Corso di Architettura degli Elaboratori e Laboratorio (M-Z)

Algebra Booleana della Commutazione

Nino Cauli

UNIVERSITÀ degli STUDI di CATANIA

Dipartimento di Matematica e Informatica

Algebra booleana della commutazione

- L’**Algebra Booleana della Commutazione** è un sistema algebrico in cui ogni variabile può assumere solo **2 valori** (0 e 1)
- Possiede 3 operazioni base definite su variabili binarie (**FUNZIONI LOGICHE FONDAMENTALI**):
- **Somma logica** o **OR**
- **Prodotto logico** o **AND**
- **Complementazione**, Negazione, Inversione o **NOT**
- Ciascuna operazione prende in ingresso una o più variabili binarie e rende in uscita una variabile binaria

OR o somma logica

- La somma logica o OR è una funzione che vale 1 solo se almeno uno dei suoi ingressi binari vale 1
- Si denota tramite gli operatori a due argomenti "+" o "$\vee$"
- La forma algebrica della somma è:

$$f(x_1, x_2) = x_1 + x_2 = x_1 \vee x_2$$

x_1	x_2	$f(x_1,x_2) = x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Proprietà base della somma logica

Proprietà commutativa:

$$x_1 + x_2 = x_2 + x_1$$

Proprietà associativa:

$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$$

Estensione a più variabili:

$$f = x_1 + x_2 + \ldots + x_n$$

Proprietà dell'elemento neutro:

$$0 + x = x$$

AND o prodotto logico

- Il prodotto logico o AND è una funzione che vale 1 solo se tutti i suoi ingressi binari valgono 1

- Si denota tramite gli operatori a due argomenti “$\cdot$” o “$\bigwedge$”

- La forma algebrica del prodotto è:

$$f(x_1, x_2) = x_1 \cdot x_2 = x_1 \wedge x_2$$

x_1	x_2	$f(x_1,x_2) = x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Proprietà base del prodotto logico

UNIVERSITÀ degli STUDI di CATANIA

Proprietà commutativa:

$$x_1 \cdot x_2 = x_2 \cdot x_1$$

Proprietà associativa:

$$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3$$

Estensione a più variabili:

$$f = x_1 \cdot x_2 \cdot \ldots \cdot x_n$$

Proprietà dell’elemento neutro:

$$1 \cdot x = x$$

NOT o complementazione

- La complementazione o NOT è una funzione che inverte il valore dell’unica variabile in ingresso
- Si denota tramite l’operatore di soprallineatura “¯” o di negazione “¬”
- La forma algebrica della complementazione è:

$$f(x) = \overline{x} = \neg x$$

- Proprietà di involuzione (doppia negazione):

$$\overline{\overline{x}} = x$$

x	f(x) = ¬x
0	1
1	0

Altre proprietà duali

SOMMA	PRODOTTO
Proprietà distributiva:	
$x + y \cdot z = (x + y) \cdot (x + z)$	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
Proprietà di idempotenza:	
$x + x = x$	$x \cdot x = x$
Proprietà di complemento:	
$x + \overline{x} = 1$	$x \cdot \overline{x} = 0$
Proprietà dello 1 e dello 0:	
$1 + x = 1$	$0 \cdot x = 0$

Teoremi di De Morgan

Addizione:

$$\overline{x_1 + x_2 + x_3 \ldots} = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \ldots$$

Prodotto:

$$\overline{x_1 \cdot x_2 \cdot x_3 \ldots} = \overline{x_1} + \overline{x_2} + \overline{x_3} \ldots$$

Funzioni logiche e tabelle di verità

- **FUNZIONE LOGICA**: Funzione con una o più variabili **BINARIE** di ingresso ed una variabile **BINARIA** di uscita
- Una Funzione Logica può essere espressa con una **TABELLA DI VERITÀ**
- Esiste **una sola tabella di verità** per ogni funzione logica
- Una tabella di verità ha **2^n righe** e **n + 1** colonne, dove *n* è il numero di variabili di ingresso

x_1	x_2	...	x_{n-1}	x_n	$f(x_1,x_2,..,x_{n-1},x_n)$
0	0	...	0	0	$y_{00...00}$
0	0	...	0	1	$y_{00...01}$
...	...	...	...	...	...
1	1	...	1	0	$y_{11...10}$
1	1	...	1	1	$y_{11...11}$

Espressioni logiche

- Combinando assieme più funzioni logiche fondamentali si ottengono le **ESPRESSIONI LOGICHE**
- Un'espressione logica è **una possibile realizzazione** di una funzione logica
- Esistono **INFINITE** espressioni logiche che realizzano la **STESSA** funzione (le tabelle di verità, al contrario, sono uniche)

$$(x_1 + x_2)(x_1 + \overline{x_2})(\overline{x_1} + x_2) = x_1 x_2$$

x_1	x_2	$f(x_1,x_2) = x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Equivalenza tra espressioni logiche

- Due espressioni logiche sono **equivalenti** se rappresentano la **stessa funzione**

- Per dimostrare l’equivalenza di due espressioni logiche si possono confrontare le loro tabelle di verità

$$(x_1 + x_2)(x_1 + \overline{x_2})(\overline{x_1} + x_2) = x_1 x_2$$

x_1	x_2	$x_1 \cdot x_2$	$(x_1 + x_2)(x_1 + \neg x_2)(\neg x_1 + x_2)$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Precedenza tra operatori logici

