1	p_2		p_4				p_8					
Copertura bit di parità	p_1													
	p_2													
	p_4													
	p_8													

Table 3. Codifica di un messaggio usando il codice di Hamming (parte 1)

Attualmente la tabella è vuota e la prima cosa che dobbiamo fare è riempirla inserendo il messaggio da codificare, ossia 10011010 come segue:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		p_1	p_2	1	p_4	0	0	1	p_8	1	0	1	0	
Copertura bit di parità	p_1													
	p_2													
	p_4													
	p_8													

Table 4. Codifica di un messaggio usando il codice di Hamming (parte 2)

a questo punto bisogna trasformare le posizioni (da 1 a 12) in binario e inserire il simbolo "✓" al posto degli 1 disponendo ciascun numero binario nella relativa colonna in base alla posizione. Per esempio il numero $(7)_{10} = (0111)_2$ quindi verrà scritto nella tabella come segue:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		p_1	p_2	1	p_4	0	0	1	p_8	1	0	1	0	
Copertura bit di parità	p_1							✓						
	p_2							✓						
	p_4							✓						
	p_8													

Table 5. Codifica di un messaggio usando il codice di Hamming (parte 3)

e ragionando allo stesso modo per le altre posizioni si ottiene:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		p_1	p_2	1	p_4	0	0	1	p_8	1	0	1	0	
Copertura bit di parità	p_1	✓		✓		✓		✓		✓		✓		
	p_2		✓	✓			✓	✓			✓	✓		
	p_4				✓	✓	✓	✓					✓	
	p_8								✓	✓	✓	✓	✓	

Table 6. Codifica di un messaggio usando il codice di Hamming (parte 4)

Ora bisogna individuare nel dato codificato le posizioni degli 1, ed in particolare in questo caso notiamo che in posizione $3, 7, 9$ e 11 ci sono tutti 1. Pertanto ci posizioniamo sopra ad una specifica posizione, per esempio la posizione 3 e coloriamo, con un colore a scelta (se si procede su carta conviene fare dei cerchi con la matita), tutti i simboli "✓" della colonna come segue:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		p_1	p_2	1	p_4	0	0	1	p_8	1	0	1	0	
Copertura bit di parità	p_1	✓		✓		✓		✓		✓		✓		
	p_2		✓	✓			✓	✓			✓	✓		
	p_4				✓	✓	✓	✓					✓	
	p_8								✓	✓	✓	✓	✓	

Table 7. Codifica di un messaggio usando il codice di Hamming (parte 5)

e ragionando allo stesso modo per le altre posizioni si ottiene:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		p_1	p_2	1	p_4	0	0	1	p_8	1	0	1	0	
Copertura bit di parità	p_1	✓		✓		✓		✓		✓		✓		
	p_2		✓	✓			✓	✓			✓	✓		
	p_4				✓	✓	✓	✓					✓	
	p_8								✓	✓	✓	✓	✓	

Table 8. Codifica di un messaggio usando il codice di Hamming (parte 6)

Infine calcoliamo i bit di parità p_1, p_2, p_4 e p_8 contando, da sinistra a destra, quanti simboli "✓" sono stati colorati (o cerchiati a matita). Per esempio per il calcolo del bit di parità p_1 abbiamo evidenziato le posizioni $3, 7, 9$ e 11 per un totale di 4 simboli "✓", pertanto ricordando la logica dei codici di parità pari (il bit di parità vale 1 se il numero di 1 nella codifica è dispari) e di parità dispari (il bit di parità vale 1 se il numero di 1 nella codifica è pari) ed utilizzando, per esempio, la logica del codice di parità pari abbiamo che $p_1 = 0$ perchè abbiamo contato un totale di 4 simboli "✓". Invece per esempio per il calcolo del bit di parità p_4 abbiamo evidenziato esclusivamente la posizione 7 per un totale di 1 simbolo "✓", portandoci alla conclusione che $p_4 = 1$. Ragionando allo stesso modo per gli altri bit di parità otteniamo:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		p_1	p_2	1	p_4	0	0	1	p_8	1	0	1	0	
Copertura bit di parità	p_1	✓		✓		✓		✓		✓		✓		0
	p_2		✓	✓			✓	✓			✓	✓		1
	p_4				✓	✓	✓	✓					✓	1
	p_8								✓	✓	✓	✓	✓	0

Table 9. Codifica di un messaggio usando il codice di Hamming (parte 7)

In conclusione il dato codificato che viene trasmesso è $\underline{0}11\underline{1}00\underline{1}0\underline{1}010$.

2.3 Decodifica con correzione degli errori

Se il ricevente riceve il messaggio codificato 011100101110 per accorgersi di un eventuale errore di trasmissione deve decodificare il messaggio e controllare la *syndrome* o *codice di controllo* $N_c = \langle p_8 p_4 p_2 p_1 \rangle$, dove i bit di parità p_i sono relativi alla procedura di decodifica e non sono quelli presenti nelle i-esime posizioni del messaggio ricevuto.

