

SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI CALCOLATORI ELETTRONICI

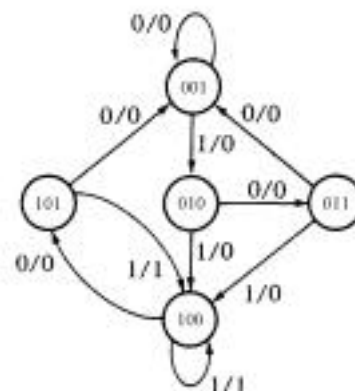
8 Giugno 2000

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

IMPORTANTE: Leggere la NOTA riportata a pag. 4 di questo documento

ESERCIZIO 1

Si consideri il grafo degli stati rappresentato in figura. Progettare la rete logica sequenziale che realizza le transizioni rappresentate nel grafo, usando flip flop di tipo T. Scrivere le espressioni delle uscite e minimizzare le funzioni delle variabili di eccitazione dei flip-flop usando le mappe di Karnaugh.



Soluzione:

Tabella di flusso

Stato presente	Stato futuro/uscita	
	x = 0	x = 1
000	ddd/d	ddd/d
001	001/0	010/0
010	011/0	100/0
011	001/0	100/0
100	101/0	100/1
101	001/0	100/1
110	ddd/d	ddd/d
111	ddd/d	ddd/d

Tabella delle transizioni

A	B	C	x	A(t+1)	T _A	B(t+1)	T _B	C(t+1)	T _C
0	0	0	0	d	d	d	d	d	d
0	0	0	1	d	d	d	d	d	d
0	0	1	0	0	0	0	0	1	0
0	0	1	1	0	0	1	1	0	1
0	1	0	0	0	0	1	0	1	1
0	1	0	1	1	1	0	1	0	0
0	1	1	0	0	0	0	1	1	0
0	1	1	1	1	1	0	1	0	1
1	0	0	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	0	0
1	0	1	0	0	1	0	0	1	0
1	0	1	1	1	0	0	0	0	1
1	1	0	0	d	d	d	d	d	d
1	1	0	1	d	d	d	d	d	d
1	1	1	0	d	d	d	d	d	d
1	1	1	1	d	d	d	d	d	d

Tabella di eccitazione di un flip-flop T

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

<p>AB</p> <p>C x</p> <table> <tr><td>00</td><td>01</td><td>11</td><td>10</td></tr> <tr><td>00</td><td>d</td><td></td><td>d</td></tr> <tr><td>01</td><td>d</td><td>1</td><td>d</td></tr> <tr><td>11</td><td></td><td>1</td><td>d</td></tr> <tr><td>10</td><td></td><td></td><td>d</td></tr> </table> <p>$T_A = Bx + AC\bar{x}$</p>	00	01	11	10	00	d		d	01	d	1	d	11		1	d	10			d	<p>AB</p> <p>C x</p> <table> <tr><td>00</td><td>01</td><td>11</td><td>10</td></tr> <tr><td>00</td><td>d</td><td></td><td>d</td></tr> <tr><td>01</td><td>d</td><td>1</td><td>d</td></tr> <tr><td>11</td><td>1</td><td>1</td><td>d</td></tr> <tr><td>10</td><td></td><td>1</td><td>d</td></tr> </table> <p>$T_B = \bar{A}x + BC$</p>	00	01	11	10	00	d		d	01	d	1	d	11	1	1	d	10		1	d	<p>AB</p> <p>C x</p> <table> <tr><td>00</td><td>01</td><td>11</td><td>10</td></tr> <tr><td>00</td><td>d</td><td>1</td><td>d</td></tr> <tr><td>01</td><td>d</td><td></td><td>d</td></tr> <tr><td>11</td><td>1</td><td>1</td><td>d</td></tr> <tr><td>10</td><td></td><td></td><td>d</td></tr> </table> <p>$T_C = \bar{C}\bar{x} + Cx$</p>	00	01	11	10	00	d	1	d	01	d		d	11	1	1	d	10			d	<p>AB</p> <p>C x</p> <table> <tr><td>00</td><td>01</td><td>11</td><td>10</td></tr> <tr><td>00</td><td>d</td><td></td><td>d</td></tr> <tr><td>01</td><td>d</td><td></td><td>d</td></tr> <tr><td>11</td><td></td><td>d</td><td>1</td></tr> <tr><td>10</td><td></td><td>d</td><td></td></tr> </table> <p>$z = Ax$</p>	00	01	11	10	00	d		d	01	d		d	11		d	1	10		d	
00	01	11	10																																																																																
00	d		d																																																																																
01	d	1	d																																																																																
11		1	d																																																																																
10			d																																																																																
00	01	11	10																																																																																
00	d		d																																																																																
01	d	1	d																																																																																
11	1	1	d																																																																																
10		1	d																																																																																
00	01	11	10																																																																																
00	d	1	d																																																																																
01	d		d																																																																																
11	1	1	d																																																																																
10			d																																																																																
00	01	11	10																																																																																
00	d		d																																																																																
01	d		d																																																																																
11		d	1																																																																																
10		d																																																																																	

