MASTER DEGREE in EMBEDDED COMPUTING SYSTEMS
A.Y. 2016 - 2017

Relazione Finale del Progetto: Error Correcting Code

Digital Systems

Professore
Luca Fanucci

Silvio Bacci

Indice

1 Specifiche di progetto

Progettare un circuito in grado di codificare una parola di 11 bit utilizzando la codifica di Hamming. La parola di uscita deve essere codificata in modo che un decodificatore sia in grado di rilevare e correggere un singolo bit errato sulla parola ricevuta e possa rilevare (ma non correggere) un doppio errore.
La codifica di Hamming implica l'aggiunta di alcuni bit di parità **Px** accanto a quelli della parola di ingresso **Dx**. Ogni bit di parità deve essere calcolato attraverso l'operazione di XOR tra alcuni bit della parola di ingresso. La tabella seguente indica, per ogni bit di parità, quali sono i bit coinvolti.

Posizione bit		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Codifica bit		P_1	P_2	D_1	P_4	D_2	D_3	D_4	P_8	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	P_0
Copertura bit	P_1			X		X		X		X		X		X		X	
	P_2			X			X	X			X	X			X	X	
	P_4					X	X	X					X	X	X	X	
	P_8									X	X	X	X	X	X	X	
	P_0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

La seconda riga della tabella riporta come deve essere costruita la parola in uscita dal codificatore (posizione 1 indica il bit meno significativo) Notare che il bit di parità **PO**, deve essere calcolato sfruttando anche gli altri bit di parità **P1**, **P2**, **P4**, **P8**.

La relazione finale del progetto deve contenere:

- Introduzione (descrizione algoritmo, possibili applicazioni, possibili architetture, etc.)
- Descrizione dell'architettura (diagramma a blocchi, ingressi/uscite, etc.)
- Codice VHDL (con commenti dettagliati)
- Testbench per la verifica
- Conclusioni

2 Algoritmo

2.1 Descrizione generale

La codifica di Hamming utilizza k bit di parità in aggiunta a n bit della parola originale, formando così una nuova parola da $n+k$ bit. In generale i bit sono numerati a partire da 1 fino a $n+k$ e tutte le posizioni che hanno come numero una potenza di 2 sono riservate per i bit di parità.
Dati k bit di parità (dove k è un intero ≥ 2) esiste un limite al numero di bit della codifica. Infatti la codifica potrà avere al massimo 2^k-1 bits. Quindi sapendo che la parola codificata sarà su $n+k$ bits, possiamo scrivere che $n+k \leq 2^k-1$, da cui otteniamo un limite al numero di bit della parola in ingresso. Per esempio, quando $k=3$, il numero totale di bits per la parola in input sarà $n+3 \leq 2^3-1$, che significa $n \leq 4$. Nel caso in cui k sia uguale a 4, avremo $n+4 \leq 2^4-1$ che significa $n \leq 11$. Dati i 2 esempi, è inoltre evidente che la codifica con $k=4$ sarà utilizzata quando la parola avrà un numero di bit superiore a 4. Infatti se le parola avesse soltanto 4 bits o meno, utilizzeremo 3 bit di parità anziché 4.

In generale ogni bit di parità viene ottenuto attraverso lo XOR di alcuni dei bit della parola in input. Usando lo XOR ciò che otteniamo alla fine non è altro che un bit di parità. In generale è indifferente il tipo di parità utilizzata (ovviamente deve essere garantita coerenza in ogni istante), anche se quella pari è più semplice da implementare perché è ottenuta direttamente attraverso lo XOR. In particolar modo i bit di parità sono distribuiti sui bit dei dati come segue:

- Il bit di parità in posizione 1 copre tutte quelle posizioni la cui codifica binaria (della posizione stessa) ha il bit meno significativo a 1.
- Il bit di parità in posizione 2 copre tutte quelle posizioni la cui codifica binaria (della posizione stessa) ha il secondo bit meno significativo a 1.
- Il bit di parità in posizione 3 copre tutte quelle posizioni la cui codifica binaria (della posizione stessa) ha il terzo bit meno significativo a 1.
- …

Alla codifica di Hamming di base può essere sempre aggiunto un ulteriore bit ottenuto attraverso lo XOR di tutti gli altri bit (compresi quelli di parità sopra menzionati). Aggiungendo questo bit di parità, la codifica di Hamming potrà rilevare e correggere un singolo errore e potrà rilevare anche doppi errori.

