Architettura degli Elaboratori
Esercizi

Ugo Dal Lago
dallago@cs.unibo.it

29 novembre 2007

1 Introduzione al Corso

Esercizio 1.
Supponiamo che M1 sia una macchina virtuale e che L1 sia il relativo linguaggio macchina. Supponiamo che esistano un traduttore T da L1 a L0 e un interprete N di L1 (entrambi scritti in L0). Supponiamo che T impieghi 5 nanosecondi per tradurre ogni istruzione di L1 in una sequenza di 4 istruzioni di L0. Supponiamo altresì che N impieghi 6 nanosecondi per ogni istruzione e che ogni istruzione di L0 possa essere eseguita in 3 nanosecondi. Per quali tra i programmi di L1, in una situazione come quella appena delineata, l'interpretazione è più efficiente della compilazione?
Soluzione.
L'interpretazione è sempre più efficiente della compilazione. Sia P un generico programma del livello L1 e sia l il numero di istruzioni di P. Il tempo necessario a tradurre P e ad eseguire il programma tradotto è

$$5 \cdot l + 4 \cdot 3 \cdot l = 17 \cdot l$$

nanosecondi. Il tempo necessario ad interpretare P è invece pari a soli $6 \cdot l$ nanosecondi. Chiaramente, $17 \cdot l > 6 \cdot l$ per tutti i valori positivi di l.

Esercizio 2.
Supponiamo che esistano un traduttore (scritto in L0) del linguaggio L1 nel linguaggio L0 e un interprete di L1 scritto nel linguaggio di L0. Supponiamo che il traduttore impieghi k nanosecondi per ogni istruzione di L1, producendo in output m istruzioni di L0 per ogni singola istruzione di L1. Supponiamo, invece, che l'interprete impieghi n nanosecondi per eseguire una generica istruzione. Infine, supponiamo che la macchina M0 impieghi j nanosecondi per eseguire ogni istruzione di M0. Quando, in una situazione come quella appena delineata, l'interpretazione è più efficiente della traduzione?

2 Organizzazione dei Sistemi di Calcolo

Esercizio 1.
Si consideri il codice C in cui le uniche parole di codice valide sono 000, 011 e 110. Nel codice C, quali errori possono essere rilevati e quali errori possono essere corretti? Dove ha sbagliato il progettista del codice C?
Soluzione.
Il codice C ha distanza di Hamming pari a 2 (la distanza di Hamming tra qualunque coppia di parole di codice è 2). Di conseguenza, possono essere rilevati solo errori singoli e nessun errore può essere corretto. Il progettista del codice C poteva aggiungere un'altra parola al codice C, ovvero 101 senza modificare la relativa distanza di Hamming. In questo modo, avrebbe ottenuto il codice con bit di parità in cui i primi due bit sono riservati ai dati e l'ultimo è il bit di controllo.

Esercizio 2.
Si costruisca un codice correttore di errore in modo tale che il codice comprenda almeno 8 parole di codice e che la distanza di Hamming tra esse sia almeno 2. Nel definire il codice cerchi di minimizzare la lunghezza delle parole di codice.
Soluzione.
Una possibilità consiste nel considerare 3 *bit per i dati (visto che* $2^3 = 8$*) e un solo bit di controllo, ossia un bit di parità. Le* 8 *parole di codice saranno le seguenti:*

000 0
001 1
010 1
011 0
100 1
101 0
110 0
111 1

Esercizio 3.
Una memoria contiene 32 celle, contenenti 8 bit ciascuna. Quanti indirizzi occorreranno per riferirsi a tutti i dati contenuti in memoria? Supponiamo di scrivere gli indirizzi come sequenze di bit. Quanti bit sono necessari?

Esercizio 4.
Supponiamo di voler confrontare due sistemi di calcolo: il sistema M non dispone di una cache ma dispone di una memoria relativamente veloce, mentre il sistema N è un sistema con cache. Per il sistema M, il tempo di accesso in memoria è m, mentre nel sistema N, l'hit ratio è h, il tempo di accesso alla cache è $\frac{m}{3}$ e il tempo di accesso alla memoria centrale è $2m$. Quale è il minimo valore di h per il quale N diventa più efficiente di M?

Esercizio 5.
Si determini la capacità di un disco con 8 superfici, 16 tracce e 8 settori per traccia, supponendo che ogni settore contenga 512 byte.

3 Rappresentazione dell'Informazione

Esercizio 1.
Convertire i seguenti numeri dalla base di partenza nelle altre tre basi elencate nella seguente tabella:

Decimale	Binaria	Ottale	Esadecimale
225	?	?	?
?	11010111	?	?
?	?	623	?
?	?	?	2AC5

Soluzione.

$$
(225)_{10} = \begin{array}{l|l l}
 & & \textit{Resti} \\
225 & 2 & \boxed{\mathit{1}} \\
112 & 2 & \boxed{\mathit{0}} \\
56 & 2 & \boxed{\mathit{0}}\uparrow \\
28 & 2 & \boxed{\mathit{0}} \\
14 & 2 & \boxed{\mathit{0}} \\
7 & 2 & \boxed{\mathit{1}} \\
3 & 2 & \boxed{\mathit{1}}\uparrow \\
\boxed{\mathit{1}} & \rightarrow &
\end{array}
= \begin{array}{c}
(E \qquad 1)_{16} \\
(\overbrace{1\,1\,1\,0}\,\overbrace{0\,0\,0\,1})_2 \\
(3 \quad 4 \quad 1)_8
\end{array}
$$