ESERCIZIO 2

Progettare una ALU che esegua le seguenti operazioni su due operandi A e B a n bit: $A+1$, $A-B$, $B-A$, $A+B$.

Soluzione

Per realizzare le quattro operazioni richieste è possibile utilizzare un parallel adder e una rete logica. La selezione delle quattro operazioni può essere effettuata usando due variabili di controllo s_1 ed s_0 e il C_{in} in ingresso al parallel adder. La rete logica avrà in ingresso gli operandi A e B e le due variabili di controllo, mentre in uscita produrrà due operandi X e Y che sommati dal parallel adder forniranno in uscita la funzione specificata su A e B. Nella tabella seguente è stata effettuata una possibile corrispondenza fra valori delle variabili di controllo e funzione desiderata:

Funzione	Variabili di selezione	X	Y	C_{in}
$A + 1$	$s_1 s_0 = 00$	A	0	1
$A - B$	$s_1 s_0 = 01$	A	\bar{B}	1
$B - A$	$s_1 s_0 = 10$	\bar{A}	B	1
$A + B$	$s_1 s_0 = 11$	A	B	0

Per ottenere le funzioni specificate nell'esercizio le variabili X e Y all'ingresso del parallel adder sono funzione di A e B (bit a bit) secondo la tabella seguente (N.B. per semplicità X si ottiene da A e Y da B):

s_1	s_0	X_i	Y_i
0	0	A_i	0
0	1	A_i	\bar{B}_i
1	0	\bar{A}_i	B_i
1	1	A_i	B_i

$$X_i = (\bar{s}_1 + s_0)A_i + s_1 \bar{s}_0 \bar{A}_i$$

$$Y_i = \bar{s}_1 s_0 \bar{B}_i + s_1 B_i$$

ESERCIZIO 3

- I trasferimenti di parole a/dalla memoria di un calcolatore sono codificate utilizzando il codice di Hamming. Si consideri la stringa seguente che rappresenta una parola codificata con il codice di Hamming (il bit meno significativo è a sinistra):
011101010001
 - 1.1. Quale è la dimensione della parola trasmessa (escludendo i bit di codice)?
 - 1.2. Verificare se la stringa ricevuta contiene un errore e in caso affermativo correggerlo
 - 1.3. Scrivere la parola corretta che si voleva trasmettere
- Si consideri un disco rigido con le seguenti caratteristiche: velocità di rotazione = 5400 giri/min, tempo medio di posizionamento = 5 ms, 34 settori per traccia di 512 byte ciascuno. Se il trasferimento di un file richiede 0.1 sec, qual è la dimensione del file (ipotizzare che la testina sia inizialmente posizionata all'inizio del primo settore del file e che il file sia memorizzato in settori appartenenti a tracce diverse).

Soluzione

- 1.1. La stringa trasmessa contiene 12 bit, dunque i bit di codice sono 4 e la parola trasmessa ha dimensione di 8 bit (il numero di bit di codice m è scelto come il più piccolo intero che soddisfa la relazione $m + b \geq 2^m - 1$, dove b indica il numero di bit della parola da codificare. In questo caso $m+b=12$ e il più piccolo intero che soddisfa la relazione precedente è 4).

