Architetture 1 – AA 2003-2004
Canale EO – Andrea Sterbini
26 Gennaio 2004

Parte 1

Esercizio 1 (5 punti)

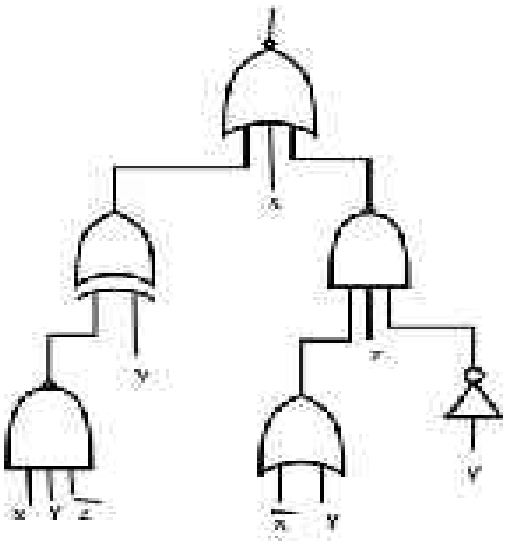
Si dimostri senza usare l' induzione perfetta l' identità:

$$x\,y + \bar{x}\,\bar{y} + x\,\bar{y}\,z = x\,z + \bar{x}\,\bar{y} + \bar{x}\,y\,z$$

Esercizio 2 (10 punti)

Data la rete combinatoria qui a fianco:

- si trovi l'espressione booleana corrispondente all' uscita (5 punti)
- si trovi una rete equivalente che usa solo porte NAND (5 punti)



Esercizio 3 (15 punti)

Si progetti il circuito combinatorio che, dato in input un numero naturale X col segno codificato in complemento a due rappresentato dai 4 bit x_3,x_2,x_1,x_0, fornisce in output il numero **Y=X*3 mod 8** rappresentato dai 3 bit y_2,y_1,y_0 e codificato secondo il codice di Gray.
Si realizzi il circuito:

- con una ROM (5 punti)
- con una PLA (5 punti)
- con multiplexer 4-a-1 e porte NOT (5 punti)

Parte 2

Esercizio 4 (30 punti)

Si progetti il circuito sequenziale sincrono che, dato in input un singolo bit **x**, fornisce l' output **z**=1 se e solo se negli ultimi 4 bit sono presenti 3 o più valori ' 1' .

- si progetti l'automa minimo (10 punti)
- si progetti il circuito usando solo flip-flop di tipo Master-Slave (10 punti)
- si realizzi il circuito usando solo multiplexer 4-a-1 e porte NOT (10 punti)

NOTA: sono sufficienti 8 stati

Esempio di input/output (in cui il tempo scorre da sinistra a destra):
input x: 00101000100100100011011010101111100101001 0
output z: 00000000000000000000011010**1**000011111000000000

Nota: il bit in grassetto era (erratamente) zero nel testo originale.

Soluzioni

Esercizio 1

L' identità è falsa, infatti, portando entrambi i membri in forma canonica si ottengono i passaggi seguenti, in cui sono evidenziati i termini diversi:

$$x\,y + \bar{x}\,\bar{y} + x\,\bar{y}\,z \qquad x\,z + \bar{x}\,\bar{y} + \bar{x}\,y\,z$$
$$x\,y(z+\bar{z}) + \bar{x}\,\bar{y}(z+\bar{z}) + x\,\bar{y}\,z \qquad x\,z(y+\bar{y}) + \bar{x}\,\bar{y}(z+\bar{z}) + \bar{x}\,y\,z$$
$$x\,y\,z + \boldsymbol{x\,y\,\bar{z}} + \bar{x}\,\bar{y}\,z + \bar{x}\,\bar{y}\,\bar{z} + z\,\bar{y}\,z \neq x\,y\,z + x\,\bar{y}\,z + \bar{x}\,\bar{y}\,z + \bar{x}\,\bar{y}\,\bar{z} + \boldsymbol{\bar{x}\,y\,z}$$

Esercizio 2

Con alcuni passaggi si vede che la funzione booleana realizzata dal circuito è la funzione costante 0 (nella terz' ultima espressione appaiono sia **y** che **not y**):

$$\overline{(x + (y\ xor\ \overline{(x\,y\,\bar{z})}) + \overline{(z\,\bar{y}(y+\bar{x}))})}$$

$$\overline{(x + (y\ xor\ \overline{(x\,y\,\bar{z})}) + (\bar{z} + y + \overline{(y+\bar{x})}))}$$

$$\overline{(x + (y\ xor\ \overline{(x\,y\,\bar{z})}) + (\bar{z} + y + \bar{y}\,x))}$$

$$\overline{(x + (y(x\,y\,\bar{z}) + \bar{y}\,\overline{(x\,y\,\bar{z})}) + \bar{z} + y + \bar{y}\,x)}$$