$$
\begin{array}{c}
(D \quad 7)_{16} \\
(\overbrace{1\,1\,0\,1}\,\overbrace{0\,1\,1\,1})_2 \\
(3 \quad 2 \quad 7)_8
\end{array}
= 2^7 + 2^6 + 2^4 + 2^2 + 2^1 + 2^0 = 128 + 64 + 16 + 4 + 2 + 1 = (215)_{10}
$$

$$
(6 \quad 2 \quad 3)_8 = 6 * 8^2 + 2 * 8^1 + 3 * 8^0 = 384 + 16 + 3 = (403)_{10}
$$

$$
\swarrow \quad \searrow \quad \searrow
$$

$$
(\underbrace{1}_{(1}\,\underbrace{10\ \ 01}_{9}\underbrace{0\ \ 011}_{3)_{16}})_2
$$

$$
(2 \quad A \quad C \quad 5)_{16} = 2 * 16^3 + 10 * 16^2 + 12 * 16^1 + 5 * 16^0 = 8192 + 2560 + 192 + 5 = (10949)_{10}
$$

$$
\swarrow \downarrow \quad \searrow \searrow
$$

$$
(0\,\underbrace{010}_{(2}\,\underbrace{101}_{5}\,\underbrace{0\ 11}_{3}\underbrace{00\ 0}_{0}\,\underbrace{101}_{5)_8})_2
$$

Esercizio 2.
Sia $D = \{cane, gatto, oca, gallina, cavallo, topo, coniglio, mucca\}$. Si consideri D come un insieme di dati rappresentabili e sia A l'alfabeto finito $\{0, 1\}$. Si definisca una funzione di codifica f di D in A^*. In particolare. si definisca f in modo tale che le codifiche degli elementi di D siano parole della medesima lunghezza l e che l sia più piccolo possibile.
Soluzione.
Una possibile definizione per f potrebbe essere la seguente:

$$
\begin{array}{rcl rcl}
f(cane) & = & 000 & f(gatto) & = & 001 \\
f(oca) & = & 010 & f(gallina) & = & 011 \\
f(cavallo) & = & 100 & f(topo) & = & 101 \\
f(coniglio) & = & 110 & f(mucca) & = & 111
\end{array}
$$

Esercizio 3.
Convertire i seguenti numeri binari (senza segno) in decimale:

11001
101110
1110101.11
110110100
111010110110

Esercizio 4.
Convertire i seguenti numeri decimali nelle basi indicate.

7562 (in ottale)
1938 (in esadecimale)
16.6875 (in binario)

Esercizio 5.
Convertire i seguenti numeri esadecimale in binario e in ottale.

$$E45A$$
$$F3A7C2$$
$$200E67A$$

Esercizio 6.
Sommare e moltiplicare le seguenti coppie di numeri (senza segno) senza convertirli in decimale.

$$\begin{array}{rl}
(110110)_2 & (110101)_2 \\
(1001)_2 & (11001101)_2 \\
(15F)_{16} & (A7)_{16} \\
(AEC061)_{16} & (B9714)_{16} \\
(15F)_{16} & (A7)_{16} \\
(367)_8 & (715)_8
\end{array}$$

Esercizio 7.
Calcolare il complemento a 2 dei seguenti numeri binari (senza segno).

$$10101110$$
$$10000000$$
$$00000000$$

Esercizio 8.
Risolvere le seguenti sottrazioni tra coppie di numeri binari (senza segno) calcolando il complemento a 2 del sottraendo.

$$11010 - 10000$$
$$100 - 110000$$

Esercizio 9.
I numeri seguenti sono rappresentati in complemento a 2. Eseguire le seguente operazioni aritmetiche e verificare le soluzioni.

$$101011 + 111000$$
$$001110 + 110010$$
$$111001 - 001010$$
$$101011 - 100110$$

Esercizio 10.
Indicare per quale delle seguenti somme di numeri binari in complemento a 2 si verifica un overflow.

$$000 + 111$$
$$111 + 110$$
$$100 + 111$$

4 Reti Combinatorie e Algebra di Boole

Esercizio 1.
Data la funzione booleana:

$$F(W,X,Y,Z) = X\overline{Y}Z + \overline{X}\,\overline{Y}Z + \overline{W}XY + W\overline{X}Y + WXY$$

(a) Costriuire la tabella di verità della funzione F.
(b) Semplificare la funzione F a un'espressione contenente essattemente 5 letterali usando l'algebra booleana. Si ricordi che una letterale è una singola variabile sia in forma diretta che negata. Per esempio, X e $\overline{X}$ sono 2 letterali diversi.
(c) Costruire la teballa di verità della funzione F ottenuta dalla semplificazione nella parte (b) e dimostrare che è la stessa della tabella costruita nella parte (a).

Soluzione.

$$\begin{array}{lrcl} & F(W,X,Y,Z) & = & X\overline{Y}Z + \overline{X}\,\overline{Y}Z + \overline{W}XY + W\overline{X}Y + WXY \\ (b) & & = & \overline{Y}Z + +\overline{W}XY + W\overline{X}Y + WXY \\ & & = & \overline{Y}Z + XY + WY = \overline{Y}Z + Y(X+W) = G \end{array}$$

(a)(c) Le funzioni F e G hanno la stessa tabella di verità.