- Per decidere quale operazione eseguire per prima in un’espressione bisogna seguire gli **ordini di precedenza degli operatori**

Operatore	Precedenza
Negazione - **NOT**	1
Prodotto - **AND**	2
Somma - **OR**	3

- Per forzare la precedenza di un operatore si possono usare le parentesi:

$$(x_1 x_2) + (x_1 \overline{x_2}) + (\overline{x_1} x_2) = x_1 x_2 + x_1 \overline{x_2} + \overline{x_1} x_2$$

$$x_1 (x_2 + x_1)(\overline{x_2} + \overline{x_1}) x_2 \neq x_1 x_2 + x_1 \overline{x_2} + \overline{x_1} x_2$$

Da espressione a funzione logica

- Per sapere quale funzione è rappresentata da una espressione logica basta calcolarne la tabella di verità

- Calcolare i valori assunti dall'espressione per tutti i valori delle variabili di ingresso

$$(x_1 + x_2)(x_1 + \overline{x_2})(\overline{x_1} + x_2)$$

x_1	x_2	$x_1 + x_2$	$x_1 + \neg x_2$	$\neg x_1 + x_2$	$f(x_1,x_2) = (x_1 + x_2)(x_1 + \neg x_2)(\neg x_1 + x_2)$
0	0	0	1	1	0
0	1	1	0	1	0
1	0	1	1	0	0
1	1	1	1	1	1

Da funzione a espressione logica

- Esistono infinite espressioni che rappresentano una funzione logica
- Esiste un metodo per trovarne almeno una a partire dalla tabella di verità della funzione?
- Esistono delle forme uniche per rappresentare una funzione?

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Mintermini

- **Mintermine**: funzione a n variabili che vale 1 solo per una specifica configurazione delle variabili

- Assumiamo di saper ottenere le espressioni rappresentanti i mintermini che valgono 1 per tutte le configurazioni in cui la funzione f_1 vale 1: {m_0, m_1, m_3, m_7}

- La **somma logica dei mintermini** equivale alla funzione cercata: $f_1 = m_0 + m_1 + m_3 + m_7$

x_1	x_2	x_3	m_0	m_1	m_3	m_7	$m_0+m_1+m_3+m_7$	f_1
0	0	0	1	0	0	0	1	1
0	0	1	0	1	0	0	1	1
0	1	0	0	0	0	0	0	0
0	1	1	0	0	1	0	1	1
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	1	1	1

Come rappresentare un mintermine

- Un mintermine di una configurazione c di n variabili può essere rappresentato come un prodotto delle sue variabili:
  - In **forma diretta** se in c la variabile **vale 1**
  - In **forma negata** se in c la variabile **vale 0**

x_1	x_2	x_3	$\neg x_1$	$\neg x_2$	x_3	$\neg x_1 \cdot \neg x_2 \cdot x_3$	m_1
0	0	0	1	1	0	0	0
0	**0**	**1**	**1**	**1**	**1**	**1**	**1**
0	1	0	1	0	0	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	1	0	1	1	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	1	0	0

Prima forma canonica

- Quindi una funzione logica può essere rappresentata da una espressione nella forma di **somma di prodotti (SOP)**:

$$\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2 x_3 + \overline{x}_1 x_2 x_3 + x_1 x_2 x_3$$

- Tale forma è unica ed è chiamata **PRIMA FORMA CANONICA**

x_1	x_2	x_3	f_1
0	**0**	**0**	**1**
0	**0**	**1**	**1**
0	1	0	0
0	**1**	**1**	**1**
1	0	0	0
1	0	1	0
1	1	0	0
1	**1**	**1**	**1**

Maxtermini

- **Maxtermine**: funzione a n variabili che vale 0 solo per una specifica configurazione delle variabili

- Assumiamo di saper ottenere le espressioni rappresentanti i maxtermini che valgono 0 per tutte le configurazioni in cui la funzione f_1 vale 0: {M_2, M_4, M_5, M_6}

- Il **prodotto logico dei maxtermini** equivale alla funzione cercata: $f_1 = M_2 \cdot M_4 \cdot M_5 \cdot M_6$

x_1	x_2	x_3	M_2	M_4	M_5	M_6	$M_2 \cdot M_4 \cdot M_5 \cdot M_6$	f_1
0	0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	1	0	0	1	1	1	0	0
0	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	0	0
1	0	1	1	1	0	1	0	0
1	1	0	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1

Come rappresentare un maxtermine

- Un maxtermine di una configurazione c di n variabili può essere rappresentato come una somma delle sue variabili:
  - In **forma diretta** se in c la variabile vale **0**
  - In **forma negata** se in c la variabile vale **1**

x_1	x_2	x_3	x_1	$\neg x_2$	x_3	$x_1 + \neg x_2 + x_3$	M_2
0	0	0	0	1	0	1	1
0	0	1	0	1	1	1	1
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	1	1	0	0	1	1	1
1	0	0	1	1	0	1	1
1	0	1	1	1	1	1	1
1	1	0	1	0	0	1	1
1	1	1	1	0	1	1	1

