

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI
CALCOLATORI ELETTRONICI
NUOVO E VECCHIO ORDINAMENTO DIDATTICO**
26 Febbraio 2002

MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI

ESERCIZIO 1 (NO: 10 punti – VO: 8 punti)

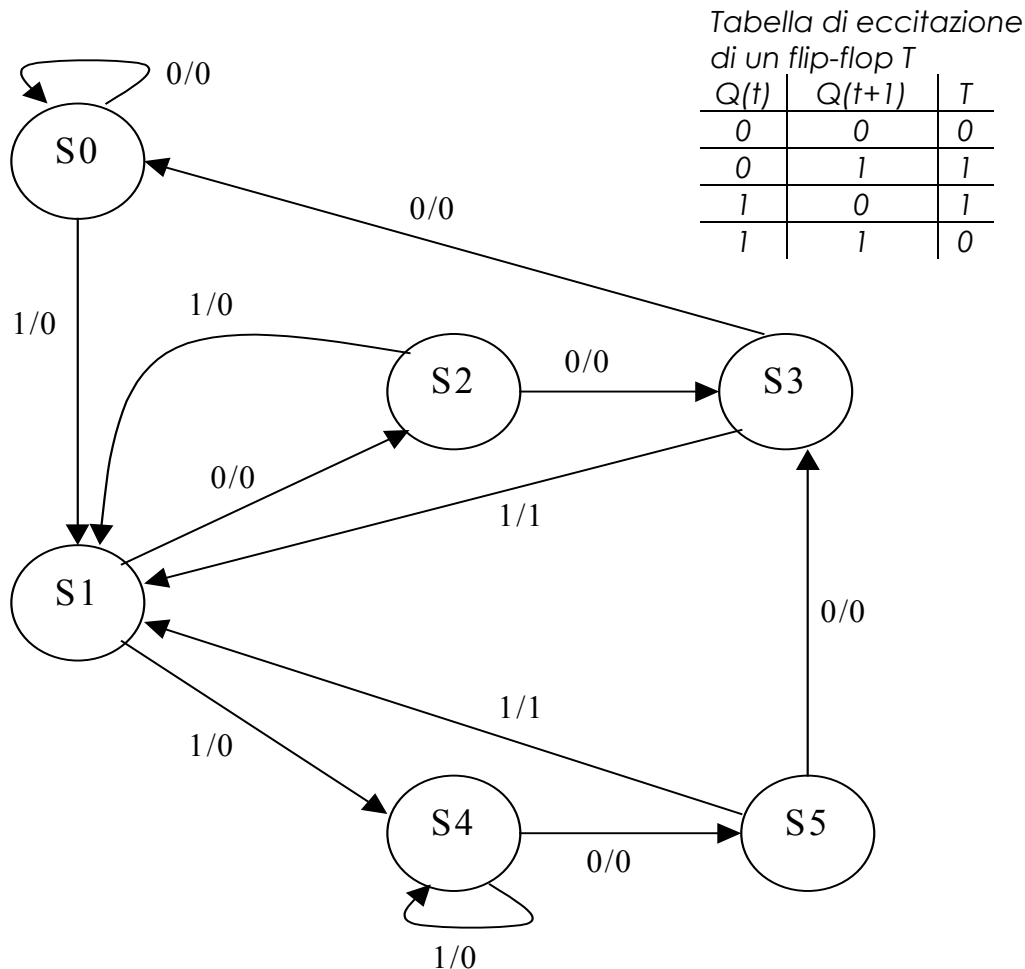
Progettare un circuito sequenziale che presenti un ingresso e una uscita, posta a 1 in corrispondenza della sequenza 1101 oppure 1001.

Calcolare le forme minime per le variabili di eccitazione dei flip flop (tipo T) usando le mappe di Karnaugh.

Qual è la conseguenza dell'uso di eventuali don't care sull'uscita?

Soluzione.