Ingressi									*Uscita*				*Uscita*
W	X	Y	Z	$X\overline{Y}Z$	$\overline{X}\,\overline{Y}Z$	$\overline{W}XY$	$W\overline{X}Y$	WXY	F	$\overline{Y}Z$	XY	WY	G
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	1	1	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	1	1	0	0	1
0	1	1	0	0	0	1	0	0	1	0	1	0	1
0	1	1	1	0	0	1	0	0	1	0	1	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	1	1	0	0	1
1	0	1	0	0	0	0	1	0	1	0	0	1	1
1	0	1	1	0	0	0	1	0	1	0	0	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	1	1	0	0	1
1	1	1	0	0	0	0	0	1	1	0	1	1	1
1	1	1	1	0	0	0	0	1	1	0	1	1	1

Esercizio 2.
Semplificare l'espressione seguente usando l'algebra booleana in modo tale che possa essere implementata usando solo le porte XOR e AND:

$$A\overline{B}C\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + \overline{A}B\overline{C}D$$

Soluzione.

$$\begin{array}{rl} & A\overline{B}C\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + \overline{A}B\overline{C}D \\ = & C\overline{D}(A\overline{B} + \overline{A}B) + \overline{C}D(A\overline{B} + \overline{A}B) \\ = & C\overline{D}(A \oplus B) + \overline{C}D(A \oplus B) \\ = & (A \oplus B)(C\overline{D} + \overline{C}D) \\ = & (A \oplus B)(C \oplus D) \end{array}$$

Esercizio 3.
L'operatore INHIBITION si indica con il simbolo / ed è definito dalla seguente operazione logica: $X/Y = X\bar{Y}$. Utilizzando le tabelle di verità, provare che:
(a) l'operatore INHIBITION non è né commutativo né associativo.
(b) l'operatore XOR è sia commutativo che associativo.

Soluzione.

X	Y	Z	X/Y	Y/X	$(X/Y)/Z$	Y/Z	$X/(Y/Z)$	$Y \oplus Z$	$X \oplus (Y \oplus Z)$	$X \oplus Y$	$Y \oplus X$	$(X \oplus Y) \oplus Z$
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1	0	0	1
0	1	0	0	1	0	1	0	1	1	1	1	1
0	1	1	0	1	0	0	0	0	0	1	1	0
1	0	0	1	0	1	0	1	0	1	1	1	1
1	0	1	1	0	0	0	1	1	0	1	1	0
1	1	0	0	0	0	1	0	1	0	0	0	0
1	1	1	0	0	0	0	1	0	1	0	0	1

(a) *Siccome $X/Y \neq Y/X$ e $(X/Y)/Z \neq X/(Y/Z)$ nella tabella, l'operatore INHIBITION non è né commutativo né associativo.*
(b) *Siccome $X \oplus Y = Y \oplus X$ e $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$ nella tabella, l'operatore XOR è sia commutativo sia associativo.*

Esercizio 4.
Provare la validità delle seguenti identità usando l'algebra booleana.
(a) $XZ + \bar{X}Y = (X+Y)(\bar{X}+Z)$
(b) $XY + Z\bar{X} + ZY = XY + \bar{X}Z$

Soluzione.
(a)

$$\begin{aligned} XZ + \bar{X}Y &= XZ(1+Y) + \bar{X}Y(1+Z) \\ &= XZ + XZY + \bar{X}Y + \bar{X}YZ \\ &= XZ + YZ(X+\bar{X}) + \bar{X}Y \\ &= XZ + YZ + \bar{X}Y \\ &= XZ + YZ + \bar{X}Y + X\bar{X} \\ &= Z(X+Y) + \bar{X}(X+Y) \\ &= (X+Y) + (\bar{X}+Z) \end{aligned}$$

(b)

$$\begin{aligned} XY + Z\bar{X} + ZY &= XY + Z\bar{X} + ZY(X+\bar{X}) \\ &= XY + \bar{X}Z + XYZ + \bar{X}YZ \\ &= XY(1+Z) + \bar{X}Z(1+Y) \\ &= XY + \bar{X}Z \end{aligned}$$

Esercizio 5.
Ridurre le seguenti espressioni booleane al numero di letterali indicato.

$$\begin{array}{rl} \bar{A}\bar{C} + ABC + A\bar{C} & \text{(tre letterali)} \\ \overline{(\bar{A}\bar{B}+C)} + C + AB + DC & \text{(tre letterali)} \\ \bar{A}B(\bar{D}+\bar{C}D) + B(A+\bar{A}CD) & \text{(un letterale)} \\ (\bar{A}+C)(\bar{A}+\bar{C})(A+B+\bar{C}D) & \text{(quattro letterali)} \end{array}$$

Esercizio 6.
Utilizzando le leggi di DeMorgan, esprimere le espressioni seguenti soltanto con gli operatori OR e NOT.

$$\begin{gathered} \bar{A}\bar{B} + \bar{A}C + \bar{B}C \\ ABC + A\bar{C} + \overline{AB}(B+C) \end{gathered}$$

Esercizio 7.
Utilizzando le leggi di DeMorgan, esprimere le espressioni dell'esercizio precedente utilizzando gli operatori AND e NOT.

Esercizio 8.
Dimostrare, usando tabelle di verità, la validità delle seguenti identità:

- La seguente legge di DeMorgan per tre variabili:

$$\overline{ABC} = \bar{A} + \bar{B} + \bar{C}$$

- La seconda legge distributiva

$$A + BC = (A + B)(A + C)$$

- Il teorema del consenso:

$$AB + \bar{A}C + BC = AB + \bar{A}C$$

Esercizio 9.
Disegnare il diagramma logico per le seguenti espressioni booleane. Il diagramma deve corrispondere esattamente all'equazione (assumendo che i complementi degli ingressi *non* siano disponibili).

$$B\bar{C} + AB + ACD$$
$$(A + B)(C + D)(\bar{A} + B + D)$$
$$(AB + \bar{A}\bar{B})(C\bar{D} + \bar{C}D)$$

5 Sintesi di Reti Combinatorie

Esercizio 1.
Costruire la tabella di verità per le seguenti funzioni ed esprimere ciascuna funzione in forma di somma di mintermini e prodotto di maxtermini.