2.2 Codifica con k = 4 e bit di parità addizionale

Consideriamo adesso il caso specifico in cui abbiamo 11 bit di parola in input, 4 bit di parità e un ulteriore bit di parità, chiamato P_0 . In questo caso sappiamo già dalla tabella riportata nelle specifiche che:

$$P_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11}$$

$$P_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11}$$

$$P_4 = D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11}$$

$$P_8 = D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11}$$

$$P_0 = D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_1 \oplus P_2 \oplus P_4 \oplus P_8$$

Quando un eventuale ricevitore vuole rilevare ed eventualmente correggere degli errori ciò che dovrà fare, sarà semplicemente ricavare i seguenti bit di controllo:

$$C_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11}$$

$$C_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11}$$

$$C_4 = P_4 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11}$$

$$C_8 = P_8 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11}$$

$$P = D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_1 \oplus P_2 \oplus P_4 \oplus P_8 \oplus P_0$$

$$C = C_8 C_4 C_2 C_1$$

Chiaramente il bit di controllo ottenuto dipende anche dal tipo di parità utilizzato. Infatti se non ci fossero errori ogni bit di controllo dovrebbe essere a 1, in caso di parità dispari, e a 0, in caso di parità pari. Sapendo che noi abbiamo utilizzato una parità pari sappiamo che questi bit saranno a 0 in caso di nessun errore (o ancora meglio, in caso di un numero di bit errati pari).

In generale possiamo trovarci di fronte ai seguenti 4 casi:

1. $C = 0$ e $P = 0$
   1.1. Nessun errore
2. $C \neq 0$ e $P = 1$
   2.1. Singolo errore rilevato
3. $C \neq 0$ e $P = 0$
   3.1. Doppio errore rilevato
4. $C = 0$ e $P = 1$
   4.1. Singolo errore rilevato nel bit P_0

Dalle specifiche non viene richiesto di correggere doppi errori, ma solo di rilevarli, quindi il caso 3 ci permette di riconoscere questo tipo di errore. Al contrario nei casi 2 e 4, avendo un singolo errore, dobbiamo essere in grado di correggerlo. Nel caso 4 la correzione è semplice perché consociamo già il bit errato, ossia P_0 . Nel caso 2 al contrario dobbiamo trovare il bit errato guardando la codifica dei 4 bit di controllo. Infatti guardando C come fosse una parola su 4 bit, la sua decodifica decimale ci fornisce esattamente il numero della posizione del bit errato.
In generale questo tipo di schema rileverà in molti casi anche più di 2 bit errati, ma lo schema stesso non garantisce che questi errori saranno rilevati. Nella maggior parte delle situazioni ci troveremo nel caso 2 quando avremo un numero di bit errati dispari e ci troveremo nel caso 3 quando avremo un numero di bit errati pari. Ovviamente gli errori a 3 bit (o numeri dispari superiori a 1 in generale) non vengono corretti da un codice Hamming di questo tipo. Se provassimo a correggere il valore ricevuto potremmo correggerlo erroneamente, in quanto il valore ricevuto è troppo lontano dal valore effettivamente inviato. Quindi sarà necessario che il canale di trasmissione sia sufficientemente affidabile, altrimenti avremo bisogno di rilevare errori di livello superiore con altre tecniche di rilevazione di errori (ad esempio, un CRC su un intero messaggio).

2.3 Esempio

Facciamo alcuni esempi pratici per capire come il ricevitore può rilevare e correggere eventuali errori. Supponiamo di avere un canale di trasmissione sufficientemente affidabile che ci fornirà al massimo 2 bit errati ad ogni trasmissione. Nella seguente tabella viene riportata la parola su 11 bit in input:

Posizione	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Valore	1	1	0	0	1	0	1	0	1	1	0

Calcoliamo adesso di bit di parità:

$$P_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$P_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$P_4 = D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$P_8 = D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$P_0 = D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_1 \oplus P_2 \oplus P_4 \oplus P_8 =$$
$$1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

Nella seguente tabella sono riportati i bit di parità e della parola per ricordarci il numero della posizione dei vari bit nella codifica:

Posizione	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Bit	P_1	P_2	D_1	P_4	D_2	D_3	D_4	P_8	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	P_0
Valore	1	1	1	1	1	0	0	0	1	0	1	0	1	1	0	1

Adesso vediamo alcuni casi differenti per capire come rilevare e, all'occorrenza, come correggere eventuali errori.

