Esercitazioni di Architettura degli Elaboratori - I
(Espressioni Booleane / Circuiti Logici)

Giorgio Bacci

A.A. 2010/2011

1 Espressioni Booleane

Un'*espressione booleana* (o *formula booleana*) ϕ su variabili (booleane) prese da un insieme $\mathcal{V} = \{A, B, C, \dots\}$ è definita (ricorsivamente) come segue:

- 0 e 1 sono espressioni booleane;
- A è un'espressione booleana, per ogni $A \in \mathcal{V}$;
- se ϕ è un'espressione booleana, allora $\overline{\phi}$ è un'espressione booleana;
- se ϕ, φ sono espressioni booleane, allora $\phi \cdot \varphi$ è un'espressione booleana;
- se ϕ, φ sono espressioni booleane, allora $\phi + \varphi$ è un'espressione booleana.

Esempi di espressioni booleane sono $\phi_1 = A + (B \cdot \overline{C})$, $\phi_2 = \overline{(A + \overline{B})} \cdot (A + 0)$, $\phi_3 = ((A \cdot B) \cdot C) + (D \cdot A)$; in definitiva una qualunque composizione degli operatori $\cdot$, $+$, $\overline{(\,)}$ a partire dalle costanti booleane 0, 1 e dalle variabili $A, B, C, \dots$ in $\mathcal{V}$. Come accade nell'artimetica classica, assumeremo delle regole di precedenza nell'applicazione degli operatori booleani in moto tale da ridurre al minimo l'utilizzo delle parentesi: $\overline{(\,)}$ ha precendenza rispetto a $\cdot$, che a sua volta a precendenza rispetto all'operatore $+$. Inoltre, gli operatori binari $\cdot$ e $+$ sono assunti come associatovi a destra, ossia per $A \cdot B \cdot C$, intendiamo $(A \cdot B) \cdot C$. Le parentesi sono usate solo se è necessario indicare un ordine diverso da quello imposto dalla precedenza. Con un uso minimo delle parentesi, le formule date in precedenza sono scritte: $\phi_1 = A + B \cdot \overline{C}$, $\phi_2 = \overline{A + \overline{B}} \cdot (A + 0)$, $\phi_3 = A \cdot B \cdot C + D \cdot A$.

Nell'aritmetica classica, data un'espressione come $E = a * b + 2$, possiamo assegnare dei valori rispettivamente ad a ed a b e valutare l'espressione che ne risulta. Per esempio, se $a = 2$ e $b = 3$ allora E vale 8. Un concetto analogo vale per le espressioni booleane, dove però sia le variabili che le formule possono assumere solamente valori in $\{0, 1\}$. Gli operatori booleani vengono interpretati nel modo che segue:

ϕ	$\overline{\phi}$
0	1
1	0

ϕ	φ	$\phi \cdot \varphi$
0	0	0
0	1	0
1	0	0
1	1	1

ϕ	φ	$\phi + \varphi$
0	0	0
0	1	1
1	0	1
1	1	1

Ogni formula ϕ in cui occorrono n variabili booleane può essere rappresentata attraverso una *tabella di verità* con 2^n righe (tutti i possibili assegnamenti delle variabili) e $n + 1$ colonne (per convenzione le prime n sono destinate ai valori associati alle variabili, delle *colonne di input*, l'ultima è il valore associato alla valutazione della formula, detta *colonna di output*). Viceversa, ad ogni tabella di verità è possibile associare un'espressione booleana ad essa equivalente. Una possibile procedura consiste nel sommare il

A	B	C	ϕ	
0	0	0	0	
0	0	1	1	$\overline{A}\,\overline{B}\,C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\,\overline{B}\,\overline{C}$
1	0	1	1	$A\,\overline{B}\,C$
1	1	0	1	$A\,B\,\overline{C}$
1	1	1	1	$A\,B\,C$

(a)

A	B	C	ϕ	
0	0	0	0	$A+B+C$
0	0	1	1	
0	1	0	0	$A+\overline{B}+C$
0	1	1	0	$A+\overline{B}+\overline{C}$
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	1	

(b)

Figura 1: (a) Procedura per ottenere una formula data da somme di prodotti; (b) procedura per ottenere una formula data da prodotti di somme.

prodotto delle variabili relative alle righe che associano il valore 1 nella colonna di output; in tale prodotto la variabile compare negata se assume valore 0, altrimenti rimane positiva. Una seconda procedura, duale alla precedente, consiste nel moltiplicare la somma delle variabili relative alle righe che nella colonna di output assumono valore 0; in tale somma le variabili compaiono negate se assumono valore 1 (un'esempio di applicazione delle due procedure è illustrato in Figura 1).

Una conseguenza dell'equivalenza tra le tabelle di verità ed espressioni booleane è che ogni espressione booleana può essere riscritta in due forme canoniche, rispettivamente dette *somma di prodotti* (SP) e *prodotto di somme* (PS), ossia le due tipologie di formule che risultano applicando le due procedure di traduzione di una tabella di verità in un'espressione booleana. Questo risultato ha anche una seconda conseguenza: data una tabella di verità non esiste un'unica espressione booleana che la rappresenti.