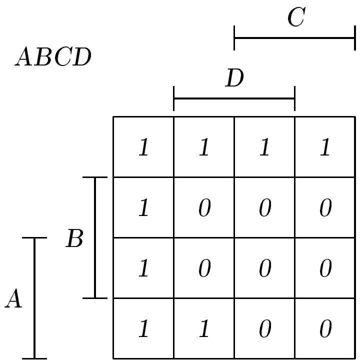
$$(AB + C)(B + AC)$$
$$(\bar{A} + B)(\bar{B} + C)$$
$$\bar{B}C + DA\bar{B} + DA\bar{C} + \bar{D}\bar{A}C$$

Esercizio 2.
Considerare la funzione:

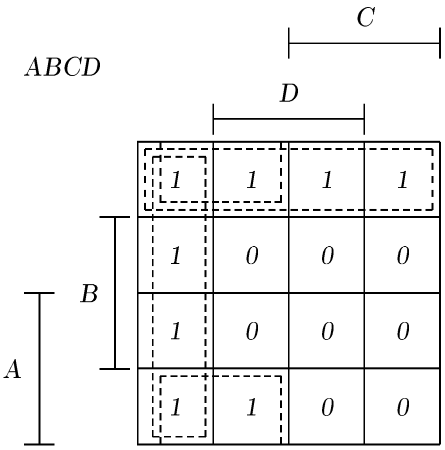
$$F(A, B, C, D) = \sum m(0, 1, 2, 3, 4, 8, 9, 12)$$

(a) Rappresentare F nella mappa di Karnaugh a 4 variabili.
(b) Semplificare F in forma di somma di prodotti utilizzando la mappa.
(c) Disegnare il diagramma logico per F usando soltanto porte NAND. Assumere che le variabile d'ingresso siano disponibili sia in forma normale che complementata.
(d) Semplificare F in forma di prodotto di somme utilizzando la mappa.
(e) Disegnare il diagramma logico per F usando soltanto porte NOR. Assumere che le variabile d'ingresso siano disponibili sia in forma normale che complementata.

Soluzione.
(a) La mappa di Karnaugh che otteniamo è la seguente:

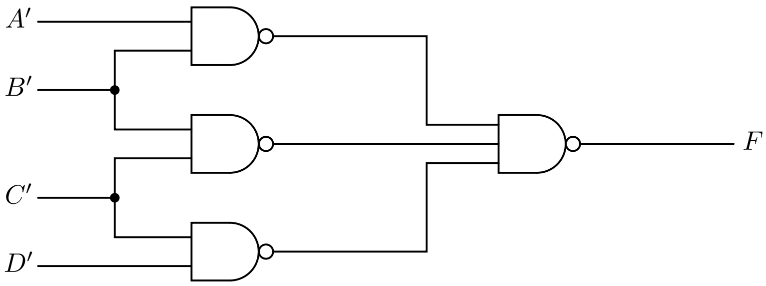


(b) *Dalla copertura seguente*

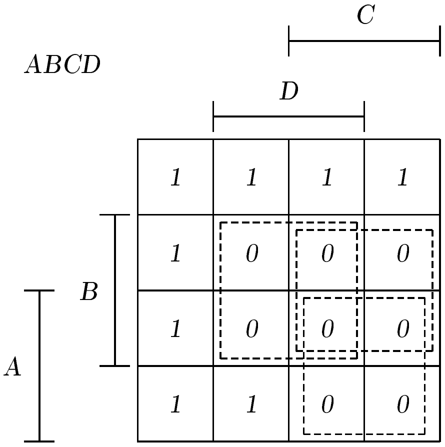


si ottiene l'espressione $F = \overline{A}\ \overline{B} + \overline{C}\ \overline{D} + \overline{B}\ \overline{C}$.

(c) *La rete ottenuta è la seguente:*

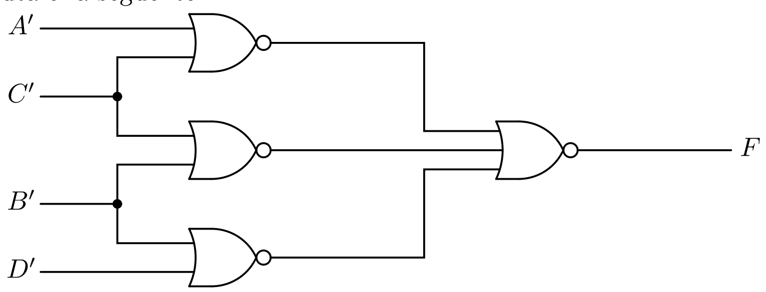


(d) *Dalla copertura seguente*



si ottiene l'espressione $\overline{F} = BD + BC + AC$, $\overline{\overline{F}} = F = (\overline{B} + \overline{D})(\overline{B} + \overline{C})(\overline{A} + \overline{C})$

(e) *La rete ottenuta è la seguente:*



Esercizio 3.
Semplificare le seguenti funzioni booleane utilizzando le mappe di Karnaugh.

$$F(A,B,C) = \sum m(0,1,5,7)$$
$$F(A,B,C) = \sum m(1,2,3,6,7)$$
$$AB + \bar{B}\bar{C} + \bar{A}B\bar{C}$$

Esercizio 4.
Semplificare le seguenti funzioni booleane utilizzando le mappe di Karnaugh.

$$F(A,B,C,D) = \sum m(0,1,2,4,5,7,11,15)$$
$$F(A,B,C,D) = \sum m(1,6,10,11,14,16)$$
$$\bar{D}C + AC + \bar{A}B + D\bar{A}C$$