2.3.1 Singolo bit errato

In questo caso supponiamo che il bit D_8 venga ricevuto come 1 anziché come 0. Il ricevitore calcolerà i seguenti bit di controllo:

$$C_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_4 = P_4 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C_8 = P_8 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$P = D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_1 \oplus P_2 \oplus P_4 \oplus P_8 \oplus P_0 = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$C = C_8 C_4 C_2 C_1 = 1100$$

In questo caso siamo nella condizione $C \neq 0$ e $P = 1$, quindi abbiamo un unico errore rilevato. Per sapere qual è il bit errato basta osservare alla codifica decimale di C. In questo caso la sua codifica è 12, perciò il bit errato sarà esattamente il bit numero 12 della parola codificata. Se andiamo nella tabella riportata sopra, possiamo subito notare come il bit numero 12 corrisponda esattamente al bit D_8 della parola in input.

2.3.2 Due bit errati

In questo caso consideriamo 2 bit errati, D_8 e P_2. Calcoliamo come sempre i bit di controllo:

$$C_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C_4 = P_4 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C_8 = P_8 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C = C_8 C_4 C_2 C_1 1110$$

$$P = D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_1 \oplus P_2 \oplus P_4 \oplus P_8 \oplus P_0 = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

In questo caso siamo nella condizione in cui $C \neq 0$ e $P = 0$, quindi è stato rilevato un doppio errore e non è possibile capire quali siano i bit errati. Ricordiamo che dalle specifiche non viene richiesto di correggere errori doppi.

2.3.3 Singolo bit errato in uno dei bit di parità

Proviamo adesso ad introdurre un errore solo nel bit P_0. Calcoliamo i bit di controllo:

$$C_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_4 = P_4 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_8 = P_8 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C = C_8 C_4 C_2 C_1 0000$$

$$P = D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_1 \oplus P_2 \oplus P_4 \oplus P_8 \oplus P_0 =$$
$$1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

In questo caso siamo nella condizione $C = 0$ e $P = 1$, quindi sappiamo subito che il bit errato è proprio P_0.

2.3.4 Tre bit errati

Anche se non abbiamo previsto questo caso, guardiamo come si comporta il sistema quando consideriamo 3 bit errati, D_8, D_6 e P_2. Calcoliamo come sempre i bit di controllo:

$$C_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_4 = P_4 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C_8 = P_8 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C = C_8 C_4 C_2 C_1 = 0100$$

$$P = D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_1 \oplus P_2 \oplus P_4 \oplus P_8 \oplus P_0 =$$
$$1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$$

In questo caso siamo nella condizione $C \neq 0$ e $P = 1$, quindi in questo caso verrebbe indicata la presenza di 1 errore quando in realtà erano 3. Come indicato già in precedenza, in generale quando il numero di bit errati è dispari ci troveremo in questa situazione, ma solo se il numero di bit errati è 1 possiamo correggere l'errore.

2.3.5 4 bit errati

Anche se non abbiamo previsto questo caso, guardiamo come si comporta il sistema quando consideriamo 4 bit errati, D_1 , D_8 , D_6 e P_2 . Calcoliamo come sempre i bit di controllo:

$$C_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C_4 = P_4 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C_8 = P_8 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C = C_8 C_4 C_2 C_1 = 1111$$

$$P = D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_1 \oplus P_2 \oplus P_4 \oplus P_8 \oplus P_0 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

In questo caso siamo nella condizione $C \neq 0$ e $P = 0$, quindi in questo caso verrebbe indicata la presenza di 2 errori quando in realtà erano 4. Come già indicato, in generale quando il numero di bit errati è pari ci troveremo in questa situazione.