Seconda forma canonica

- Quindi una funzione logica può essere rappresentata da una espressione nella forma di **prodotto di somme (POS)**:

$$(x_1 + \overline{x}_2 + x_3)(\overline{x}_1 + x_2 + x_3)(\overline{x}_1 + x_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_2 + x_3)$$

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	**1**	**0**	**0**
0	1	1	1
1	**0**	**0**	**0**
1	**0**	**1**	**0**
1	**1**	**0**	**0**
1	1	1	1

- Tale forma è unica ed è chiamata **SECONDA FORMA CANONICA**

Forma minima

- Una espressione si dice in **forma minima** quando non esiste nessun altra espressione equivalente con un costo inferiore

- Per espressioni SOP e POS usiamo il criterio di **costo dei LETTERALI** (ma ne esistono altri): il costo di un espressione è dato dal **numero di comparse di variabili** nell'espressione stessa

- Un'espressione in forma minima è più semplice ed economica da realizzare come circuito rispetto alle altre forme

$$(x_1 + x_2)(x_1 + \overline{x_2})(\overline{x_1} + x_2) = x_1 x_2$$

Costo 6 **Costo 2**

Da prima forma canonica a forma minima

Per passare da prima forma canonica a forma minima si possono seguire i seguenti passi:

- Usando la **proprietà distributiva**, associare le coppie di mintermini che posseggono una sola variabile in forma discordante (diretta e negata)

$$\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2 x_3 = \overline{x}_1\overline{x}_2(\overline{x}_3 + x_3)$$

- Usare la **legge di complemento** per trasformare in 1 le somme di variabili complementari

$$\overline{x}_1\overline{x}_2(\overline{x}_3 + x_3) = \overline{x}_1\overline{x}_2$$

- Usare la **legge di idempotenza** per duplicare dei mintermini nel caso fosse necessario

$$x + x = x$$

Minimizzazione esempio 1

$$
\begin{aligned}
\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2 x_3 + \overline{x}_1 x_2 x_3 + x_1 x_2 x_3 &= \\
= \overline{x}_1\overline{x}_2(\overline{x}_3 + x_3) + x_2 x_3(\overline{x}_1 + x_1) &= \quad (distributiva) \\
= \overline{x}_1\overline{x}_2 \cdot 1 + x_2 x_3 \cdot 1 &= \quad (complemento) \\
= \overline{x}_1\overline{x}_2 + x_2 x_3 &= \quad (forma\ minima)
\end{aligned}
$$

Minimizzazione esempio 2

$$
\begin{aligned}
\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 &= \\
\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 &= \quad (idempotenza) \\
= \overline{x}_1\overline{x}_2(\overline{x}_3 + x_3) + \overline{x}_1\overline{x}_3(\overline{x}_2 + x_2) + x_1\overline{x}_2(\overline{x}_3 + x_3) &= \quad (distributiva) \\
= \overline{x}_1\overline{x}_2 \cdot 1 + \overline{x}_1\overline{x}_3 \cdot 1 + x_1\overline{x}_2 \cdot 1 &= \quad (complemento) \\
= \overline{x}_1\overline{x}_2 + \overline{x}_1\overline{x}_3 + x_1\overline{x}_2 &= \\
= \overline{x}_2(\overline{x}_1 + x_1) + \overline{x}_1\overline{x}_3 &= \quad (distributiva) \\
= \overline{x}_2 \cdot 1 + \overline{x}_1\overline{x}_3 &= \quad (complemento) \\
= \overline{x}_2 + \overline{x}_1\overline{x}_3 &= \quad (forma\ minima)
\end{aligned}
$$

Metodo di Karnaugh

- Semplificazione a forma minima può essere un **processo complicato**
- Il **metodo di Karnaugh** è un metodo **geometrico** che facilita il processo
- Si rappresenta la tabella di verità in forma differente (**mappa di Karnaugh**)
- Si effettua la minimizzazione raggruppando geometricamente i mintermini
- Vantaggioso per **funzioni a poche variabili** (3 o 4)

Mappe di Karnaugh

- **Mappa bidimensionale** che rappresenta una tabella di verità
- Per funzioni a 3 variabili le colonne rappresentano coppie di due variabili e le righe la terza variabile
- Sono ordinate in modo che **caselle adiacenti abbiano solo una variabile dal valore differente**
- Le mappe a 4 variabili sono un'estensione con 2 variabili nelle righe e le altre 2 nelle colonne