Esercizio 5.
Semplificare in forma di prodotto di somme le seguenti funzioni booleane utilizzando le mappe di Karnaugh.

$$F(A,B,C,D) = \sum m(0,2,5,6,7,8,10)$$
$$F(A,B,C,D) = \prod M(1,3,5,7,13,15)$$

Esercizio 4.
Semplificare le seguenti funzioni booleane non completamente specificate, utilizzando la condizione di indifferenza d.

$$F(A,B,C,D) = \sum m(0,1,2,3,7,8,10) \quad d(A,B,C,D) = \sum m(5,6,11,15)$$
$$F(A,B,C,D) = \sum m(3,4,13,15) \quad d(A,B,C,D) = \sum m(1,2,5,6,8,10,12,14)$$

Esercizio 6.
Semplificare le seguenti funzioni booleane trovando i primi implicanti essenziali.

$$F(A,B,C,D) = \sum m(0,2,4,5,6,7,8,10,13,15)$$
$$F(A,B,C,D) = \sum m(0,2,3,5,7,8,10,11,14,15)$$

Esercizio 7.
Si vuole progettare una rete combinatoria che funga da controllore per un sistema di allarme. Tre ingressi provengono da altrettanti sensori, che rilevano la presenza di estranei in tre siti distinti. Esiste poi un quarto ingresso, proveniente da un interruttore, tramite il quale si controlla l'attivazione dell'intero sistema di allarme. Due uscite sono collegate ad altrettanti dispositivi che producono un segnale sonoro.

- Il primo dispositivo deve emettere un suono se in almeno due siti è presente un estraneo.
- Il secondo dispositivo deve emettere un suono se in almeno un sito è presente un estraneo.

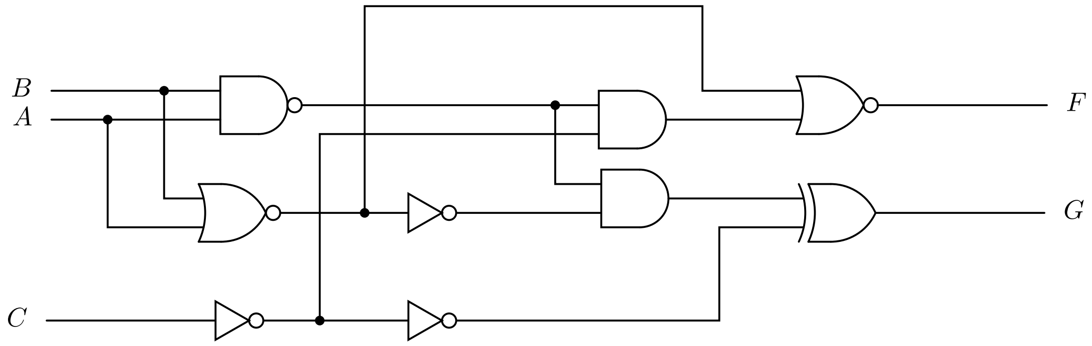
Si costruisca, utilizzando le mappe di Karnaugh, una rete combinatoria che soddisfi le specifiche di cui sopra.

Esercizio 8.
Si vuole progettare una rete combinatoria che funga da controllore per i sistemi di riscaldamento e refrigerazione di una cella frigorifera. Due ingressi sono collegati ad un sensore che rileva la temperatura interna alla cella (per semplicità si supponga che le temperature possibili siano solo quattro, ossia le temperature comprese tra 0 gradi centigradi e 3 gradi centigradi). Altri due ingressi sono collegati ad un manopola tramite la quale è possibile selezionare la temperatura attesa. Le due uscite sono collegate ad un dispositivo di refrigerazione e ad un dispositivo di riscaldamento. Si costruisca, utilizzando le mappe di Karnaugh, una rete combinatoria che soddisfi le specifiche di cui sopra.

Esercizio 9.
Si vuole progettare una rete combinatoria che funga da controllore per una piscina. Quattro ingressi rilevano la presenza di acqua a tre livelli distinti, all'interno della piscina, ossia a 20, 50 e 100 centimetri dal fondo. Un quarto ingresso è collegato ad un interrutore tramite il quale selezionare una tra le modalità di funzionamento "acqua alta" e "acqua bassa". Nella modalità "acqua alta" il livello dell'acqua deve essere compreso tra 50 e 100 cm, mentre in modalità acqua bassa, il livello dell'acqua deve essere compreso tra 20 e 50 centimetri. Un'uscita è collegata ad un rubinetto, che viene aperto quando il segnale vale 1 e chiuso quando il segnale vale 0. Una seconda uscita è collegata ad uno scolo, che risulta aperto quando il segnale vale 1 e chiuso quando il segnale vale 0. Si costruisca, utilizzando le mappe di Karnaugh, una rete combinatoria che soddisfi le specifiche di cui sopra.