2.3.4 Nessun errore

Per completezza di trattazione guardiamo infine il caso in cui non ci siano errori. Calcoliamo i bit di controllo:

$$C_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_4 = P_4 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C_8 = P_8 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C = C_8 C_4 C_2 C_1 = 0000$$

$$P = D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus P_1 \oplus P_2 \oplus P_4 \oplus P_8 \oplus P_0 = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$$

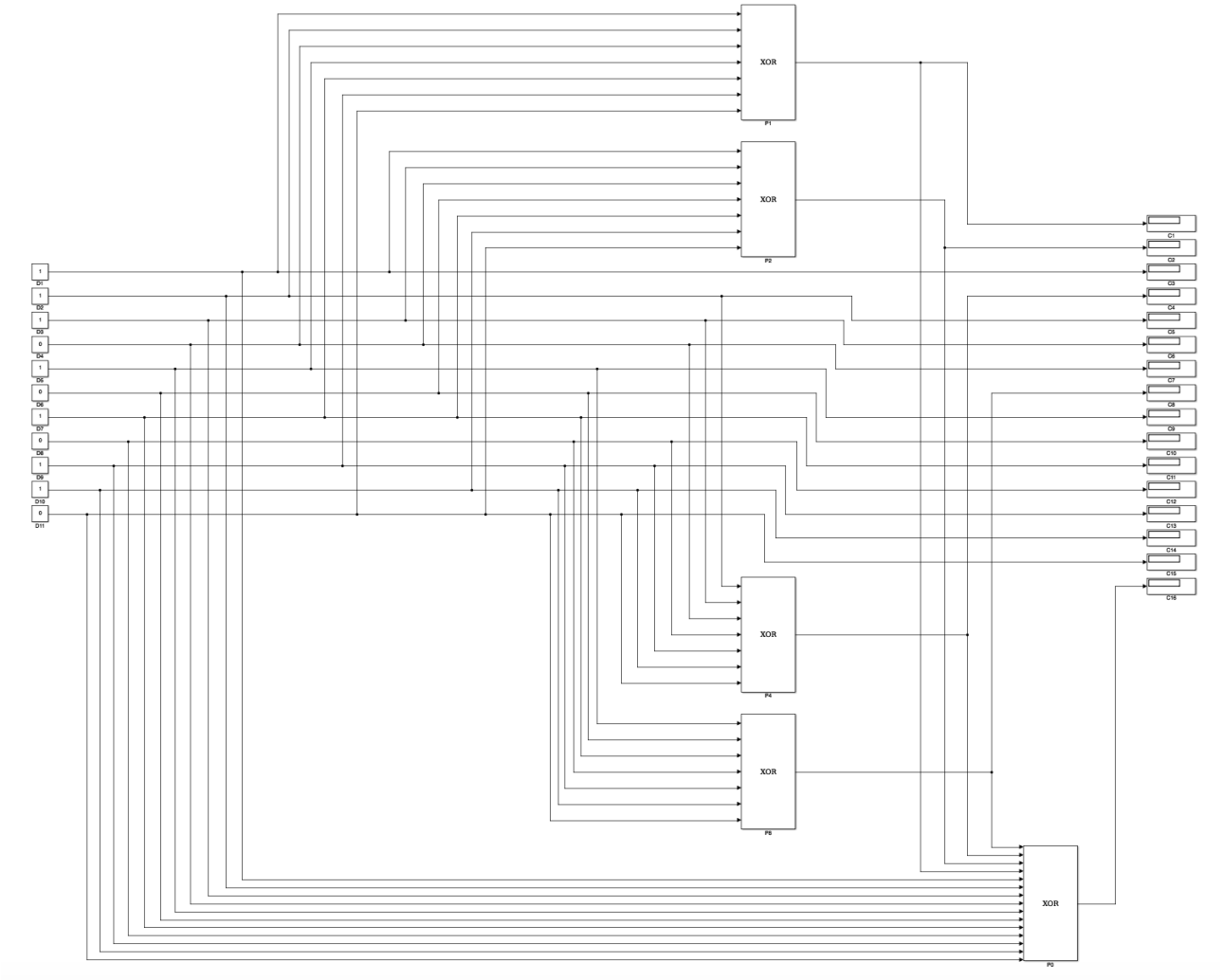
In questo caso siamo nella condizione $C = 0$ e $P = 0$, quindi non so stati rilevati errori.

2.4 Applicazioni

La codifica di Hamming è ancora oggi largamente utilizzata per fare rilevazione e correzione di errori all'interno dei supporti di memorizzazione. Infatti nelle memorie di tipo DRAM, le varie interferenze elettro-magnetiche possono portare alla variazione di un bit. Recenti studi hanno dimostrato che nelle moderne memorie DRAM il numero di bit errati per ogni ora di lavoro può arrivare a valori compresi fra 10^{-10} e 10^{-17}. Questo fa capire come in generale il numero di bit errati non sia mai così elevato e quindi una tecnica semplice ma efficace come l' Hamming code considerato è ritenuta più che sufficiente.