Scriviamo innanzi tutto il grafo degli stati del sistema:



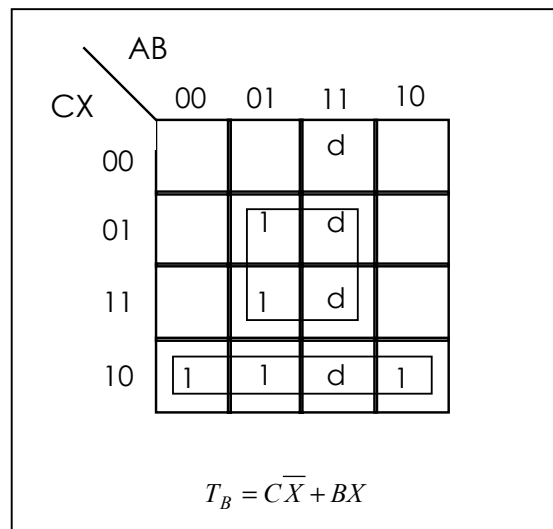
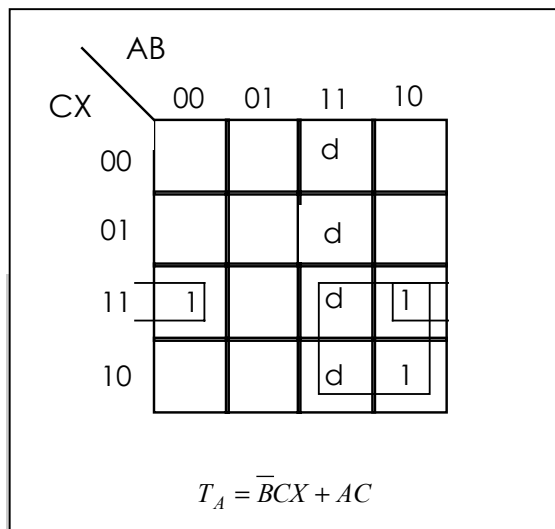
I

Indicando con A, B, C le variabili di stato, con X l'ingresso e con Z l'uscita, la relativa tabella delle transizioni è:

A	B	C	A'	T _A	B'	T _B	C'	T _C	Z
X=0									
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	0	1	0
0	1	0	0	0	1	0	1	1	0
0	1	1	0	0	0	1	0	1	0
1	0	0	1	0	0	0	1	1	0
1	0	1	0	1	1	1	1	0	0
1	1	0	D	D	D	D	D	D	0
1	1	1	D	D	D	D	D	D	0
X=1									
0	0	0	0	0	0	0	1	1	0
0	0	1	1	1	0	0	0	1	0
0	1	0	0	0	0	1	1	1	0
0	1	1	0	0	0	1	1	0	1
1	0	0	1	0	0	0	0	0	0
1	0	1	0	1	0	0	1	0	1
1	1	0	D	D	D	D	D	D	0
1	1	1	D	D	D	D	D	D	0

Si noti che nel caso dell'uscita non è stato usato il *don't care*, per rispettare il vincolo di progetto richiedente che tale variabile assuma il valore 1 solo dove indicato nel grafo degli stati.

Possiamo ora disegnare le mappe di Karnaugh in tutti i casi:



		AB			
		00	01	11	10
CX	00		1	d	1
	01	1	1	d	
	11	1		d	
	10	1	1	d	

$$T_C = \bar{A} \cdot \bar{B}C + \bar{A} \cdot \bar{C}X + \bar{A}C \cdot \bar{X} + B\bar{X}$$

		AB			
		00	01	11	10
CX	00				
	01				
	11		1		1
	10				

$$Z = \bar{A}BCX + \bar{A}\bar{B}CX$$

ESERCIZIO 2 (NO: 7 punti – VO: 6 punti)

I trasferimenti di parole a/dalla memoria di un calcolatore sono codificate utilizzando il codice di Hamming. Si consideri la stringa di 12 bit 101001101110 (il bit meno significativo è a sinistra), risultata della codifica di una parola di N bit secondo il codice di Hamming. **Spiegando bene ogni passo del ragionamento:**

- 1) (2 punti) calcolare N, supponendo di aver fatto uso del numero minimo di bit di controllo necessario per una stringa di 12 bit;
- 2) (VO: 2 – NO: 3 punti) scrivere la parola di N bit a partire dalla stringa data;
- 3) (3 punti) indicare eventuali errori nella stringa codificata, specificando quale dei bit è stato alterato.