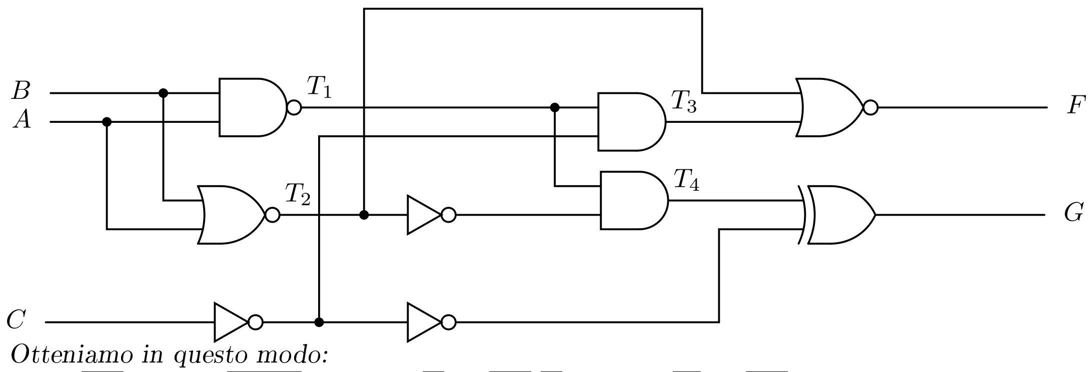
6 Complementi sulle Reti Combinatorie

Esercizio 1.
Analizzare il circuito seguente come segue:



(a) Determinare le espressioni booleane per le uscite F e G in funzioni dei tre ingressi A, B, e C.
(b) Verificare che tale circuito implementi effettivamente un full adder.

Soluzione.
(a) Etichettiamo la rete in questione con T_1, T_2, T_3 e T_4:



Otteniamo in questo modo:
$T_1 = (\overline{AB}),\ T_2 = (\overline{A+B}),\ T_3 = T_1\overline{C} = (\overline{AB})\overline{C},\ T_4 = T_1\overline{T_2} = (\overline{AB})(A+B)$

$F = \overline{(T_1 + T_3)} = (\overline{(\overline{A+B}) + (\overline{AB})\ \overline{C}}) = (A+B)(AB+C) =$
$(A+B)(A+C)(B+C) = AB + AC + BC$
$G = C \oplus T_4 = C \oplus [(\overline{AB})(A+B)] = C \oplus [(\overline{A}+\overline{B})(A+B)] = C \oplus (A \oplus B) = A \oplus B \oplus C$

(b) *Un full-adder ha 3 variabili in ingresso (A, B e C), e 2 variabili in uscita (S e R). Mella tabella seguente, le funzioni F e G corrispondono alle uscite R e S di un full adder rispettivamente.*

Ingressi			*Uscite*		*Uscite*	
A	*B*	*C*	*F*	*G*	*R*	*S*
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	1	0	1	0
1	1	0	1	0	1	0
1	1	1	1	1	1	1

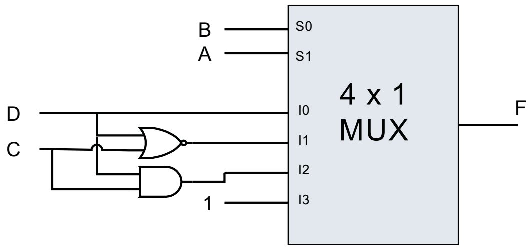
Esercizio 2.
Implementare la funzione seguente con un multiplexer 4-1 e alcune porte logiche esterne:

$$F(A, B, C, D) = \sum m(1, 3, 4, 11, 12, 13, 14, 15)$$

Si noti che un multiplexer 4-1 ha 2 ingressi di selezione (S_1, S_0) e 4 ingressi per i dati (I_0, I_1, I_2, I_3). In aggiunta al multiplexer 4-1 e ai segnali delle variabili A, B, C, D, sono disponibili **solo** le porte logiche AND, NAND, OR, NOR, e i segnali 0 e 1.

Soluzione.
Applicando le prime 2 variabili A e B alle linee di selezione S_1 e S_0, la funzione F può essere definita in termini delle variabili C e D.

A	B	C	D	F	
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	
0	1	0	1	0	
0	1	1	0	0	$F = \overline{(C + D)}$
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	$F = CD$
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



Esercizio 3.
Progettare un moltiplicatore binario per la moltiplicazione di due numeri binari senzo segno a 2 bit, utilizzando solo le porte logiche AND, un sommatore binario parallelo a 2 bit, e i segnali 0 e 1. Indicare i bit del moltiplicando come b_1b_0, i bit del moltiplicatore come a_1a_0, e i bit del prodotto come $s_3s_2s_1s_0$. Si noti che un sommatore binario parallelo a n bit ha $2n$ variabili di ingresso e $n + 1$ variabili di uscita.

Esercizio 4.
Un circuito combinatorio è definito dalle due seguenti funzioni booleane:

$$F_1(A, B, C) = \prod M(1, 2, 4, 7)$$
$$F_2(A, B, C) = A \oplus B \oplus C$$

Implementare il circuito utilizzando un decodificatore e porte logiche OR.

Esercizio 5.
Un decodificatore con l'ingresso di abilitazione E (enable) è attivato quando $E = 1$, nel qual caso opera normalmente. Tale decodificatore è disattivato quando $E = 0$, nel qual caso genera 0 su tutte le uscite, qualunque sia il valore applicato agli ingressi. Realizzare un decodificatore $3 - 8$ usando due decodificatori $2 - 4$ con enable.

Esercizio 6.
Realizzare un circuito per la sottrazione tra 2 numeri binari senza segno utilizzando un sommatore binario a 4 bit.

Esercizio 7.

Si considerino le tre funzioni booleane seguenti:

$$\begin{aligned} F_1(A,B,C) &= \bar{A} + \bar{B}\bar{C} + AC \\ F_2(A,B,C) &= A + \bar{B}\bar{C} + \bar{A}B \\ F_3(A,B,C) &= \bar{A} + \bar{B}C + AB \end{aligned}$$

Si costruisca una rete combinatoria con tre entrate A, B e C e tre uscite F_1, F_2 e F_3 che corrisponda alle tre funzioni booleane di cui sopra. Nella realizzazione della rete combinatoria si utilizzino un decodificatore e porte esterne.