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

x_3 \ $x_1 x_2$	0 0	0 1	1 1	1 0
0	1	0	0	0
1	1	1	1	0

Come usare le mappe di Karnaugh

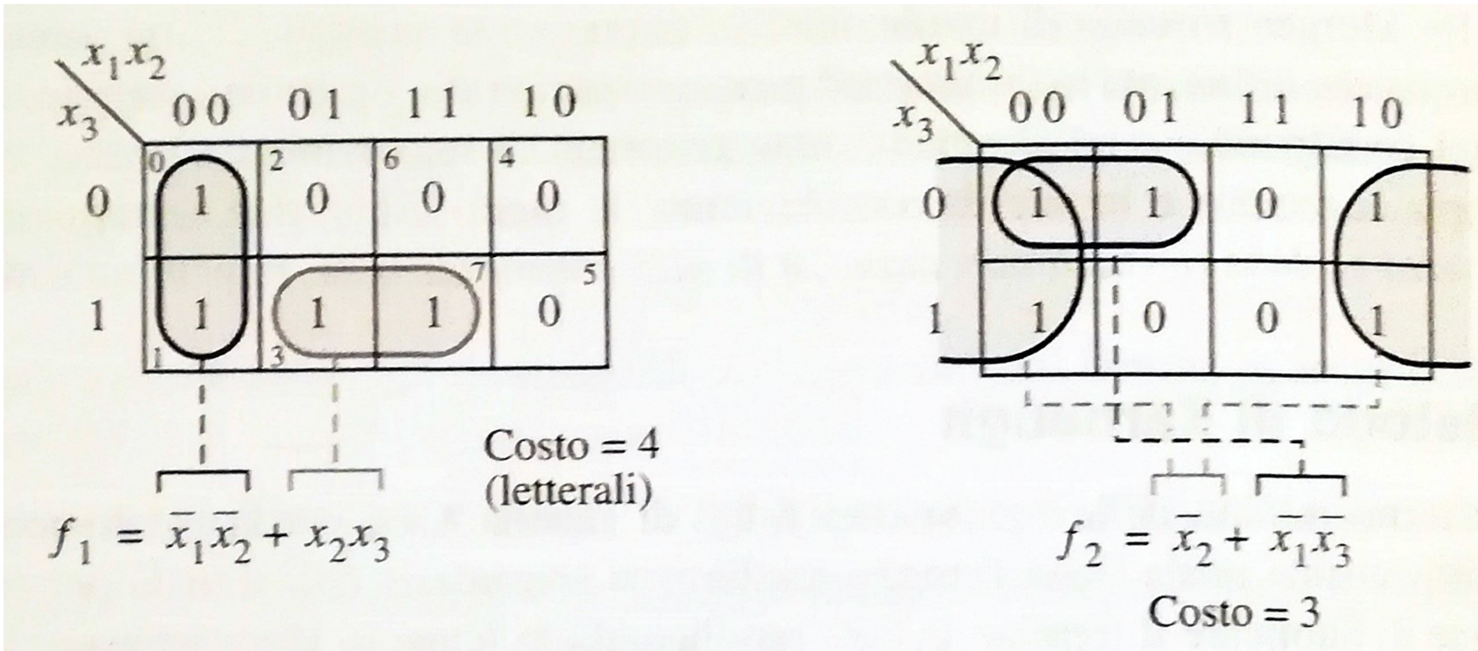
- Raggruppare le caselle di valore 1 **adiacenti orizzontalmente e verticalmente**

- Continuare a raggruppare fino a formare gruppi di grandezza massima di un **numero di caselle multiplo di 2** (2, 4, 8, ...)

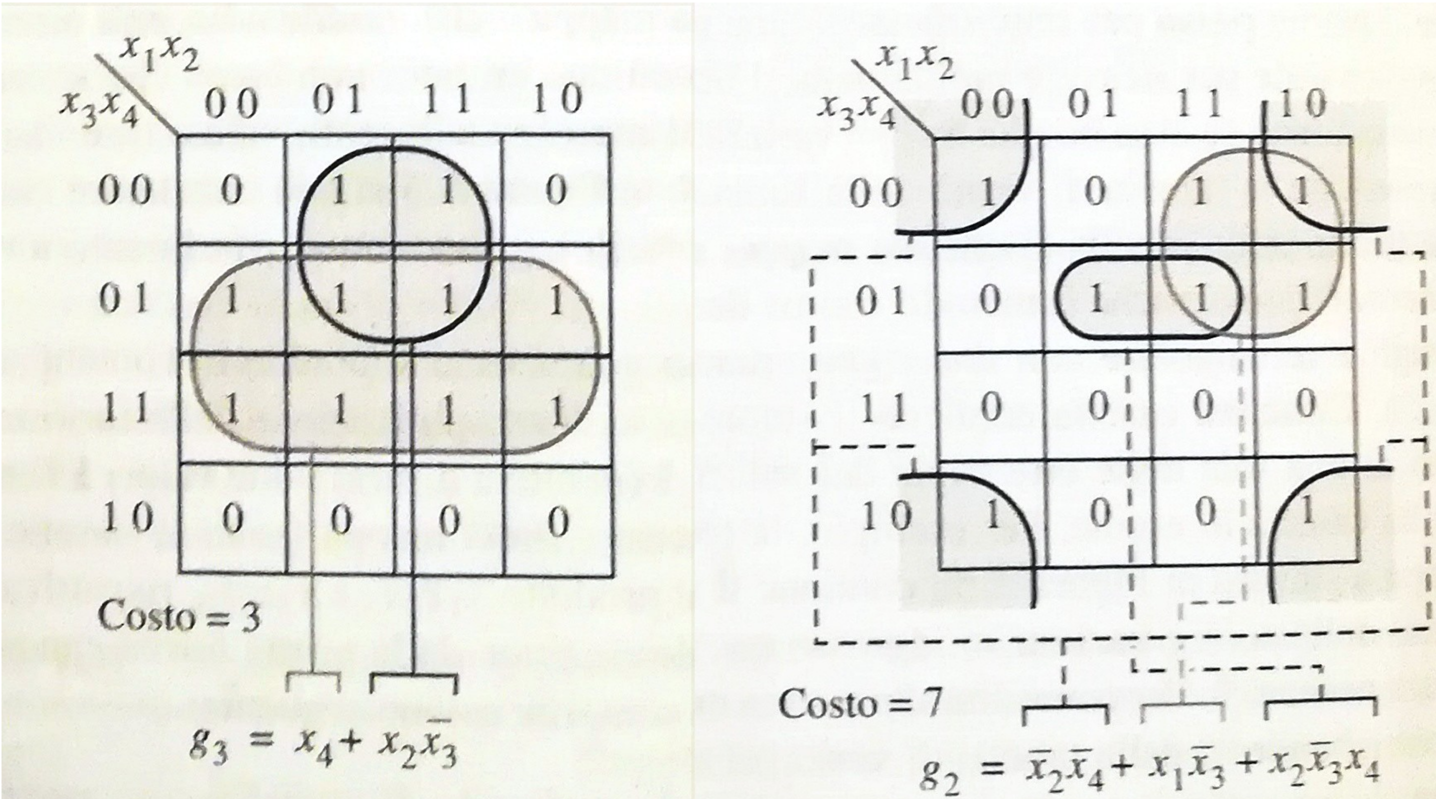
- Ogni gruppo rappresenta il **prodotto delle sue variabili con lo stesso valore** (forma diretta se 1 e negata se 0)