Si prendano in cosiderazione le formule $\phi_1 = A + \overline{B}\,C$ e $\phi_2 = A + \overline{B + \overline{C}}$. È facile verificare che esse hanno la stessa tabella di verità mostrata in Figura 1, ovvero sono equilaventi alle formule

$$\phi_{SP} = \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C$$
$$\phi_{PS} = (A + B + C)\,(A + \overline{B} + C)\,(A + \overline{B} + \overline{C})$$

ricavate nell'esempio precedente. Le formule ϕ_1 e ϕ_2 hanno un ovvio pregio: sono più concise. È quindi lecito chiedersi se, data un'espressione booleana ne esiste un'altra ad essa equivalente che sia "più semplice", ossia che utilizzi il minor numero possibile di operatori logici. A tale proposito vi sono diverse proprietà dell'algebra booleana che risultano utili nel manipolare le espressioni logiche:

Legge	**prodotto**	**somma**
Identità	$1 \cdot \phi = \phi$	$0 + \phi = \phi$
Elemento nullo	$0 \cdot \phi = 0$	$1 + \phi = 1$
Idempotenza	$\phi \cdot \phi = \phi$	$\phi + \phi = \phi$
Inverso	$\phi \cdot \overline{\phi} = 0$	$\phi + \overline{\phi} = 1$
Commutatività	$\phi \cdot \psi = \psi \cdot \phi$	$\phi + \psi = \psi + \phi$
Associatività	$(\phi \cdot \psi) \cdot \varphi = \phi \cdot (\psi \cdot \varphi)$	$(\phi + \psi) + \varphi = \phi + (\psi + \varphi)$
Distributività	$\phi \cdot (\psi + \varphi) = (\phi \cdot \psi) + (\phi \cdot \varphi)$	$\phi + (\psi \cdot \varphi) = (\phi + \psi) \cdot (\phi + \varphi)$
Assorbimento	$\phi \cdot (\phi + \psi) = \phi$	$\phi + (\phi \cdot \psi) = \phi$
DeMorgan	$\overline{\phi \cdot \psi} = \overline{\phi} + \overline{\psi}$	$\overline{\phi + \psi} = \overline{\phi} \cdot \overline{\psi}$
Doppia negazione	$\overline{\overline{\phi}} = \phi$	

Queste regole posso essere utilizzate per semplificare un'espressione dell'algebra booleana oppure renderla in una qualche forma normale nota (ad esempio SP o PS). Vediamo un esempio pratico di come si utilizzino queste leggi di equivalenza dimostrando l'equivalenza tra ϕ_{SP} e ϕ_1.

$$
\begin{aligned}
\phi_{SP} &= \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C \\
&= \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,(B\,\overline{C} + B\,C) && [\,\text{distributività}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,B\,(\overline{C} + C) && [\,\text{distributività}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,B\,(C + \overline{C}) && [\,\text{commutatività}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,B\,1 && [\,\text{inverso}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,B && [\,\text{identità}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,\overline{B}\,(\overline{C} + C) + A\,B && [\,\text{distributività}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,\overline{B}\,(C + \overline{C}) + A\,B && [\,\text{commutatività}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,\overline{B}\,1 + A\,B && [\,\text{inverso}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,\overline{B} + A\,B && [\,\text{identità}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,(\overline{B} + B) && [\,\text{distributività}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,(B + \overline{B}) && [\,\text{commutatività}\,] \\
&= \overline{A}\,\overline{B}\,C + A\,1 && [\,\text{inverso}\,] \\
&= \overline{A}\,\overline{B}\,C + A && [\,\text{identità}\,] \\
&= A + \overline{A}\,\overline{B}\,C && [\,\text{commutatività}\,] \\
&= (A + \overline{A})\,(A + \overline{B}\,C) && [\,\text{distributività}\,] \\
&= 1\,(A + \overline{B}\,C) && [\,\text{inverso}\,] \\
&= A + \overline{B}\,C = \phi_1 && [\,\text{identità}\,]
\end{aligned}
$$

In questo esempio si sono volute applicare le regole in modo rigoroso, mostrando l'effettiva procedura formale di semplificazione. Una volta presa dimestichezza con le regole di semplificazione, molti passaggi potranno essere applicati simultaneamente, e da ora in poi assumeremo che tutte le regole possano essere applicate a meno di commutatività (degli addendi/moltiplicandi).

Esercizio 1.1 *Si dimostrino le equivalenze $\phi_1 = \phi_2$, $\phi_{SP} = \phi_{PS}$ e $\phi_{SP} = \phi_2$, applicando le regole base di equivalenza delle espressioni booleane (mostrando tutti i passaggi intermedi).*

Esempio 1.2

Semplificare la seguente espressione booleana mostrando i passaggi intermedi

$$\phi_{(1.2)} = (A + (B\,\overline{A}\,\overline{B}))\,\overline{(\overline{A} + \overline{B})}\,.$$

$$
\begin{aligned}
\phi_{(1.2)} &= (A + (B\,\overline{A}\,\overline{B}))\,\overline{(\overline{A} + \overline{B})} \\
&= (A + (\overline{A}\,0))\,\overline{(\overline{A} + \overline{B})} && [\,\text{inverso}\,] \\
&= (A + \overline{A})\,\overline{(\overline{A} + \overline{B})} && [\,\text{elemento nullo}\,] \\
&= 1\,\overline{(\overline{A} + \overline{B})} && [\,\text{inverso}\,]
\end{aligned}
$$

$$
\begin{aligned}
&= \overline{(\overline{A} + \overline{B})} && [\text{identità}] \\
&= \overline{\overline{A\,B}} && [\text{de morgan}] \\
&= A\,B && [\text{doppia negazione}]
\end{aligned}
$$