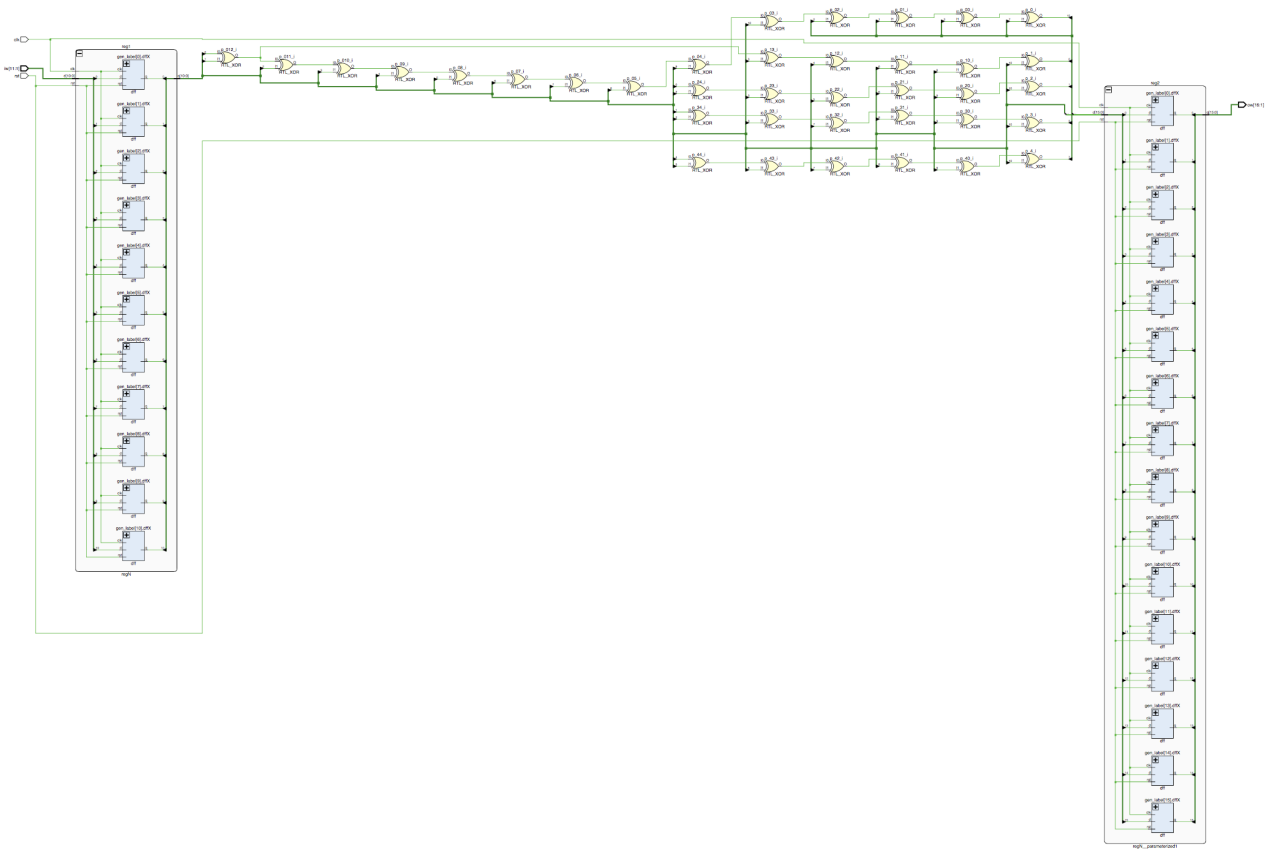
3 Architettura implementata

L'architettura implementata può essere riassunta attraverso il seguente schema a blocchi:



In generale il precedente modello Matlab è stato utilizzato per generare un certo numero di Testbech (confermati anche dagli script Matlab allegati). Questo modello infatti, ci fornisce il risultato esatto che il sistema deve produrre al cambiare dei vari input. Da notare che fra i testbench utilizzati, ce ne sono alcuni che testano anche il decoder che è stato implementato anche se non richiesto. In particolar modo per testare il funzionamento del decoder il testbench è stato scelto grazie allo script Matlab, non avendo creato uno schema a blocchi per il decoder che sarebbe stato di difficile comprensione. Per la precisione sono state implementate 2 versioni del decoder, dell'encoder e dei testbench. Nella prima versione è stata implementata la circuiteria come semplice rete combinatoria, mentre nella seconda la rete è stata inserita fra 2 registri (uno di input e uno di output) per creare un path registro-logica-registro utile al momento della sintesi con il tool Vivado.