- Si ottiene un espressione SOP in forma minima dove ogni gruppo rappresenta uno dei prodotti dell'espressione

Esempi mappe a 3 variabili



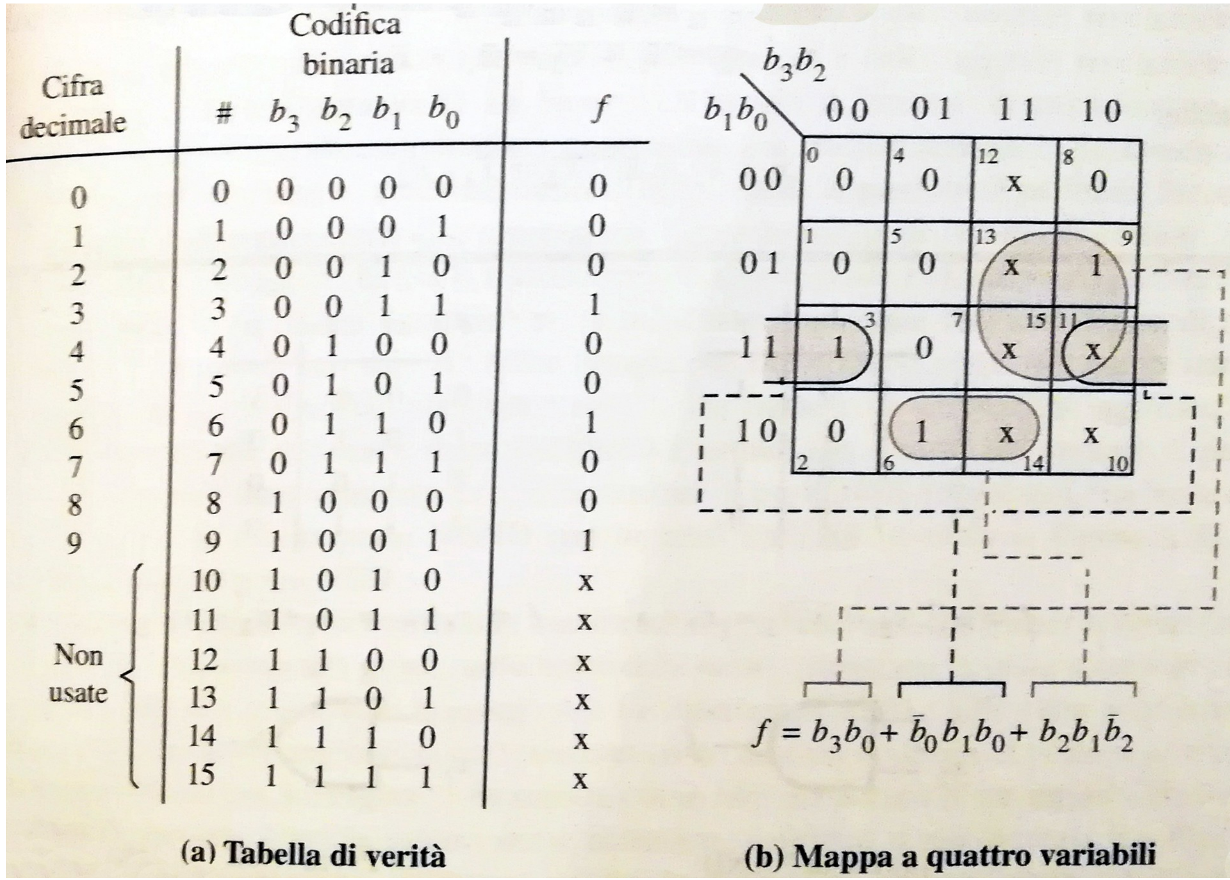
Esempi mappe a 4 variabili



Condizione di indifferenza

- Spesso capita che una funzione logica **non sia definita su tutte le combinazioni di valori delle sue variabili**
- Le variabili non usate si dice siano in **condizione di indifferenza** (don't care condition)
- Nella tabella di verità vengono indicate con il simbolo "X"
- Il loro valore (0 o 1) si può scegliere in modo da minimizzare il più possibile la forma minima (non sempre facile)