Esempio 1.3

Riscrivere nella forma somma di prodotti (SP) la seguente espressione booleana mostrando i passaggi intermedi

$$\phi_{(1.3)} = (A + B\,\overline{C})\,(A\,(A + \overline{B}\,C)) + \overline{A + B}$$

$$
\begin{aligned}
\phi_{(1.3)} &= (A + B\,\overline{C})\,(A\,(A + \overline{B}\,C)) + \overline{A + B} \\
&= (A + B\,\overline{C})\,(A\,A + A\,\overline{B}\,C) + \overline{A + B} && [\text{distribitività}] \\
&= (A + B\,\overline{C})\,(A + A\,\overline{B}\,C) + \overline{A + B} && [\text{idempotenza}] \\
&= A + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,\overline{B}\,B\,\overline{C}\,C + \overline{A + B} && [\text{distributività + idempotenza}] \\
&= A + A\,\overline{B}\,C + A\,B\,\overline{C} + \overline{A + B} && [\text{inverso + identità + idempotenza}] \\
&= A + A\,\overline{B}\,C + A\,B\,\overline{C} + \overline{A}\,\overline{B} && [\text{de morgan}]
\end{aligned}
$$

Si noti che nonostante l'espressione ottenuta sia in forma SP, essa può essere semplificata ulteriormente.

Esercizio 1.4 *Si semplifichi il più possibile l'espressione ottenuta nell'esempio 1.3.*

1.1 Completezza degli operatori NAND e NOR

Le leggi di De Morgan suggeriscono una caratterizzazione alternativa per le espressioni booleane, che fanno utilizzo di due connettivi booleani derivati: $\phi * \psi = \overline{\phi \cdot \psi}$ (NOR) e $\phi \oplus \psi = \overline{\phi + \psi}$ (NAND), le cui tabelle di verità sono le seguenti:

ϕ	ψ	$\phi * \psi$
0	0	1
0	1	1
1	0	1
1	1	0

ϕ	ψ	$\phi \oplus \psi$
0	0	1
0	1	0
1	0	0
1	1	0

Infatti, data una formula in forma SP è facile vedere che applicando le sole leggi della doppia negazione, dell'idempotenza e di De Morgan è possibile ottenere un'espressione ad essa equivalente che faccia uso **solo** di connettivi NAND oppure di connettivi NOR. Vediamo due "semplici" esempi:

$$
\begin{aligned}
A\,B\,\overline{C} + \overline{A}\,\overline{B} &= \overline{\overline{A\,B\,\overline{C} + \overline{A}\,\overline{B}}} \\
&= \overline{A\,B\,\overline{C}} * \overline{\overline{A}\,\overline{B}} \\
&= (\overline{A} + \overline{B} + C) * (A + B) \\
&= (\overline{\overline{\overline{A} + \overline{B}} + C}) * \overline{\overline{A + B}} \\
&= ((A * B) + C) * (\overline{A} * \overline{B}) \\
&= (\overline{\overline{(A * B) + C}}) * ((\overline{A} + \overline{A}) * (\overline{B} + \overline{B})) \\
&= (((A * B) * (A * B)) * C) * ((A * A) * (B * B))
\end{aligned}
$$

$$
\begin{aligned}
A\,B\,\overline{C} + \overline{A}\,\overline{B} &= \overline{\overline{(A\,B)\,\overline{C} + \overline{A}\,\overline{B}}} \\
&= \overline{((A\,B)\,\overline{C}) \oplus (\overline{A}\,\overline{B})} \\
&= \overline{((A\,B)\,\overline{C}) \oplus (A \oplus B)} \\
&= \overline{\overline{(A\,B)\,\overline{C}} \oplus (A \oplus B)} \\
&= \overline{(\overline{A\,B} \oplus C) \oplus (A \oplus B)} \\
&= \overline{((A \oplus B) \oplus C) \oplus (A \oplus B)} \\
&= \big(((A \oplus B) \oplus C) \oplus (A \oplus B)\big) \oplus \big(((A \oplus B) \oplus C) \oplus (A \oplus B)\big)
\end{aligned}
$$

La traduzione necessita di un po' di dimestichezza con le equivalenze algebriche. Tuttavia, esiste una procedura standard che fa uso delle seguenti rappresentazioni degli operatori booleani:

in forma NAND	**in forma NOR**
$\overline{\phi} = (\phi * \phi)$	$\overline{\phi} = (\phi \oplus \phi)$
$\phi \cdot \psi = \overline{\phi * \psi}$	$\phi \cdot \psi = \overline{\overline{\phi} \oplus \overline{\psi}}$
$\phi + \psi = (\overline{\phi} * \overline{\psi})$	$\phi + \psi = \overline{\phi \oplus \psi}$

1. rendere tutte le somme e moltiplicazioni binarie (raccogliendo tra le parentesi);
2. tradurre tutte le somme e moltiplicazioni nella rispettiva rappresentazione NAND/NOR;
3. eliminare tutte le doppie negazioni (secondo la legge della doppia negazione);
4. tradurre tutte le negazioni rimaste utilizzando la rispettiva rappresentazione NAND/NOR.