Soluzione.

- 1) Deve venire rispettata la condizione:

$$2^K \geq N + K + 1 \quad (1),$$

dove K è il numero di bit di controllo inseriti. Essendo $N + K = 12$, si evince dalla (1) che il numero minimo di bit di controllo richiesto è 4. Da cui $N = 8$.

- 2) Nella codifica di Hamming, la sequenza in ingresso presenta la seguente struttura:

c_0	c_1	b_0	c_2	b_1	b_2	b_3	c_3	b_4	b_5	b_6	b_7
1	0	1	0	0	1	1	0	1	1	1	0

Dove $c_0 \dots c_3$ sono i quattro bit costituenti il vettore di controllo, e $b_0 \dots b_7$ gli otto bit trasmessi. La sequenza ricevuta è 10111110.

- 3) Per verificare la presenza di un errore, dobbiamo ricalcolare il vettore di controllo a partire dalla sequenza ricevuta. Si ha:

$$c'_0 = b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0$$

$$c'_1 = b_0 \oplus b_2 \oplus b_3 \oplus b_5 \oplus b_6 = 1$$

$$c'_2 = b_1 \oplus b_2 \oplus b_3 \oplus b_7 = 0$$

$$c'_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 1$$

Il passo successivo è calcolare il vettore di errore dato dalla differenza dei vettori di controllo c e c' (ricordiamo che somma e differenza tra bit producono lo stesso risultato):

$$e_0 = c_0 \oplus c'_0 = 1$$

$$e_1 = c_1 \oplus c'_1 = 1$$

$$e_2 = c_2 \oplus c'_2 = 0$$

$$e_3 = c_3 \oplus c'_3 = 1$$

Poiché il vettore risultante 1011 non è nullo, vi è un errore nella stringa di 12 bit data e precisamente nella posizione indicata dal vettore di errore tradotto in notazione decimale. Il bit sbagliato è quindi l'undicesimo (b_6), e la parola corretta è 10111100.

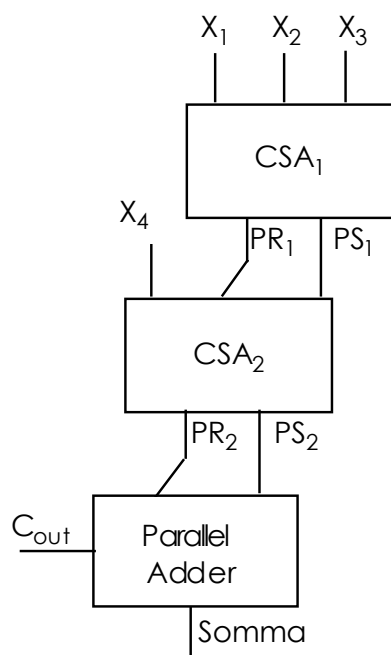
ESERCIZIO 3 (NO: 8 punti – VO:7)

Si vogliano sommare i seguenti quattro numeri a 8 bit: $X_1 = 00110010$, $X_2 = 01011100$, $X_3 = 01010111$, $X_4 = 01101110$. Disegnare lo schema che permette di eseguire tale somma usando due addizionatori del tipo "Carry Save Adder" ed un "Parallel Adder" finale. Gli addizionatori "Carry Save Adder" lavorano su tre operandi. Precisare il valore assunto dalla Pseudosomma e dal Pseudoriporto all'uscita del primo e del secondo "Carry Save Adder" e le operazioni eseguite per ottenere la somma finale. (VO: 4 punti)

(solo VO: 3 punti) Nell'ipotesi che d sia il ritardo per il calcolo di somma e riporto per un full-adder, calcolare il ritardo del presente sistema, indicando il vantaggio che deriva rispetto all'uso di soli parallel adder.