Condizione di indifferenza (esempio)



Cifra decimale	#	b_3	b_2	b_1	b_0	f
0	0	0	0	0	0	0
1	1	0	0	0	1	0
2	2	0	0	1	0	0
3	3	0	0	1	1	1
4	4	0	1	0	0	0
5	5	0	1	0	1	0
6	6	0	1	1	0	1
7	7	0	1	1	1	0
8	8	1	0	0	0	0
9	9	1	0	0	1	1
Non usate	10	1	0	1	0	x
	11	1	0	1	1	x
	12	1	1	0	0	x
	13	1	1	0	1	x
	14	1	1	1	0	x
	15	1	1	1	1	x

(Codifica binaria: b_3 b_2 b_1 b_0)

(a) Tabella di verità

b_1b_0 \ b_3b_2	00	01	11	10
00	0 (0)	0 (4)	x (12)	0 (8)
01	0 (1)	0 (5)	x (13)	1 (9)
11	1 (3)	0 (7)	x (15)	x (11)
10	0 (2)	1 (6)	x (14)	x (10)

$$f = b_3b_0 + \bar{b}_0 b_1 b_0 + b_2 b_1 \bar{b}_2$$

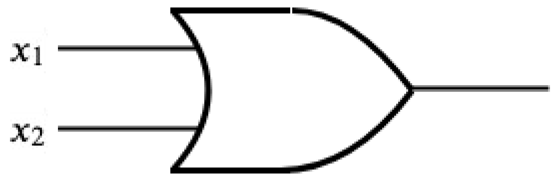
(b) Mappa a quattro variabili

Circuiti logici

- Le operazioni logiche base (**AND, OR, NOT**) possono essere realizzate da semplici **circuiti elettronici**
- Questi circuiti base vengono chiamati **PORTE**
- Una rete di porte logiche collegate tra loro è chiamata **RETE COMBINATORIA**
- Una rete combinatoria ha n ingressi binari ed m uscite binarie con n e $m \geq 1$

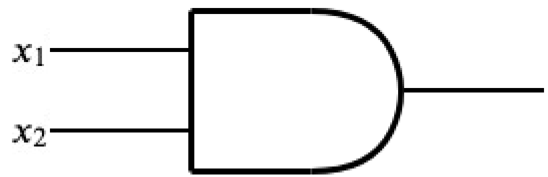
Le porte delle operazioni fondamentali possono essere **rappresentate graficamente**:

OR



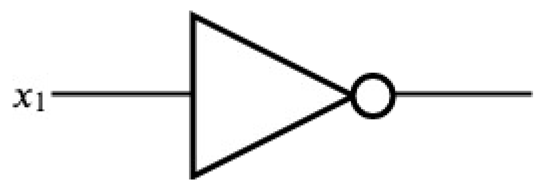
$f = x_1 + x_2$

AND



$f = x_1 \cdot x_2$

NOT



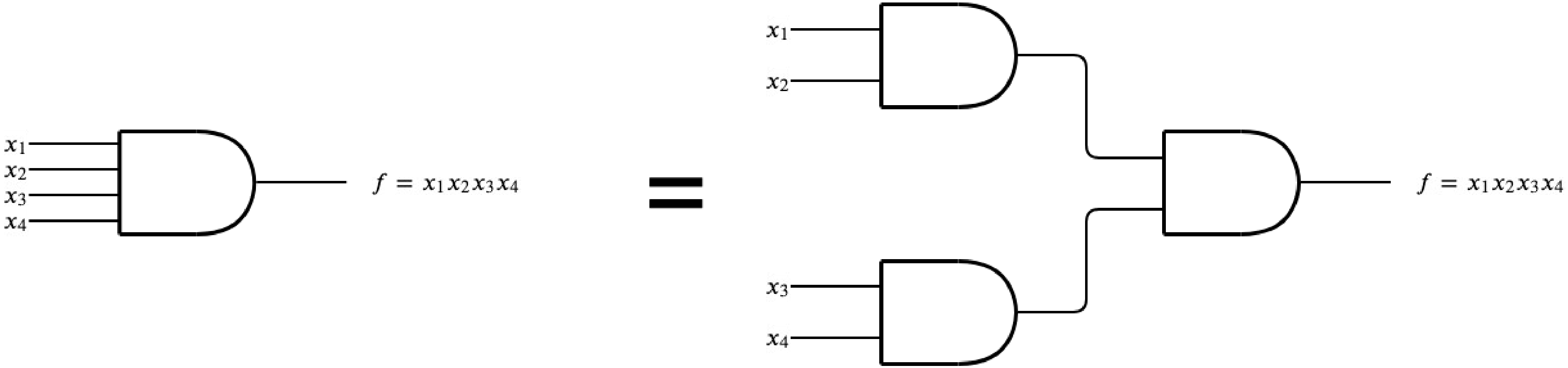
$f = \overline{x}_1$

Collegando entrate e uscite delle porte si possono rappresentare le reti combinatorie