Esercizio 1.5 *Riscrivere le seguenti espressioni logiche utilizzando **solo** connettivi NAND, e successivamente **solo** connettivi NOR, applicando l'algoritmo di traduzione standard.*

$$\overline{(A+B)}\,(\overline{A}+C) \qquad\qquad A\,B + C \qquad\qquad A\,B + \overline{C}\,.$$

2 Circuiti logici

Tra le espressioni booleane e i circuiti logici c'è un'ovvia corrispondenza: ad ogni connettivo booleano corrisponde esattamente una porta logica con la stessa interpretazione logica. Questa corrispondenza è molto utile quando è rischiesto di sintetizzare un circuito combinatorio a partire da una specifica data. Infatti, tutte le proprietà delle espressioni booleane che abbiamo visto nella sezione precedente possono essere applicate per ottenere circuiti minimali: ovvero che faccia uso del minor numero possibile di porte logiche. Un'altro risultato molto utile (per lo più in termini implementativi) è la completezza dell'algebra booleana rispetto alle porte (o operatori) NAND e NOR. Riuscire a descrivere ogni espressione booleana o tabella di verità attraverso un circuito che utilizzi un'unica tipologia di porta logica permette di poter produrre un solo tipo di porta logica "universale", ossia, che vada bene per qualunque implementazione hardware.

In questa sezione vedremo alcuni esempi pratici, tralasciando la parte di teoria che può comunque essere trovata nel libro di testo del corso.

Esempio 2.1

Si progetti un circuito combinatorio che funzioni da "operatore di maggioranza": dati n bit di ingresso, l'uscita Out vale 1 se e solo se il numero di bit in ingresso che valgono 1 è maggiore o uguale al numero di bit in ingresso che valgono 0.

1. Si progetti un circuito con il minor numero di porte NOT, AND e OR per $n = 3$;
2. Si progetti un circuito con sole porte NAND per $n = 3$.

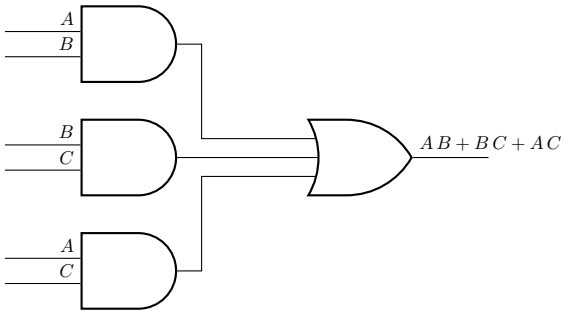
Risolviamo dapprima il punto (1). Il modo più semplice di ottenere un circuito combinatorio è di definire la tabella di verità corrispondente. Da quest'ultima ottenere un'espressione booleana è banale, basta infatti applicare il metodo di traduzione standard.

A	B	C	ϕ	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}\,B\,C$
1	0	0	0	
1	0	1	1	$A\,\overline{B}\,C$
1	1	0	1	$A\,B\,\overline{C}$
1	1	1	1	$A\,B\,C$

L'espressione deve essere semplificata utilizzando le regole algebriche:

$$
\begin{aligned}
Out &= \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C \\
&= \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,(\overline{C} + C) && [\text{distributività (prodotto)}] \\
&= \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B && [\text{inverso + identità}] \\
&= \overline{A}\,B\,C + A\,(\overline{B}\,C + B) && [\text{distributività (prodotto)}] \\
&= \overline{A}\,B\,C + A\,(\overline{B} + B)\,(B + C) && [\text{distributività (somma)}] \\
&= \overline{A}\,B\,C + A\,(B + C) && [\text{inverso + identità}] \\
&= \overline{A}\,B\,C + A\,B + A\,C && [\text{distributività (prodotto)}] \\
&= B\,(\overline{A}\,C + A) + A\,C && [\text{distributività (prodotto)}] \\
&= B\,(\overline{A} + A)\,(A + C) + A\,C && [\text{distributività (somma)}] \\
&= B\,(A + C) + A\,C && [\text{inverso + identità}] \\
&= A\,B + B\,C + A\,C && [\text{distributività}]
\end{aligned}
$$

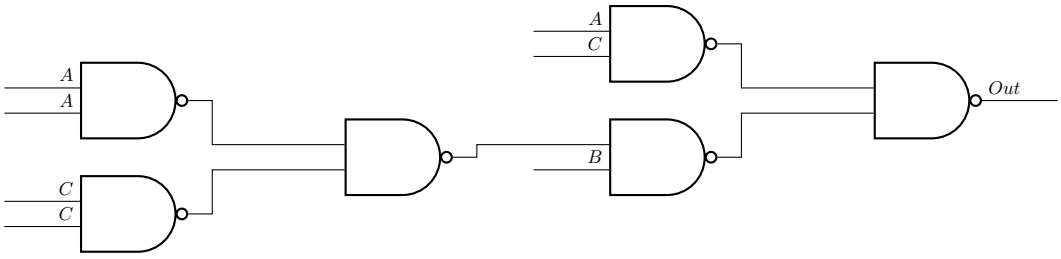
Il circuito corrispondente è il seguente:



Per risolvere il punto (2) applichiamo l'algoritmo di traduzione in forma NAND sull'espressione che abbiamo ricavato al penultimo passo di semplificazione (solamente per un fatto di comodità).

$$
\begin{aligned}
Out &= B\,(A + C) + A\,C \\
&= \big(B\,(A + C)\big) + (A\,C) && [\text{passo 1: somme e prodotti resi binari}] \\
&= \overline{\overline{\big(B * (\overline{A} * \overline{C})\big)}} * \overline{\overline{(A * C)}} && [\text{passo 2: somme e prodotti in forma NAND}] \\
&= \big(B * (\overline{A} * \overline{C})\big) * (A * C) && [\text{passo 3: eliminazione doppie negazioni}] \\
&= \Big(B * \big((A * A) * (C * C)\big)\Big) * (A * C) && [\text{passo 4: negazioni in forma NAND}]
\end{aligned}
$$

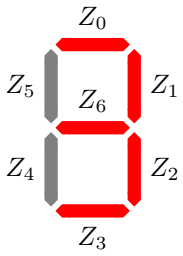
Il circuito ottenuto dalla traduzione:



Esempio 2.2

Si progetti un circuito combinatorio che ricevuti in ingresso 3 bit, I_0, I_1, I_2 visualizzi in due display analogici, *Zeros* e *Ones*, rispettivamente, il numero di bit uguali a 0 e il numero di bit uguali a 1. Si definiscano le tabelle di verità per ognuno dei led che compongono i due display, utilizzando i nomi Z_i e O_i (per $0 \leq i \leq 6$) rispettivamente per i display *Zeros* e *Ones*.

Un display analogico è composto da sette led, quindi definiamo sette tabelle di verità per ciascun display. A titolo d'esempio, mostriamo solo il caso del display *Zeros*, che visualizza il numero di zero nei bit d'ingresso I_0, I_1, I_2. Qui sotto è mostrata la disposizione dei led, ed un esempio di output nel caso $I_0 = I_1 = I_2 = 0$.



Disposizione dei led sul display

I_0	I_1	I_2	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6
0	0	0	1	1	1	1	0	0	1
0	0	1	1	1	0	1	1	0	1
0	1	0	1	1	0	1	1	0	1
0	1	1	0	1	1	0	0	0	0
1	0	0	1	1	0	1	1	0	1
1	0	1	0	1	1	0	0	0	0
1	1	0	0	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	0

Dalla tabella di verità si ottengono le espressioni booleane relative ai led Z_i (per $0 \leq i \leq 6$) applicando le tecniche di traduzione. Ogni espressione dovrà essere poi semplificata prima di ottenere il circuito.

Di seguito mostriamo tutti i passaggi solo per il caso del led Z_0, la cui espressione booleana è stata ottenuta applicando la traduzione nella forma PS (motivo: ci sono solo tre 0 nella tabella, quindi è più conveniente applicare questa traduzione piuttosto che quella nella forma SP).

$$
\begin{aligned}
Z_0 &= (I_0 + \overline{I_1} + \overline{I_2})\,(\overline{I_0} + I_1 + \overline{I_2})\,(\overline{I_0} + \overline{I_1} + I_2) \\
&= (I_0 + \overline{I_1} + \overline{I_2})\,(\overline{I_0} + \big((I_1 + \overline{I_2})\,(\overline{I_1} + I_2)\big)) && [\text{distributività (somma)}] \\
&= (I_0 + \overline{I_1} + \overline{I_2})\,(\overline{I_0} + I_1\,I_2 + \overline{I_1}\,\overline{I_2}) && [\text{distributività (prodotto) + inverso + identità}] \\
&= I_0\,I_1\,I_2 + I_0\,\overline{I_1}\,\overline{I_2} + \overline{I_0}\,\overline{I_1} + \overline{I_1}\,\overline{I_2} + \overline{I_0}\,\overline{I_2} && [\text{distributività + inverso + identità + idempotenza}] \\
&= I_0\,I_1\,I_2 + \overline{I_0}\,\overline{I_1} + \overline{I_0}\,\overline{I_2} + \overline{I_1}\,\overline{I_2}\,(I_0 + 1) && [\text{distributività}] \\
&= I_0\,I_1\,I_2 + \overline{I_0}\,\overline{I_1} + \overline{I_0}\,\overline{I_2} + \overline{I_1}\,\overline{I_2} && [\text{elemento nullo + identità}]
\end{aligned}
$$

Le espressioni semplificate per i led rimanenti sono le seguenti:

$$
\begin{aligned}
Z_1 &= 1 & Z_4 &= \overline{I_0}\,I_1\,\overline{I_2} + I_0\,I_1\,I_2 + I_0\,\overline{I_1}\,\overline{I_2} + \overline{I_0}\,\overline{I_1}\,I_2 \\
Z_2 &= \overline{I_0}\,\overline{I_1}\,\overline{I_2} + \overline{I_0}\,\overline{I_1} + \overline{I_0}\,\overline{I_2} + I_1\,I_2 & Z_5 &= I_0\,I_1\,I_2 \\
Z_3 &= Z_0 & Z_6 &= \overline{I_0}\,\overline{I_1} + \overline{I_0}\,\overline{I_2} + \overline{I_1}\,\overline{I_2}
\end{aligned}
$$