$$\overline{(x + (x\,y\,\bar{z} + \bar{y}(\bar{x} + \bar{y} + z)) + \bar{z} + y + \bar{y}\,x)}$$

$$\overline{(x + (x\,y\,\bar{z} + \bar{y}\,\bar{x} + \bar{y} + \bar{y}\,z) + \bar{z} + y + \bar{y}\,x)}$$

$$\overline{(x + x\,y\,\bar{z} + \bar{y}\,\bar{x} + \boldsymbol{\bar{y}} + \bar{y}\,z + \bar{z} + \boldsymbol{y} + \bar{y}\,x)}$$

$$\overline{(1)}$$

$$0$$

Per cui il circuito che la realizza non usa nessuna porta ed è la costante ' 0' .

Sono valide (anche se sono molto meno efficienti) anche soluzioni ottenute sostituendo le varie porte del circuito con circuiti che usano solo NAND:

- x + y con not(not x not y)
- x xor y con x not y + not x y = not(not(x not y) not(not x y))
- not x con not(x x)
- x nor y con not x not y

Esercizio 3

Il codice di Gray a 3 bit si ottiene facilmente partendo da 000 e cambiando un bit alla volta come per le intestazioni delle mappe di Karnaugh, quindi la sequenza è: 000, 001, 011, 010, 110, 111, 101, 100.

La tabella di verità della funzione Y è quindi quella qui a destra ->

Per realizzare il circuito con una ROM basta copiare le tre colonne y_2,y_1,y_0.

Per realizzarlo con una PLA bisogna costruire le mappe di Karnaugh che sono qua sotto.

Le funzioni minimizzate ottenute dalle mappe sono:

$$y_2 = x_2\bar{x}_1 + x_2\bar{x}_0 + \bar{x}_2 x_1 x_0$$
$$y_1 = x_2\bar{x}_0 + x_2 x_1 + \bar{x}_2\bar{x}_1 x_0$$
$$y_0 = x_1$$

Si noti che il termine $x_2\bar{x}_0$ è presente sia in y_2 che in y_1 e quindi la corrispondente colonna della PLA deve apparire una sola volta con due pallini OR.

Per realizzare il circuito con Multiplexer 4-a-1 bisogna scrivere le tabelle dei multiplexer che assomigliano a quelle delle mappe di Karnaugh tranne per l' ordine delle due ultime righe e colonne.

X	*x3*	*x2*	*x1*	*x0*	*Y*	*y2*	*y1*	*y 0*
0	0	0	0	0	0	0	0	0
1	0	0	0	1	3	0	1	0
2	0	0	1	0	6	1	0	1
3	0	0	1	1	1	0	0	1
4	0	1	0	0	4	1	1	0
5	0	1	0	1	7	1	0	0
6	0	1	1	0	2	0	1	1
7	0	1	1	1	5	1	1	1
-8	1	0	0	0	0	0	0	0
-7	1	0	0	1	3	0	1	0
-6	1	0	1	0	6	1	0	1
-5	1	0	1	1	1	0	0	1
-4	1	1	0	0	4	1	1	0
-3	1	1	0	1	7	1	0	0
-2	1	1	1	0	2	0	1	1
-1	1	1	1	1	5	1	1	1

y_2	*00*	*01*	*11*	*10*
00	0	0	0	1
01	1	1	1	0
11	1	1	1	0
10	0	0	0	1
	x_2	x_2	x_2	x_2n

y_1	*00*	*01*	*11*	*10*
00	0	1	0	0
01	1	0	1	1
11	1	0	1	1
10	0	1	0	0
	x_2	x_2n	x_2	x_2

y_0	*00*	*01*	*11*	*10*
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1
	0	0	1	1

Come fatto a lezione riuso le tabelle delle mappe di Karnaugh e scrivo sotto ogni colonna la funzione che corrisponde alla colonna.

Con **x_2n** indico x_2 negato.

ATTENZIONE: Nel disegnare i multiplexer con i corrispondenti input bisogna ricordarsi che le colonne della mappa di Karnaugh sono nell' ordine 1,2,4,3 e quindi l' ordine delle funzioni di input va opportunamente corretto.

Quindi gli input del MUX che produce y_2 sono, nell'ordine: **x_2, x_2, not x_2, x_2**.

Gli input del MUX che produce y_1 sono, nell' ordine **x_2, not x_2, x_2, x_2**.

Non è necessario un MUX per y_0 visto che y_0 è uguale a x_1

Esercizio 4

L' automa deve riconoscere le stringhe **0111, 1011, 1101, 1110, 1111**.