- 1.2. Se indichiamo con b i bit della parola inviata e con c i bit di codice, la stringa di 12 bit deve essere interpretata nel modo seguente:

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

Il calcolo dei bit di errore è il seguente:

$$e_0 = c_0 \oplus b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$e_1 = c_1 \oplus b_0 \oplus b_2 \oplus b_3 \oplus b_5 \oplus b_6 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$e_2 = c_2 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_7 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$e_3 = c_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 0$$

$$\text{Posizione dell'errore} = \sum_{i=0}^3 e_i 2^i = 1 + 1 \cdot 2 + 1 \cdot 2^2 + 0 \cdot 2^3 = 7$$

Nella stringa il 7° bit (che corrisponde a b_3) è errato.

- 1.3. La parola di 8 bit trasmessa (bit meno significativo a sinistra) è dunque: 10110001.

2. Indicato con x il numero di settori occupato dal file, il tempo medio di lettura si calcola come: $(x - 1)T_{\text{pos}} + (x - 1)T_{\text{lat}} + x T_{\text{trasf}}$, dove i primi due termini, che rappresentano rispettivamente il tempo medio di posizionamento e il tempo medio di latenza, tengono conto che la testina è inizialmente posizionata sul primo settore del file. Il tempo di latenza si calcola come la metà del tempo di rotazione: $T_{\text{lat}} = \frac{1}{2} T_{\text{rot}} = \frac{1}{2} \cdot 60/5400 = 5.56 \text{ ms}$. Il tempo di trasferimento di un settore si calcola come $T_{\text{rot}}/34 = 0.33 \text{ ms}$ perché durante una rotazione completa viene letta una traccia. Il valore di x si ricava facilmente dalla seguente equazione i tempi sono espressi in ms):

$$100 = (x - 1) \cdot 5 + (x - 1) \cdot 5.56 + x \cdot 0.33, \quad x = 10. \text{ Il file occupa 5 KB.}$$

ESERCIZIO 4

Spiegare in maniera chiara e sintetica le caratteristiche della modalità di esecuzione delle operazioni di I/O da programma, I/O mediante interruzioni e I/O mediante DMA..

Soluzione:

Vedi dispense del corso

ESERCIZIO 5

Sia data la sequenza di istruzioni RISC (A, B, X, Y sono riferimenti a locazioni di memoria)

MOV R1, A

MOV R2, B

ADD R3, R1, R2

MOV R4, X

MOV R5, Y

ADD R6, R4, R5

ADD R7, R3, R4

ADD R8, R6, R5

Individuare le istruzioni che operano su dati dipendenti. Spiegare perché possono creare dei "ritardi" nell'esecuzione della pipeline e proporre delle tecniche per risolvere questo problema.

Soluzione

Nelle sequenza di istruzioni assegnata si possono individuare due tipi di dipendenza: il primo riguarda le due istruzioni 'ADD R3, R1, R2' e 'ADD R6, R4, R5' la cui esecuzione dipende dalla velocità del caricamento dei registri R1, R2, R4 ed R5 con i dati letti dalla memoria. Le due istruzioni di somma possono essere bloccate per un certo numero di cicli in attesa che vengano caricati i valori corretti nei registri che contengono gli addendi. Per ovviare a questo inconveniente il compilatore può riordinare il codice spostando la prima istruzione di somma subito prima della seconda istruzione di somma. In questo modo l'istruzione 'ADD R3, R1, R2' viene chiamata dopo un certo numero di cicli dall'istruzione MOV consentendo il caricamento dei registri R1 e R2. La seconda istruzione di ADD a questo punto viene chiamata più tardi rispetto all'organizzazione originaria del codice, riducendo così il ritardo nell'esecuzione. Le successive istruzioni di somma operano su dati dipendenti nel senso che gli addendi di una somma sono i risultati di somme precedenti. In questo caso per evitare ritardi si usa una tecnica 'hardware' detta 'data forwarding' che consente di inviare i risultati di una istruzione nei registri di ingresso della ALU senza aspettare la scrittura dei registri di CPU.

NOTA	Esercizio 1	Esercizio 2	Esercizio 3	Esercizio 4	Esercizio 5
Vecchio Ordinamento	6 punti	5 punti	dom. 1: 5 punti dom. 2: 5 punti	6 punti	6 punti
Nuovo Ordinamento	9 punti	8 punti	7 punti	9 punti	Assente