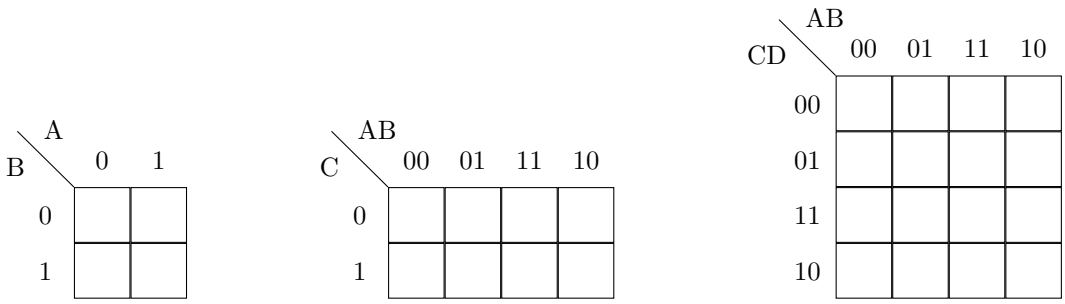


Figura 2: Forma base delle Mappe di Karnaugh: (a) due variabili, (b) tre variabili, (c) quatto variabili.

Esercizio 2.3 *Si completi l'esempio 2.2, applicando il procedimento mostrato nel caso del display Zeros al caso del display Ones.*

3 Mappe di Karnaugh (INF)

Le semplificazioni di un'espressione booleana possono essere effettuate mediante i teoremi dell'algebra di booleana. Esiste però un metodo "più pratico" di semplificazione: le *mappe di Karnaugh*. Tale metodo è di semplice utilizzo nel caso in cui le espressioni contengano poche variabili, in genere fino a quattro o cinque, risulta meno efficace se il numero delle variabili è maggiore di cinque. In Figura 2 sono riportate le mappe di Karnaugh per funzioni di due, tre o quattro variabili.

Ogni mappa rappresenta una tavola di verità, dove i valori sono disposti in modo tale da facilitare la semplificazione delle formule. Caselle con un lato in comune sono dette adiacenti; debbono essere considerate adiacenti anche le caselle all'estremità di una riga o di una colonna, come se la mappa fosse disegnata su una superficie toroidale chiusa (una ciambella). Le caselle inoltre sono disposte in modo tale che passando da una qualsiasi cella ad una adiacente sulla stessa riga o sulla stessa colonna cambia di valore una sola variabile. Questa disposizione è cruciale ai fini delle tecniche di semplificazione che andremo a descrivere, e spiega il motivo per il quale gli indici delle colonne e delle righe nella mappa di Karnaugh a quattro variabili sono indicizzate secondo l'ordinamento 00, 01, 11, 10, invece di quello canonico (ossia, 00, 01, 01, 11).

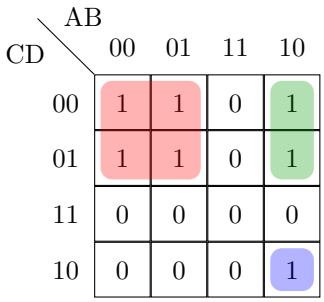
Definizione 3.1 (Mintermine) *Data una mappa di Karnaugh un* mintermine *è un insieme di caselle adiacenti che soddisfa i seguenti requisiti:*

- *tutte le caselle contengono lo stesso valore;*
- *le caselle formano un quadrato oppure un rettangolo sulla mappa (secondo le regole di adiacenza);*
- *il numero di caselle che compongono il mintermine è una potenza perfetta di* 2*.*

Un mintermine si dice positivo *se le caselle contengono il valore* 1*, si dice* negativo *se contengono il valore* 0*. Un mintermine è detto* massimale *se non è contenuto (interamente) in un altro mintermine.*

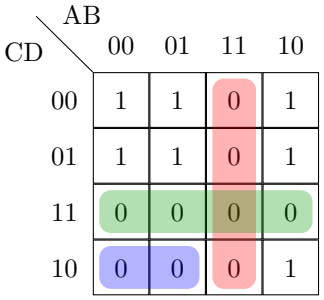
Esattamente come avveniva nel caso delle tabelle di verità, anche per le mappe di Karnaugh esistono due procedure per la derivazione di un'espressione booleana. Le due procedure differiscono in base alla tipologia dei mintermini che vengono utilizzati: positivi o negativi. Il metodo procede nel modo seguente. Si localizzano sulla mappa i mintermini positivi che raccolgono tutti gli 1 della mappa. Ad ogni mintemine positivo viene associato il prodotto delle **sole** variabili che al suo interno **non** cambiano il proprio valore, negando quelle con valore 0. La formula che rappresenta la mappa è data dalla somma dei prodotti associati ai mintermini positivi che sono stati raccolti. Dualmente, si può procedere anche raccogliendo tutti gli 0 della mappa mediante mintermini negativi, ai quali vengono però associati le somme delle variabili che non cambiano di valore, negandole quando sono a 1.

Vediamo con un esempio come si utilizzano le due tecniche appena descritte.