Soluzione

Schema:



L'addizionatore "Carry Save Adder" fornisce in uscita due parole a 8 bit: la Pseudosomma e il Pseudoriporto. La Pseudosomma (PS) rappresenta il risultato della somma dei tre addendi senza considerare i riporti, mentre il Pseudoriporto (PR) rappresenta solo i riporti relativi a ciascuno stadio. Il risultato completo lo si ottiene sommando $PS + 2 PR$. Un possibile schema per eseguire la somma richiesta è riportato in figura, dove con un tratto obliquo si è indicato lo "scorrimento" verso sinistra (moltiplicazione per 2) necessario per sommare il PR al PS.

Somma di $X_1 + X_2 + X_3$:

$$PS_1 = 00111001, PR_1 = 01010110; 2 PR_1 = 10101100$$

Somma di $X_4 + PS_1 + 2 PR_1$:

$$PS_2 = 11111011, PR_2 = 00101100; 2 PR_2 = 01011000$$

$$\text{Somma finale} = PS_2 + 2 PR_2 = 01010011, C_{out} = 1$$

Per effettuare la somma di 4 addendi usando solo parallel adder sarebbero necessari 3 parallel adder, dunque il vantaggio non è nel numero di componenti. Il vantaggio risiede nella maggior velocità dell'operazione: gli addizionatori del tipo carry save non hanno i ritardi dovuti alla propagazione del riporto, dunque il risultato si ottiene dopo un ritardo corrispondente a una rete a due o tre livelli, a seconda della rete usata per calcolare somma e riporto. L'unico ritardo dovuto alla propagazione del riporto è presente solo nell'ultimo stadio. Se indichiamo con d il ritardo per il calcolo di somma e riporto per un full-adder, un parallel adder a 8 bit produce il risultato con un ritardo pari a $8d$, mentre un carry save adder produce il risultato in termini di PS e PR con un ritardo pari a d . Nel caso in esame (somma di 4 addendi con 2 addizionatori carry-save e un addizionatore parallelo) avremo un ritardo pari a $d + d + 8d = 10d$, mentre usando 3 parallel adder il ritardo nella produzione del risultato sarebbe pari a $3 \cdot 8d = 24d$.

ESERCIZIO 4 (NO: 8 punti - VO: 7 punti)

Un calcolatore ha un sistema di memoria virtuale a tre livelli costituita da: cache, memoria primaria e disco. La lettura di una parola che si trova già memorizzata nella cache richiede 15 ns. La lettura di una parola dalla memoria primaria e il suo trasferimento in cache richiedono complessivamente 40 ns. La lettura di una parola dal disco e il suo trasferimento in memoria primaria richiedono complessivamente 10 ms. La probabilità che una parola si trovi già in cache è pari a 0.95 mentre la probabilità che una parola si trovi in memoria primaria quando non è presente nella cache è pari a 0.6. Calcolare il tempo medio di accesso al sistema di memoria.

Soluzione:

Si devono considerare tre possibili situazioni:

Posizione della parola richiesta	Probabilità	Tempo di accesso totale
In cache	0.95	15 ns
Non in cache ma in memoria primaria	$(1 - 0.95) * 0.6 = (0.05) * 0.6 = 0.03$	$40\text{ns} + 15\text{ ns} = 55\text{ ns}$
Non in cache né in memoria primaria	$(1 - 0.95) * (1 - 0.6) = 0.05 * 0.4 = 0.02$	$10\text{ ms} + 40\text{ ns} + 15\text{ ns} = 10.000.055\text{ ns}$

Pertanto il tempo medio di accesso alla gerarchia di memoria è dato da:

$$T_m = 0.95 * 15 + 0.03 * 55 + 0.02 * 10.000.055 = 200.017\text{ ns}$$

ESERCIZIO 5 (VO: 5 punti)

Spiegare in modo chiaro e sintetico in cosa consiste la classificazione di Flynn delle architetture parallele.

Soluzione.

Vedere le dispense del corso.