Esercizio 8.
Implementare la funzione booleana seguente attraverso un multiplexer con 8 ingressi per i dati e una singola porta NOT.

$$F(A,B,C,D) = \sum m(1,3,4,11,12,13,14,15)$$

Esercizio 9.
Implementare con un PLA le funzioni booleane seguenti.

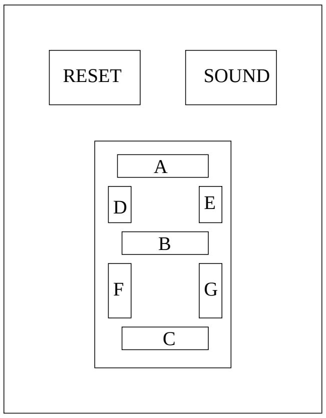
$$\begin{aligned} F_1(A,B,C) &= A + \bar{B} + AC + \bar{A}B\bar{C} \\ F_2(A,B,C) &= \overline{AC + BC} \end{aligned}$$

7 Sintesi di Reti Sequenziali

Esercizio 1.
Si vuole progettare una rete sequenziale sincrona che funga da controllore per un sistema di allarme. Un segnale binario proviene da un sensore, che rileva la presenza di estranei. L'unica uscita è collegata ad un dispositivo sonoro, che deve essere attivato in un generico istante t se e solo se in almeno tre dei quattro fronti di salita del clock precedenti a t è stata rilevata la presenza di un estraneo. Si costruisca una rete sequenziale sincrona (contenente solamente flip-flop di tipo D) che soddisfi le specifiche di cui sopra.

Esercizio 2.
Si vuole progettare una rete sequenziale sincrona che funga da riconoscitore di sequenza. Il valore binario in uscita deve valere 1 se e solo se i quattro valori binari letti sull'unica entrata nei quattro fronti di salita precedenti sono esattamente 1, 0, 1 e 0, in quest'ordine. Si costruisca una rete sequenziale sincrona (contenente solamente flip-flop di tipo JK) che soddisfi le specifiche di cui sopra.

Esercizio 3.
Si progetti una rete sequenziale che controlli un piccolo gadget elettronico. Il gadget ha l'aspetto sequente:



Notiamo che il gadget comprende tre dispositivi:

- Un display contenente sette rettangoli. I sette rettangoli possono essere illuminati o meno. Ciascuno di essi corrisponde ad un'uscita del circuito sequenziale che si vuole progettare. Le sette uscite saranno quindi A, B, C, D, E, F, G. Ciascuna di esse sarà valorizzata a 0 o 1 a seconda che si voglia o meno che il corrispondente rettangolo sia illuminato.
- Un pulsante, che corrisponde all'ingresso $RESET$ del circuito sequenziale. Se il pulsante è premuto, $RESET$ vale 1, altrimenti $RESET$ vale 0.
- Un piccolo altoparlante, che può emettere un suono. L'altoparlante è comandato dall'uscita $SOUND$, che vale 1 se e solo se occorre emettere un suono.

Vogliamo che il gadget funga da conto alla rovescia. Inizialmente viene visualizzato sul display il numero 9, poi si passa a 8, poi a 7, e così via. Quando si raggiunge lo 0, l'altoparlante emette un suono e il conto alla rovescia finisce. Il passaggio da uno stato all'altro avviene in corrispondenza di ciascun fronte di salità di un clock. Il pulsante ha lo scopo di far ripartire il conto alla rovescia da 9.

8 Microprogrammazione

Esercizio 1.
L'istruzione

`ABT` *byte*

non esiste in IJVM. L'esecuzione dell'istruzione `ABT` ha come effetto l'aggiunta di una costante a 8 bit *byte* (passata come parametro e da considerarsi come un intero con segno) alla parola che si trova in cima allo stack. Si estenda il microcodice di Mic-1 con una sequenza di microistruzioni per `ABT`. Si valuti poi il miglioramento in performance indotto dall'adozione di `ABT` rispetto all'utilizzo di istruzioni di IJVM.

Soluzione.
Una possibile estensione del microcodice di Mic-1 è la seguente:

abt1	`H=TOS`
abt2	`TOS=MDR=H+MBR`
abt3	`PC=PC+1; fetch`
abt4	`MAR=SP; wr; goto Main1`

Di conseguenza, `ABT` *richiede 5 cicli di clock. D'altra parte, l'istruzione*

`ABT` *byte*

può essere simulata tramite la sequenza

```
BIPUSH byte
IADD
```

che però richiede 8 *cicli di clock. Il miglioramento è quindi pari a* 3/8*, ossia al* 37,5%*.*

Esercizio 2.
L'istruzione

$$\texttt{ACP}\ \ \mathit{offset}_1\ \ \mathit{offset}_2$$

non esiste in IJVM. L'esecuzione dell'istruzione `ACP` ha come effetto il calcolo della somma di due parole di memoria provenienti dalla porzione costante. La somma va poi copiata in cima allo stack, nella prima posizione libera. I parametri offset_1 e offset_2 sono interi senza segno a 8 bit, vanno entrambi interpretati come spiazzamenti rispetto al registro `CPP` e sono utilizzati per recuperare le due parole di memoria da sommare. Si estenda il microcodice di Mic-1 con una sequenza di microistruzioni per `ACP`. Si valuti poi il miglioramento in performance indotto dall'adozione di `ACP` rispetto all'utilizzo di istruzioni di IJVM.