Viene riportato di seguito lo schematico relativo all'encoder prodotto dal tool Vivado al momento della sintesi:



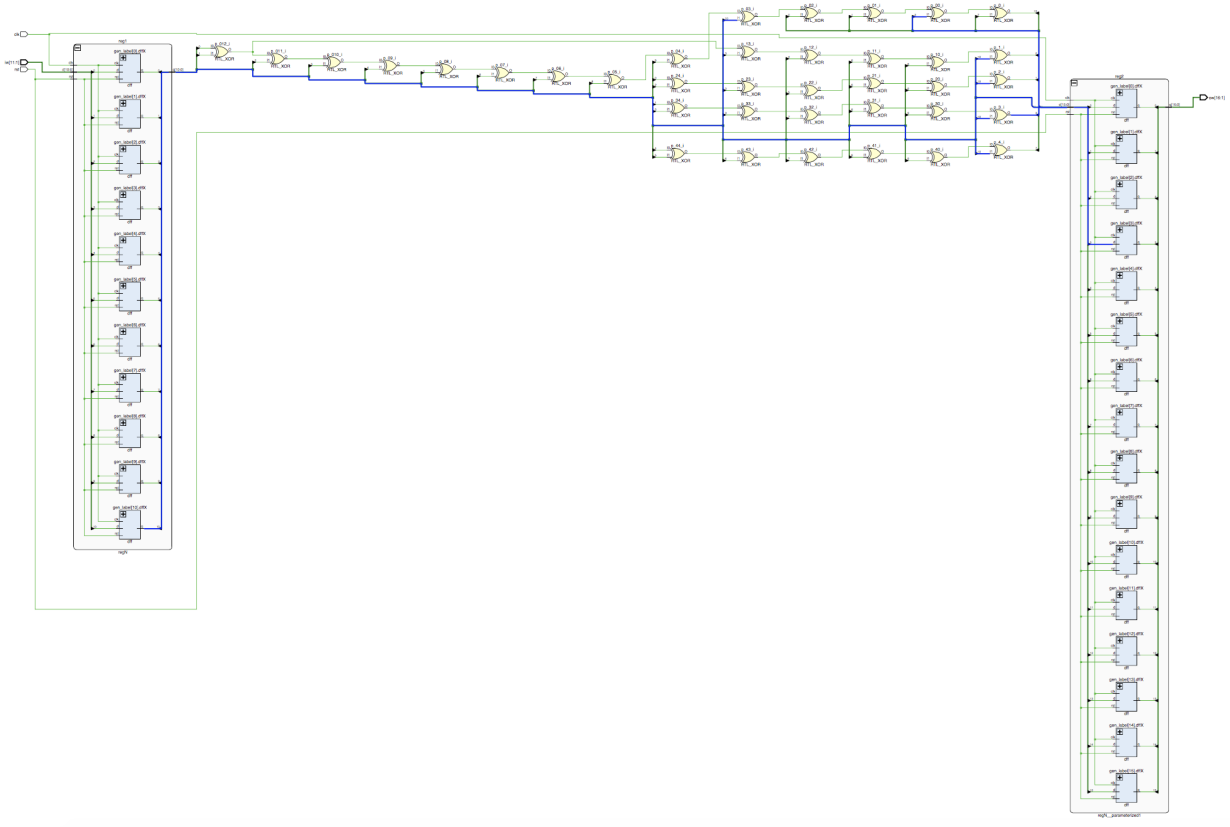
4 Sintesi con Vivado Tool

4.1 Introduzione

Come già detto, sono state realizzate 2 versioni differenti sia dell'encoder che del decoder. Questo perché nella soluzione originale, dovendo creare una semplice rete combinatoria, non erano stati introdotti registri. Per poter procedere alla sintesi della rete sono stati introdotti 2 registri, uno in ingresso e uno in uscita, per creare così un percorso registro-logica-registro e poter così procedere alla sintesi della rete stessa analizzando in dettaglio i requisiti temporali. Inoltre è necessario precisare che la sintesi del decoder non è stata effettuata per i motivi indicati nel paragrafo 4.3.