Quindi inizialmente costruiamo gli stati che dovranno rappresentare le sequenti sottostringhe:

vuoto, 0, 01, 011, 1, 10, 101, 11, 110 e 111

Le transizioni sono:

Da	*0*	*1*
vuoto	0/0	1/0
0	0/0	01/0
01	10/0	011/0
011	110/0	111/1
1	10/0	11/0
10	0/0	101/0
101	10/0	011/1
11	110/0	111/0
110	0/0	101/1
111	110/1	111/1

Transizioni

0	1-01								
01	0-10 1-011	0-10 01-011							
011	X	X	X						
1	0-10 1-11	0-10 01-11	011-11	X					
10	1-101	01-101	10-0 011-101	X	10-0 11-101				
101	X	X	X	110-10	X	X			
11	0-110 1-111	0-110 01-111	10-110 011-111	X	10-110 11-111	0-110 101-111	X		
110	X	X	X	110-0 111-101	X	X	10-0 011-101	X	
111	X	X	X	X	X	X	X	X	X
	vuoto	**0**	**01**	**011**	**1**	**10**	**101**	**11**	**110**

Tabella di esclusione

Si osserva che l' automa non è minimizzabile ulteriormente (le caselle grigie diventano X osservando le corrispondenti transizioni).

L' automa minimo (contrariamente a quanto scritto) necessita di 10 stati.

NOTA: per questo la correzione non terrà conto di semplici errori nel progetto dell' automa.

NOTA: nel caso della transizione **11->111** potrebbe essere corretto anche dare un output ' 1' (sebbene l' esercizio dica "negli ultimi 4 bit sono presenti 3 o più valori ' 1' "e in questo caso se ne sono letti solo 3). Se si cambia l' output quest transizione diventa identica alla transizione **011->111** e quindi l' automa è minimizzabile perchè i due stati **011** e **11** hanno identiche transizioni.

Stato	*Codifica*
vuoto	**1110**
0	**1100**
01	**1001**
011	**0011**
1	**1101**
10	**1010**
101	**0101**
11	**1011**

Per la costruzione della tabella di transizione dobbiamo scegliere una codifica per gli stati. Supponiamo di scegliere la seguente (in cui, per esempio, cerco di mantenere il codice e ci aggiungo dei bit a sinistra, in grassetto). Altre codifiche sono possibili, si è liberi di scegliere quella che si preferisce.

Nella tabella sono mostrate solo le 20 righe che corrispondono alle transizioni da ciascuno dei 10 stati quando l' input x vale 0 oppure 1. Tutte le altre righe sono don' t care.

Le colonne S ed R di ciascun flip-flop sono ottenute usando la tabella inversa dei flip-flop SR, che fornisce per ogni transizione y->Y da realizzare i corrispondenti input da fornire al flip-flop.

y	*Y*	*S*	*R*
0	0	0	-
0	1	1	0
1	0	0	1
1	1	-	0

Tabella inversa del flip-flop SR master-slave

Nella prima stesura ho scritto 0 invece di -.

x	y_3	y_2	y_1	y_0	Y_3	Y_2	Y_1	Y_0	*z*	S_3	R_3	S_2	R_2	S_1	R_1	S_0	R_0
0	0	0	1	1	0	1	1	0	0	0	-	1	0	-	0	0	1
0	0	1	0	1	1	0	1	0	0	1	0	0	1	1	0	0	1
0	0	1	1	0	1	1	0	0	0	1	0	-	0	0	1	0	-
0	0	1	1	1	0	1	1	0	1	0	-	-	0	-	0	0	1
0	1	0	0	1	1	0	1	0	0	-	0	0	-	1	0	0	1
0	1	0	1	0	1	1	0	0	0	-	0	1	0	0	1	0	-
0	1	0	1	1	0	1	1	0	0	0	1	1	0	-	0	0	1
0	1	1	0	0	1	1	0	0	0	-	0	-	0	0	-	0	-
0	1	1	0	1	1	0	1	0	0	-	0	0	1	1	0	0	1
0	1	1	1	0	1	1	0	0	0	-	0	-	0	0	1	0	-
1	0	0	1	1	0	1	1	1	1	0	-	1	0	-	0	-	0
1	0	1	0	1	0	0	1	1	1	0	-	0	1	1	0	-	0
1	0	1	1	0	0	1	0	1	1	0	-	-	0	0	1	1	0
1	0	1	1	1	0	1	1	1	1	0	-	-	0	-	0	-	0
1	1	0	0	1	0	0	1	1	0	0	1	0	-	1	0	-	0
1	1	0	1	0	0	1	0	1	0	0	1	1	0	0	1	1	0
1	1	0	1	1	0	1	1	1	0	0	1	1	0	-	0	-	0
1	1	1	0	0	1	0	0	1	0	-	0	0	1	0	-	1	0
1	1	1	0	1	1	0	1	1	0	-	0	0	1	1	0	-	0
1	1	1	1	0	1	1	0	1	0	-	0	-	0	0	1	1	0