CD \ AB	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	0	0	0	0
10	0	0	0	1

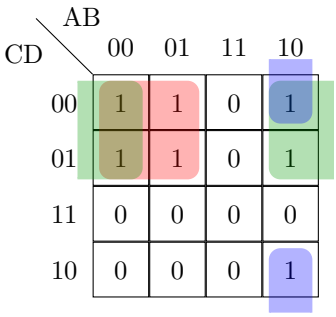
$\overline{A}\,\overline{C} + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C\,\overline{D}$



CD \ AB	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	0	0	0	0
10	0	0	0	1

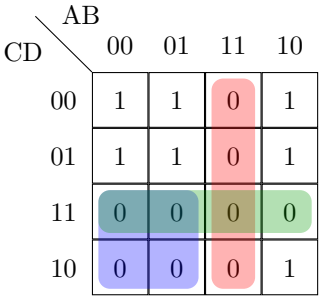
$(\overline{A} + \overline{B}) \cdot (\overline{C} + \overline{D}) \cdot (A + \overline{C} + D)$

Rispetto alla tecnica che traduce le tabelle di verità in espessioni algebriche, questo metodo è più comodo perchè restituisce formule già semplificate. Se consideriamo le espressioni appena ottenute, però, quelli che hanno più esperienza con le semplificazioni algebriche si saranno sicuramente accorti che sono ulteriormente semplificabili. Fortunatamente non sono gli unici a potersene accorgene, infatti, lo si nota anche osservando che i mintermini considerati nel raccoglimento non sono massimali.



CD \ AB	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	0	0	0	0
10	0	0	0	1

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

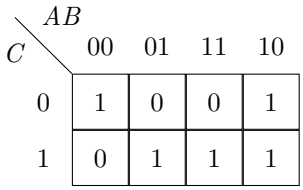


CD \ AB	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	0	0	0	0
10	0	0	0	1

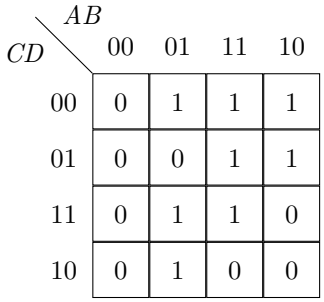
$(\overline{A} + \overline{B}) \cdot (\overline{C} + \overline{D}) \cdot (A + \overline{C})$

L'utilizzo di soli mintermini massimali garantisce che la traduzione risultante sia "più semplificata". Per ottenere un'espressione totalmente semplificata è necessario non solo fare uso di soli mintermini massimali, ma anche usarne il numero minore possibile durante la fase di raccoglimento dei valori nella mappa.

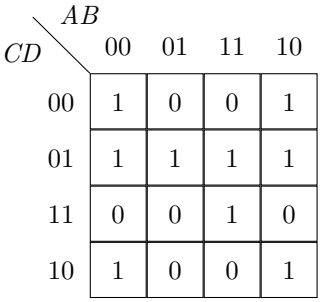
Esercizio 3.2 *Si semplifichino le seguenti mappe di Karnaugh adottando il numero minore raggruppamenti massimali. Si utilizzino allo scopo sia il metodo con mintermini positivi che quello con mintermini negativi.*



C \ AB	00	01	11	10
0	1	0	0	1
1	0	1	1	1



CD \ AB	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	1	1	0
10	0	1	0	0



CD \ AB	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	0	0	1	0
10	1	0	0	1

Le mappe di Karnaugh sostituiscono le tabelle di verità nella fase di traduzione della specifica di un problema in un'espressione booleana. Il vantaggio di questa procedura sta nel poter evitare la fase di semplificazione della formula attraverso le equivalenze algebriche. Vediamone un esempio rifacendo un'esercizio visto nella sezione precedente (ma con qualche bit in più).

Esempio 3.3

Si progetti un circuito combinatorio che funzioni da "operatore di maggioranza": dati n bit di ingresso, l'uscita *Out* vale 1 se e solo se il numero di bit in ingresso che valgono 1 è maggiore o uguale al numero di bit in ingresso che valgono 0. Si progetti il circuito per $n = 4$, disegnando la mappa di Karnaugh e ottenendo una espressione booleana minimizzata.

Identifichiamo i 4 bit di ingresso con le variabili I_0, I_1, I_2, I_3, e definiamo la mappa di Karnaugh mettendo un 1 secondo la specifica del problema. Sulla mappa di Karnaugh così definita raccogliamo tutti i mintermini positivi massimali nel modo che segue:

I_2I_3 \ I_0I_1	00	01	11	10
00	0	0	1	0
01	0	1	1	1
11	1	1	1	1
10	0	1	1	1

$$Out = I_0\,I_1 + I_2\,I_3 + I_1\,I_3 + I_0\,I_3 + I_0\,I_3 + I_0\,I_2$$

La definizione del circuito è ovvia, ed è lasciata come esercizio.