Porte a più ingressi

- Grazie alla proprietà associativa AND e OR possono essere estese a più di 2 ingressi
- Graficamente rappresentati da una porta logica con più ingressi
- Equivale a mettere in due livelli a **cascata** o ad **albero** porte AND o OR a due ingressi

Esempio porta AND a 4 ingressi:

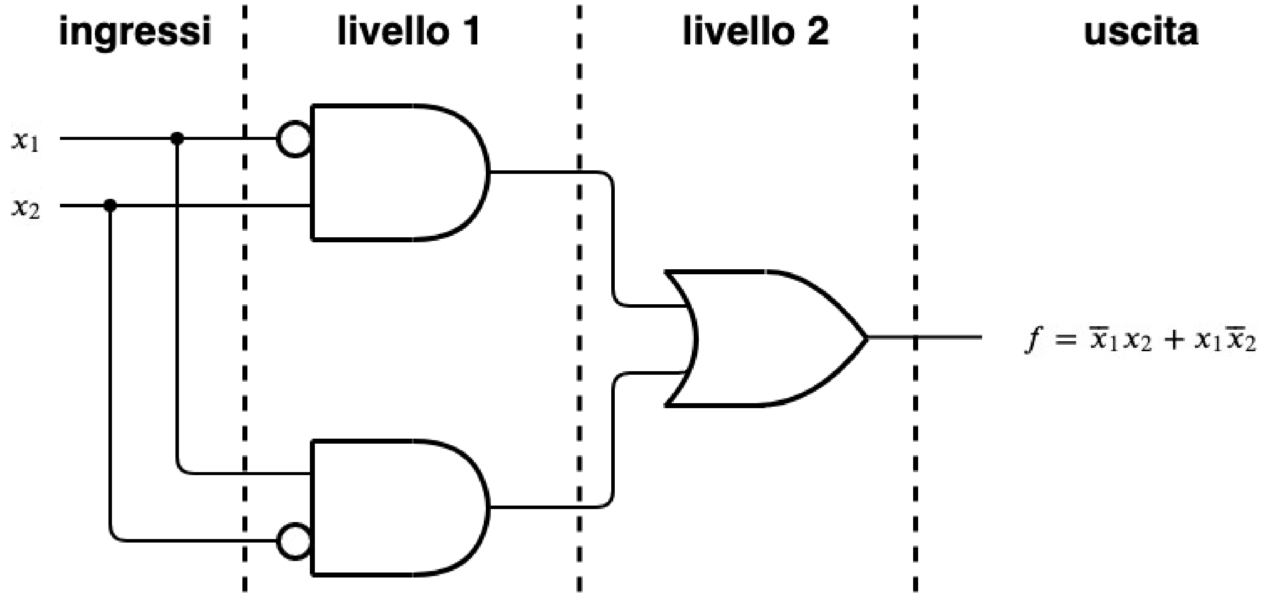


Equivalenza tra reti combinatorie e espressioni

Un espressione logica può essere rappresentata come rete combinatoria con:

- Una porta per ogni operatore logico presente nell’espressione
- Le porte collegate tra di loro ad albero seguendo i livelli di priorità nell’espressione

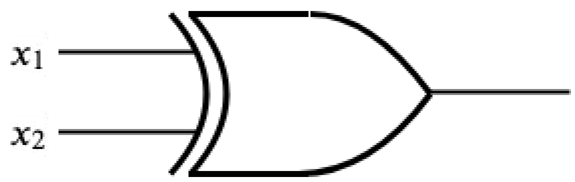
Le espressioni SOP e POS corrispondono a reti a due livelli:



Differenza simmetrica o XOR (OR esclusivo)

- La funzione che vale 1 solo se gli 1 nei suoi ingressi sono in numero dispari è chiamata **differenza simmetrica** o **XOR**
- Si denota tramite l'operatore a due argomenti "$\oplus$"

Lo **XOR** è rappresentato dalla seguente **porta logica**:



$$f = x_1 \oplus x_2 = \overline{x}_1 x_2 + x_1 \overline{x}_2$$

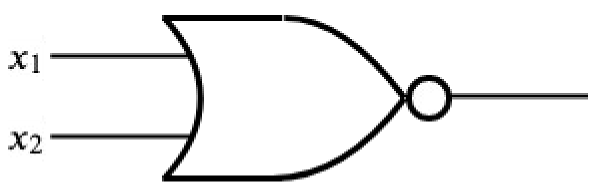
x_1	x_2	$f(x_1,x_2) = x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

NAND e NOR

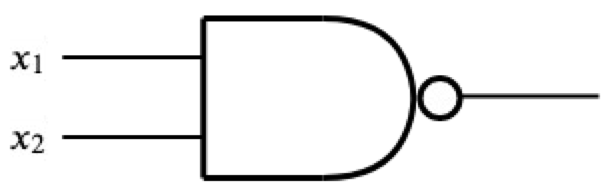
- Due funzioni di largo uso pratico sono l'AND negato (NAND) e l'OR negato (NOR)

- Si denotano tramite gli operatori a due argomenti:
- **NAND** "$\uparrow$"
- **NOR** "$\downarrow$"