Tabella delle transizioni (a sinistra)
e delle funzioni di eccitazione dei flip-flop(a destra)

A questo punto, per progettare il circuito usando multiplexer 4-a-1 costruiamo le tabelle delle funzioni z, S_3, R_3, ... , S_0, R_0 al variare sia di x che delle y_i:

x=0					***z***	***x=1***				
	00	01	10	11			00	01	10	11
00	-	-	-			00	-	-	-	1
01	-			1		01	-	1	1	1
10	-					10	-			
11				-		11				-
	0	0	0	y_2			0	ny_3	ny_3	ny_3
F						G				

Possiamo usare 3 MUX che hanno input, nell' ordine:

1. 0,0,0,y_2, e produce F
2. 0, not y_3, not y_3, not y_3 e produce G
3. F, G, 0, 0 e produce z

I primi due usano y_1,y_0 come selettori, il terzo usa 0,x per selezionare F o G.

x=0					***S_3***	***x=1***				
	00	01	10	11			00	01	10	11
00	-	-	-			00	-	-	-	
01	-	1	1			01	-			
10	-	-	-			10	-			
11	-	-	-	-		11	-	-	-	-
	0	1	1	0			0	0	0	0
F						G=0				

Possiamo usare due MUX con input, nell' ordine:

1. 0, 1, 1, 0 che produce F
2. F, 0, 0, 0 che produce S_3

Il primo usa y_1,y_0 come selettori, il secondo usa 0,x per selezionare F quando x=0

x=0					*R₃*	*x=1*				
	00	01	10	11			00	01	10	11
00	-	-	-	-		00	-	-	-	-
01	-			-		01	-	-	-	-
10	-			1		10	-	1	1	1
11				-		11				-
	0	0	0	1			0	ny_2	ny_2	1

F G

x=0					*S₂*	*x=1*				
	00	01	10	11			00	01	10	11
00	-	-	-	1		00	-	-	-	1
01	-		-	-		01	-		-	-
10	-		1	1		10	-		1	1
11	-		-	-		11			-	-
	0	0	1	1			0	0	1	1

$F=y_1$ $G=F=y_1$

Possiamo usare 3 MUX:

1. 0,0,0,1 che produce F
2. 0, not y_2, not y_2, 1 che produce G
3. F, G, 0, 0 che produce R_3,

I primi due usano y_1,y_0, come selettori, il terzo usa 0,x per selezionare F o G a seconda del valore di x.

In questo caso la funzione non dipende da x, ed anzi corrisponde alla funzione

$S_2=y_1$

x=0					*R₂*	*x=1*				
	00	01	10	11			00	01	10	11
00	-	-	-			00	-	-	-	
01	-	1				01	-	1		
10	-	-				10	-	-		
11		1		-		11	1	1		-
	0	1	0	0			1	1	0	0

F $G=y_1$

Possiamo usare 2 MUX con input:

1. 0, 1, 0, 0 che produce F
2. F, y_1, 0, 0 che produce R_2

Il primo usa y_1,y_0, come selettori, il secondo usa 0,x per selezionare F o G a seconda del valore di x.

x=0					S_1	*x=1*				
	00	01	10	11			00	01	10	11
00	-	-	-	-		00	-	-	-	-
01	-	1		-		01	-	1		-
10	-	1		-		10	-	1		-
11		1		-		11		1		-
	0	1	0	1			0	1	0	1
		$F=y_0$								$G=F=y_0$

In questo caso la funzione non dipende da x, ed anzi sorrisponde alla funzione

$S_1=y_0$.

x=0					R_1	*x=1*				
	00	01	10	11			00	01	10	11
00	-	-	-			00	-	-	-	
01	-		1			01	-		1	
10	-		1			10	-		1	
11	-		1	-		11	-		1	-
	0	0	1	0			0	0	1	0
		F								G=F

In questo caso la funzione non dipende da x, quindi basta un solo MUX con input:

0, 0, 1, 0,

che produce R_1, e usa come selettori y_1,y_0,

x=0					S_0	*x=1*				
	00	01	10	11			00	01	10	11
00	-	-	-			00	-	-	-	-
01	-					01	-	1	-	-
10	-					10	-	1	-	-
11				-		11	1	1	-	-
	0	0	0	0			1	1	1	1
		F=0								G=1

In questo caso la funzione è

S_0=x

x=0					***R_0***	***x=1***				
	00	01	10	11			00	01	10	11
00	-	-	-	1		00	-	-	-	
01	-	1	-	1		01	-			
10	-	1	-	1		10	-			
11	-	1	-	-		11				-
	1	1	1	1			0	0	0	0

F=1 G=0

In questo caso R_0=not x. Non c'è bisogno di MUX.