4.2 Sintesi Encoder

Durante la sintesi dell'encoder (come già detto la versione con i 2 registri), la soluzione trovata da Vivado prevede 8 LUT e 27 Flip Flop. Al momento sintesi è stato creato un clock con un periodo di 4 ns, cioè con una frequenza di 250 Mhz che è stato associato al clock dei registri. Al momento dell'implementazione è stato individuato il cosiddetto path critico, cioè quello che collega l'input D_{11} all'output C_4 :



Il path indicato presenta un worst negative slack positivo pari a $T_{slack} = 2.319\,ns$, mentre il total delay sul path è dato da $T_{Total\ Delay} = 1.649\,ns$ (dove $T_{Net\ Delay} = 0.992\,ns$ e $T_{Logic\ Delay} = 0.657\,ns$). Per il calcolo della frequenza massima del sistema, dobbiamo calcolare quindi il periodo di clock minimo utilizzando la formula:

$$T_{CK} \geq T_{setup} + T_{logic} + T_{cq}$$

Poniamo $T_D = T_{setup} + T_{logic} + T_{cq}$. Sapendo che vogliamo trovare esattamente il minimo periodo di clock, consideriamo la seguente uguaglianza:

$$T_{CK} = T_D + T_{slack} \quad \text{con} \quad T_{slack} = 0$$

Inoltre sappiamo che:

$$T_{logic} = Net\ Delay\ di\ Vivado$$
$$T_{cq} = Logic\ Delay\ di\ Vivado$$
$$T_D = T_{CK} - T_{slack}$$
$$T_{setup} = T_D - T_{cq} - T_{logic}$$

Prendiamo in considerazione i dati relativi al periodo di clock pari a $T_{CK} = 4\,ns$, pertanto:

$T_{CK} = 4\,ns$

$T_{slack} = 2.319\,ns$

$T_D = 1.681\,ns$

$T_{cq} = 0.657\,ns$

$T_{logic} = 0.992\,ns$

$T_{setup} = T_{CK} - T_{slack} - T_{cq} - T_{logic} = 4 - 2.319 - 0.657 - 0.992 = 0.032\,ns$

Per ottenere il periodo di clock minimo dobbiamo porre $T_{slack} = 0$, ovvero dobbiamo utilizzare un $T_{CK} = T_D$. Pertanto il minimo periodo che possiamo utilizzare sarà $T_{CK} = 1.681\,ns$, che produce un frequenza massima pari a $F_{MAX} = 594.884\ Mhz$.

Come conferma, si potrebbe utilizzare il periodo di clock così trovato per effettuare una nuova sintesi. Ciò che ci si aspetta sarebbe ottenere un worst negative slack nullo. In realtà cambiando il clock, probabilmente il tool Vivado ottimizza l'architettura (come vedremo ogni volta che viene cambiato utilizzato varia anche il path critico) producendo ancora una volta uno slack positivo. Ovviamente la procedura può essere reiterata fino a che non si annulla lo slack stesso. Nel sottoparagrafo 4.2.1 sono stati riportati tutti i risultati ottenuti reiterando la procedura sopra indicata.

Il tool non ha riportato warning rilevanti, se non quelli relativi al non assegnamento delle variabili ad I/O pin della scheda selezionata e all'utilizzato dei valori di default per i segnali stessi. Questi warning non sono da considerarsi rilevanti in quanto riguardano soltanto la parte più pratica della sintesi, cioè l'assegnamento dei segnali ad I/O pin specifici della scheda.

4.2.1 Risultati ottenuti nel calcolo della frequenza massima

Nella seguente tabella sono stati riportati i valori ottenuti dal tool Vivado ad ogni iterazione:

# Iterazioni	T_{CK}	T_{slack}	T_{D}	T_{cq}	T_{logic}	T_{setup}	T_{CKnext}	F_{MAX}
1	4 ns	2.319 ns	1.681 ns	0.657 ns	0.992 ns	0.032 ns	1.681 ns	250 MHz
2	1.681 ns	0.411 ns	1.27 ns	0.438 ns	0.803 ns	0.029 ns	1.27 ns	594.884 MHz
3	1.27 ns	0.145 ns	1.125 ns	0.438 ns	0.628 ns	0.059 ns	1.125 ns	787.401 MHz
4	1.125 ns	-0.023 ns	1.148 ns	0.438 ns	0.645 ns	0.065 ns	1.148 ns	888.889 MHz
5	1.148 ns	-0.163 ns	1.311 ns	0.535 ns	0.747 ns	0.029 ns	1.311 ns	871.080 MHz
6	1.311 ns	0.227 ns	1.084 ns	0.438 ns	0.616 ns	0.030 ns	1.084 ns	762.776 MHz
7	1.084 ns	-0.166 ns	1.25 ns	0.490 ns	0.734 ns	0.026 ns	1.25 ns	922.509 MHz
8	1.25 ns	-0.036 ns	1.286 ns	0.535 ns	0.692 ns	0.059 ns	1.286 ns	800 MHz
9	1.286 ns	0.186 ns	1.1 ns	0.438 ns	0.631 ns	0.031 ns	1.1 ns	777.605 MHz
10	1.1 ns	-0.060 ns	1.16 ns	0.438 ns	0.694 ns	0.028 ns	1.16 ns	909.091 MHz
11	1.16 ns	0.011 ns	1.149 ns	0.535 ns	0.584 ns	0.030 ns	1.149 ns	862.069 MHz
12	1.149 ns	0.009 ns	1.14 ns	0.535 ns	0.576 ns	0.029 ns	1.14 ns	870.322 MHz
13	1.14 ns	-0.008 ns	1.148 ns	0.438 ns	0.645 ns	0.065 ns	1.148 ns	877.193 MHz
14	1.148 ns	-0.163 ns	1.311 ns	0.535 ns	0.747 ns	0.029 ns	1.311 ns	871.080 MHz

Nella seguente tabella sono stati riportati i path critici individuati ad ogni singola iterazione:

Iterazione	Sorgente (Data input)	Destinazione (Codifica)
1	D_{11}	C_4
2	D_6	C_8
3	D_6	C_8
4	D_6	C_8
5	D_7	C_2
6	D_9	C_4
7	D_1	C_1
8	D_8	C_8
9	D_6	C_{16}
10	D_7	C_8
11	D_{11}	C_2
12	D_7	C_2
13	D_5	C_1
14	D_7	C_2

Come si può notare al passo 14 troviamo un valore per il clock esattamente identico a quello trovato al passo 5, pertanto dal passo 14 in poi le iterazioni si ripetono senza mai trovare uno slack nullo. È da sottolineare inoltre come, al passo 3, sia stato ottenuto un worst pulse width slack (non trattato a lezione) negativo. Quest'ultimo parametro ci fornirebbe un vincolo sul periodo minimo di clock utilizzabile pari a $T_{CK} = 1.592\,ns$, quindi una frequenza massima di $F_{MAX} = 628.141\,MHz$. Le iterazioni sono state comunque portate al termine (o almeno fino alla scoperta di un ciclo infinito), sia per completezza, sia perché questo parametro non è stato trattato a lezione e quindi non è stata data particolare attenzione allo stesso, come suggerito anche dal dottorando Gabriele Meoni, contattato alla luce del problema evidenziato. Pertanto, considerando solo il parametro worst negative slack, il minore tempo di clock raggiungibile è $T_{CK} = 1.149\,ns$ che corrisponde ad una frequenza pari a $F_{MAX} = 870.322\,MHz$.

4.3 Sintesi Decoder

Come già anticipato, non è stata effettuata la sintesi del decoder in quanto nel codice VHDL prodotto andiamo a complementare un bit in una posizione variabile indicata dalla codifica decimale dei bit di controllo. La codifica decimale della posizione si trasforma dal punto di vista del tool Vivado in un range di valori dinamico e quindi non costante che non permettono al tool stesso di procedere con la sintesi. Essendo inevitabile un approccio dinamico (dettato dalla natura del circuito stesso) la sintesi del componente non è stata fatta. La mancata sintesi non viene evidenziata come un problema in quanto non richiesta dalle specifiche di progetto. Ricordo infatti che le specifiche richiedevano di sviluppare soltanto il componente encoder, mentre la circuiteria del decoder sono state aggiunte per completezza di trattazione.

5 Conclusioni

In generale questo tipo di Error Correcting Code è molto utile in quanto è possibile rilevare 2 errori e correggerne 1 senza dover osservare dati passati. Si tratta quindi di una soluzione semplice da implementare e piuttosto efficace, ideale per quelle applicazioni che di per sé sono poco soggette ad errori e che necessitano di un ulteriore controllo (Hamming code) che non sia troppo pesante.