Esercizio 3.4 *Si progetti un circuito combinatorio che ricevuti in ingresso* 4 *bit,* I_0, I_1, I_2, I_3 *rappresentati un numero naturale, nella notazione binaria, e restituisca in uscita il valore* 1 *se l'ingresso rappresenta un numero pari, e il valore* 0 *in caso contrario.*

1. *Si determini la mappa di Karnaugh e la relativa espressione booleana.*
2. *Si rappresenti il circuito con* ***sole*** *porte NAND/NOR.*

3.1 Mappe di Karnaugh per più di quattro variabili

Le mappe di Karnaugh per più di quattro variabili devono essere costruite sempre rispettando la regola che nel passaggio da una casella a quella adiacente sulla stessa riga o sulla stessa colonna deve cambiare una sola variabile. Per quanto riguarda la semplificazione di una funzione a cinque variabili essa può, essere fatta mediante due mappe di Karnaugh a quattro variabili. Le adiacenze possono essere localizzate pensando di sovrapporre le due mappe e considerando adiacenti le caselle che si corrispondono verticalmente.

$A = 0$

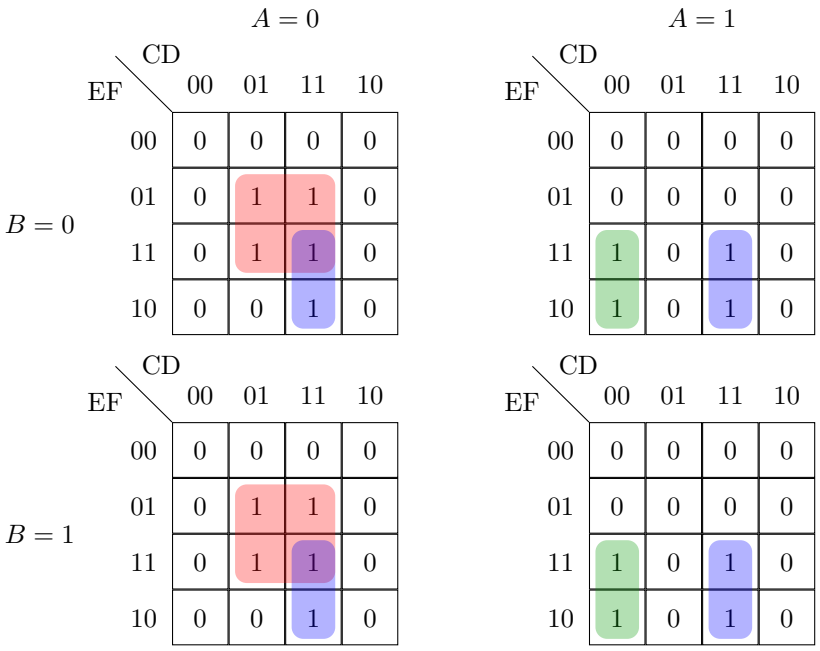
DE \ BC	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	1	0

$A = 1$

DE \ BC	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	0	1	0
10	1	0	1	0

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

Naturalmente all'aumentare del numero delle variabili aumenta il numero di caselle della mappa di Karnaugh corrispondente e di conseguenza anche la difficoltà nell'individuare i mintermini massimali. Come ultimo esempio, vediamo come si definisce una mappa di Karnaugh a sei variabili. In questa occasione è comodo pensare ad un concetto di adiacenza tra mappe: solo le mappe tra loro adiacenti possono essere sovrapposte per trovare i mintermini a cavallo tra sotto-mappe differenti. Il numero di sovrapposizioni simultanee tra sotto-mappe possono essere solo una potenza perfetta di 2.



Nell'esempio in figura, tutti i mintermini considerati sono a cavallo tra sotto-mappe distinte. Si noti che per il mintermine in rosso la sovrapposizione è tra le sotto-mappe per le quali $A = 0$, mentre nel caso del mintermine in blu la sovrapposizione è tra tutte e quattro le sotto-mappe.

Esercizio 3.5 *Si scriva l'espressione booleana corrispondente ai raggruppamenti fatti nell'esempio dalla mappa di Karnaugh a sei variabili, qui sopra.*

3.2 Mappe di Karnaugh con condizioni di indifferenza

Accade spesso che il valore di output di un'assegnata tabella di verità non venga specificato per alcune combinazioni delle variabili d'ingresso, o perché queste combinazioni non possono verificarsi oppure perché più in generale, non interessa conoscere i valori dell'uscita corrispondenti a tali combinazioni. Si parla così di condizioni di indifferenza. In questa situazione l'uscita, che può assumere indifferentemente il valore 0 o 1, viene riportata sulla mappa di Karnaugh con il simbolo "–". Le condizioni di indifferenza possono essere sfruttate al fine di semplificare l'espressione booleana associata alla mappa assegnando il valore 1 o 0 quando ciò è conveniente.

(a) (b)

Nell'esempio sopra in (b) si considera la condizione di indifferenza nel caso $A = 0$, $B = 0$, $C = 0$ come se avesse valore 1. L'espressione che ne risulta è più semplificata rispetto a quella che ne risulterebbe dai raggruppamenti fatti in (a) (Domanda: quali sono le due espressioni?).

Esercizio 3.6 *Si progetti un circuito combinatorio che dati n bit di ingresso, restituisca in uscita* 1 *se il numero di bit in ingresso che valgono* 1 *è pari a* 2, 0 *se il numero è pari a* 3.

Si definiscano le mappe di Karnaugh (con condizioni di indifferenza) nei casi $n = 3$, $n = 4$ *e* $n = 5$, *e in ciascuno di essi si determini l'espressione booleana semplificata associata.*