Sono rappresentati dalle seguenti **porte logiche**:



$$f = x_1 \downarrow x_2 = \overline{x_1 + x_2} = \overline{x}_1 \overline{x}_2$$



$$f = x_1 \uparrow x_2 = \overline{x_1 x_2} = \overline{x}_1 + \overline{x}_2$$

x_1	x_2	$\neg(x_1 + x_2)$	$\neg(x_1 x_2)$
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0

Porte universali

- NAND e NOR sono porte **UNIVERSALI**: Si può realizzare una qualsiasi funzione combinatoria con reti logiche di soli NAND o soli NOR

- Grazie alle leggi di **De Morgan** e alla legge di **involuzione** possiamo passare da una **SOP ad una rete di NAND**:

$$
\begin{aligned}
(x_1 \uparrow x_2) \uparrow (x_3 \uparrow x_4) &= \overline{(\overline{x_1 \cdot x_2}) \cdot (\overline{x_3 \cdot x_4})} = \quad (De\ Morgan) \\
&= \overline{\overline{x_1 x_2}} + \overline{\overline{x_3 x_4}} = \quad (Involuzione) \\
&= x_1 x_2 + x_3 x_4 \quad (Sum\ of\ Products)
\end{aligned}
$$

SOP come rete di NAND

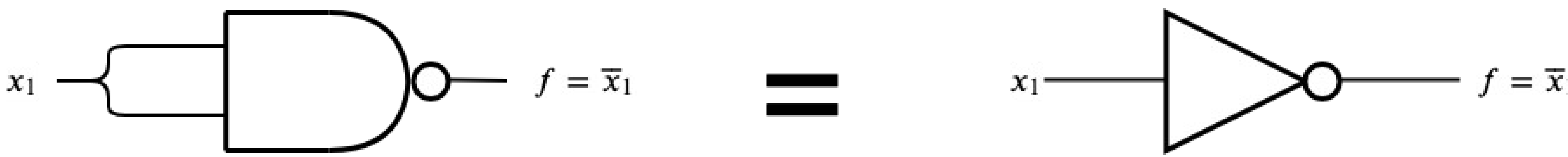
- È possibile trasformare un'espressione **SOP di porte binarie** in una **rete di NAND** in questo modo:
  - Cambiare tutte le porte **AND e OR** con porte **NAND**
  - **Mantenere gli ordini di priorità** tra le operazioni dell'espressione di partenza

- Per essere in grado di realizzare con porte NAND **qualsiasi espressione in forma SOP** ci manca sapere:
  - Come rendere una porta NOT con porte NAND
  - Come realizzare le porte NAND a più ingressi

NOT come circuito di NAND

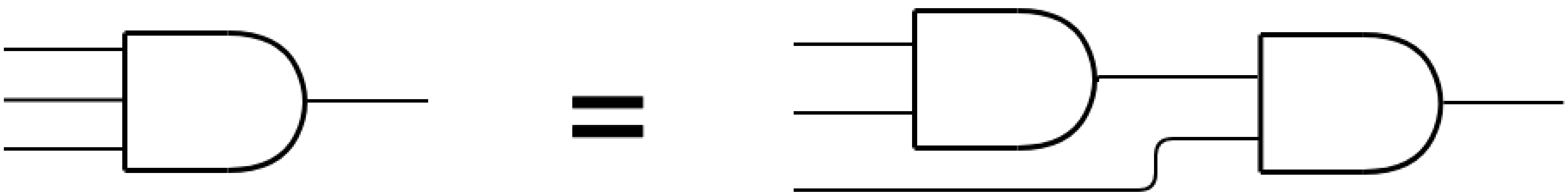
- Una porta **NAND con ingressi unificati** si comporta come una porta **NOT** negando la variabile di ingresso

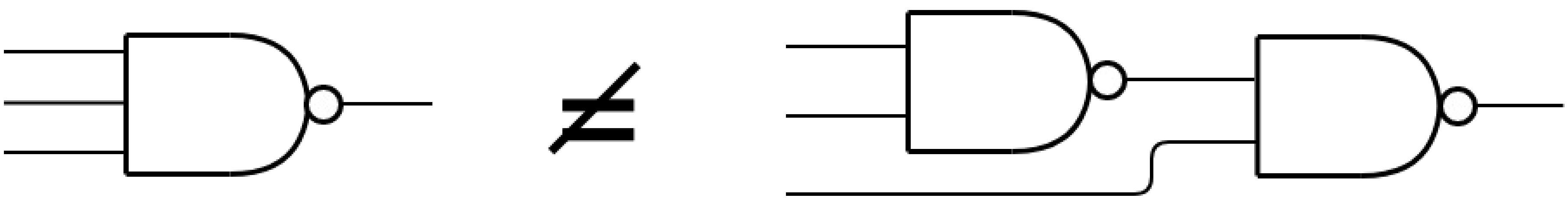
- Lo stesso vale per la porta NOR



Porte NAND a più ingressi

- Le operazioni NAND e NOR godono della proprietà **commutativa**
- Sfortunatamente **non godono della proprietà associativa**





Porte NAND a più ingressi

Grazie alla **legge di involuzione** e la **rappresentazione della porta NOR come NAND** possiamo rappresentare una porta NAND a più ingressi