Soluzione.
Una possibile estensione del microcodice di Mic-1 è la seguente:

```
acp1  PC=PC+1; fetch
acp2  H=MBRU
acp3  MAR=CPP+H; rd
acp4  H=MBRU
acp5  MAR=CPP+H; rd
acp6  H=MDR
acp7  MAR=SP=SP+1
acp8  PC=PC+1; fetch
acp9  MDR=TOS=MDR+H; write
```

Di conseguenza, `ACP` *richiede* 10 *cicli di clock. D'altra parte, l'istruzione*

$$\texttt{ACP}\ \ \mathit{offset}_1\ \ \mathit{offset}_2$$

può essere simulata tramite la sequenza

$$\begin{array}{l}\texttt{LDC\_W}\ \ \mathit{offset}_1\\ \texttt{LDC\_W}\ \ \mathit{offset}_2\\ \texttt{IADD}\end{array}$$

che però richiede 20 *cicli di clock. Il miglioramento è quindi pari a* 1/2*, ossia al* 50%*.*

Esercizio 3.
L'istruzione

`POPTWO`

non esiste in IJVM. L'istruzione `POPTWO` non fa altro che rimuovere la parola che si trova in seconda posizione, in cima allo stack (e che quindi, ha indirizzo immediatamente inferiore a quello della parola che si trova in cima allo stack). La parola che si trova in cima allo stack non deve essere rimossa. Si estenda il microcodice di Mic-1 con una sequenza di microistruzioni per `POPTWO`. Si valuti poi il miglioramento in performance indotto dall'adozione di `POPTWO` rispetto all'utilizzo di istruzioni di IJVM.

Soluzione.
Una possibile estensione del microcodice di Mic-1 è la seguente:

```
poptwo1  MAR=SP=SP-1
poptwo2  MDR=TOS; wr; goto Main1
```

Di conseguenza, `POPTWO` *richiede* 3 *cicli di clock. D'altra parte, l'istruzione*

`POPTWO`

può essere simulata tramite la sequenza

```
SWAP
POP
```

che però richiede 11 *cicli di clock. Il miglioramento è quindi pari a* 8/11 .

Esercizio 4.
L'istruzione

`IADDTHREE`

non esiste in IJVM. L'istruzione `IADDTHREE` non fa altro che rimuovere le tre parole che si trovano in cima allo stack, sommarle, e mettere il risultato di tale operazione di somma in cima allo stack. Si estenda il microcodice di Mic-1 con una sequenza di microistruzioni per `IADDTHREE`. Si valuti poi il miglioramento in performance indotto dall'adozione di `IADDTHREE` rispetto all'utilizzo di istruzioni di IJVM.

Soluzione.
Una possibile estensione del microcodice di Mic-1 è la seguente:

```
iaddthree1   MAR=SP=SP-1; rd
iaddthree2   H=TOS
iaddthree3   MAR=SP=SP-1; rd
iaddthree4   H=H+MDR
iaddthree5   TOS=MDR=H+MDR; wr; goto Main1
```

Di conseguenza, `IADDTHREE` *richiede* 6 *cicli di clock. D'altra parte, l'istruzione*

`IADDTHREE`

può essere simulata tramite la sequenza

```
IADD
IADD
```

che però richiede 8 *cicli di clock. Il miglioramento è quindi pari a* 1/4 .

Esercizio 5.
L'istruzione

`MULTHREE`

non esiste in IJVM. L'istruzione `MULTHREE` rimuove la parola che si trova in cima allo stack, la moltiplica per 3 e posiziona il risultato in cima allo stack. Si estenda il microcodice di Mic-1 con una sequenza di microistruzioni per `MULTHREE`. Si valuti poi il miglioramento in performance indotto dall'adozione di `MULTHREE` rispetto all'utilizzo di istruzioni di IJVM.

Esercizio 6.
L'istruzione

`SWAPM` *varnum*

non esiste in IJVM. L'istruzione `SWAPM` scambia fra loro il contenuto della parola che si trova il cima allo stack e il contenuto della variabile locale con spiazzamento *varnum*. Si estenda il microcodice di Mic-1 con una sequenza di microistruzioni per `SWAPM`. Si valuti poi il miglioramento in performance indotto dall'adozione di `SWAPM` rispetto all'utilizzo di istruzioni di IJVM.

9 Programmazione in IJVM

Esercizio 1.
Nel modello di memoria di IJVM, sia a il valore della parola a 32 bit con indirizzo pari a SP e sia b il valore della parola con indirizzo pari a $\mathtt{SP} - 1$. Si scriva una sequenza di istruzioni IJVM che toglie dallo stack le due parole che stanno in cima allo stack e mette sullo stack una parola con valore

$$((b \ \mathtt{AND} \ 4) + a) \ \mathtt{OR} \ 5$$

Soluzione.
Un possibile programma IJVM potrebbe essere il seguente:

```
SWAP
BIPUSH 4
IAND
IADD
BIPUSH 5
IOR
```

Esercizio 2.
Scrivere un metodo IJVM che prenda in input due parole a 32 bit, ne restituisca il prodotto. Si supponga che i valori in input siano interi non negativi.

Soluzione.
Il metodo in questione lavora su due parametri, che avranno spiazzamento 1 *e* 2*, rispettivamente. Il metodo avrà bisogno anche di due variabili locali, con spiazzamento* 3 *e* 4 *rispettivamente. La prima funge da contatore, mentre la seconda tiene traccia delle somme parziali durante il calcolo.*

```
        BIPUSH 0
        ISTORE 3
        BIPUSH 0
        ISTORE 4
CICLO:  ILOAD 3
        ILOAD 1
        IF_CMPEQ FINE
        INC 3 1
        ILOAD 4
        ILOAD 2
        IADD
        ISTORE 4
        GOTO CICLO
 FINE:  ILOAD 4
        IRETURN
```

Esercizio 3.
Scrivere un metodo IJVM che prenda in input una parola a 32 bit e ne restituisca il fattoriale.