Procediamo dunque con l'algoritmo di decodifica inserendo il messaggio codificato ricevuto nella seguente tabella:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		0	1	1	1	0	0	1	0	1	1	1	0	
Copertura bit di parità	p_1													
	p_2													
	p_4													
	p_8													

Table 10. Decodifica di un messaggio usando il codice di Hamming (parte 1)

la procedura nel posizionamento dei simboli "✓" è identica alla codifica pertanto otteniamo:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		0	1	1	1	0	0	1	0	1	1	1	0	
Copertura bit di parità	p_1	✓		✓		✓		✓		✓		✓		
	p_2		✓	✓			✓	✓			✓	✓		
	p_4				✓	✓	✓	✓					✓	
	p_8								✓	✓	✓	✓	✓	

Table 11. Decodifica di un messaggio usando il codice di Hamming (parte 2)

così come è identica la colorazione dei simboli "✓" in base alle posizioni e agli 1 e di conseguenza è identico anche il calcolo dei bit di parità:

Posizione cifra		1	2	3	4	5	6	7	8	9	10	11	12	
Dato codificato		0	1	1	1	0	0	1	0	1	1	1	0	
Copertura bit di parità	p_1	✓		✓		✓		✓		✓		✓		0
	p_2		✓	✓			✓	✓			✓	✓		1
	p_4				✓	✓	✓	✓					✓	0
	p_8								✓	✓	✓	✓	✓	1

Table 12. Decodifica di un messaggio usando il codice di Hamming (parte 3)

quindi come si intuisce la *syndrone* $N_c = \langle p_8 p_4 p_2 p_1 \rangle = (1010)_2 = (10)_{10}$ quindi c'è stato un errore di trasmissione ed in particolare sapendo a priori che era stato modificato solo un bit, tra il trasmittente e il ricevente, possiamo dire con certezza che il bit in posizione 10 deve essere invertito da 1 a 0, infatti se così fosse otterremmo una *syndrome* $N_c = 0$ e quindi nessun errore di trasmissione.

2.4 Circuito finale

Il circuito finale è il seguente:

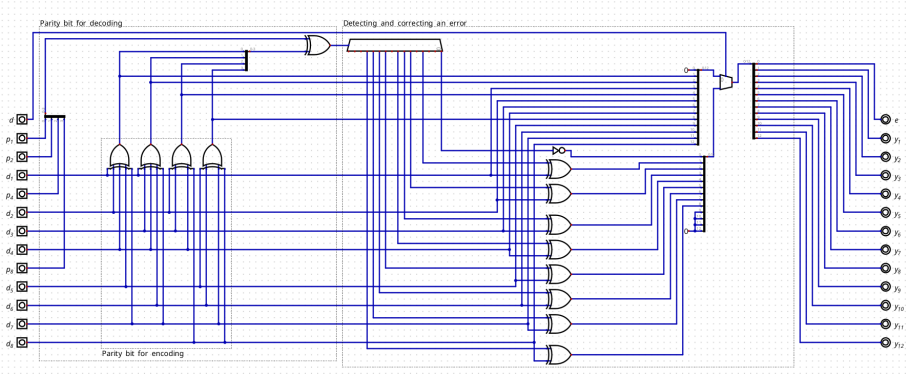

Figure 1. Circuito finale

consultabile dal file `circuit.dig` tramite il simulatore Digital.

2.4.1 Codifica

I bit di parità p_1, p_2, p_4, p_8 sono determinati in modo che ogni gruppo di bit controllato abbia una parità pari. Ogni bit di parità è calcolato come segue:

$$\begin{aligned} p_1 &= d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_7 \\ p_2 &= d_1 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \\ p_4 &= d_2 \oplus d_3 \oplus d_4 \oplus d_8 \\ p_8 &= d_5 \oplus d_6 \oplus d_7 \oplus d_8 \end{aligned}$$

dove $\oplus$ rappresenta l'operazione XOR.

2.4.2 Decodifica

Durante la decodifica, il ricevitore controlla la validità del messaggio verificando i bit di parità, generando un valore di *syndrome* per identificare eventuali errori.

La sindrome è un numero binario a 4 bit, calcolato come segue:

$$\begin{aligned} s_1 &= p_1 \oplus d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_7 \\ s_2 &= p_2 \oplus d_1 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \\ s_4 &= p_4 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_8 \\ s_8 &= p_8 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_8 \end{aligned}$$

Il risultato della *syndrome* $N_c = \langle s_8 s_4 s_2 s_1 \rangle$ identifica la posizione del bit errato (da 1 a 12). Ad esempio:

- Se $N_c = 0$, il messaggio è corretto.
- Se il valore è diverso da 0, esso indica la posizione del bit errato.

2.4.3 Correzione degli errori

Un decoder riceve il valore della sindrome e identifica la posizione del bit errato. Le porte XOR sono utilizzate per correggere il bit: se un bit è errato, viene invertito (da 0 a 1 o da 1 a 0).

2.4.4 Diversificazione tra codifica e decodifica

Il multiplexer differenzia tra le operazioni di codifica e decodifica:

- Codifica ($d = 0$): Il circuito calcola i bit di parità e li aggiunge ai dati originali.
- Decodifica ($d = 1$): Il circuito rileva gli errori, calcola la sindrome e corregge il bit errato, se necessario.

2.5 Conclusioni

Questo circuito fornisce una soluzione completa per la codifica, il rilevamento e la correzione degli errori di un singolo bit utilizzando il codice di Hamming. È particolarmente utile in sistemi di trasmissione dati, dove la correzione di errori è fondamentale per garantire l'affidabilità delle comunicazioni.