LABORATORIO DI ARCHITETTURA DEI CALCOLATORI
lezione n° 2

Prof. Rosario Cerbone

rosario.cerbone@uniparhenope.it

http://digilander.libero.it/rosario.cerbone

a.a. 2008-2009

Riepilogo

- In questa lezione vengono riassunti i concetti fondamentali sulle funzioni booleane e loro minimizzazione con l'utilizzo delle mappe di Karnaugh

Funzioni booleane

- Ciascuna variabile booleana può assumere uno dei due stati '1' o '0'. Due variabili prese insieme possono individuare $2^2 = 4$ stati. La variabile *A* può essere presente come *A* o !*A*. Una seconda variabile *B* può anch'essa essere presente come *B* o !*B* . Queste due variabili prese insieme possono dare luogo a
- quattro combinazioni: *AB*, *A!B* , !*AB*, !*A!B*. Assegnando 1 alla variabile vera e 0 alla variabile complementata possiamo riscrivere i quattro stati come 11,10,01,00.
- Tre variabili possono essere scritte in 2^3 differenti combinazioni, dando luogo a 2^3 stati;
- *n* variabili danno luogo a 2^n stati.

Tabella delle combinazioni

- La tabella delle combinazioni è un'elencazione sistematica di tutte le combinazioni che un gruppo di variabili binarie può assumere, ordinate secondo la sequenza numerica binaria. Date tre variabili *A*, *B*, *C*, sono possibili $2^3 = 8$ differenti combinazioni di queste variabili prese insieme.
- Di seguito è mostrata la tabella delle combinazioni delle tre variabili *A*, *B*, *C*.
- *A B C* — *Comb*
- 0 0 0 — *!A !B !C*
- 0 0 1 — *!A !B C*
- 0 1 0 — *!A B !C*
- 0 1 1 — *!A B C*
- 1 0 0 — *A !B !C*
- 1 0 1 — *A !B C*
- 1 1 0 — *A B !C*
- 1 1 1 — *A B C*

Tabella della verità

- Una funzione booleana viene univocamente definita dalla sua tabella della verità. La tabella della verità è un'estensione della tabella delle combinazioni: a questa viene aggiunta sulla destra una colonna nella quale è indicato lo stato che la funzione assume in corrispondenza di ogni combinazione delle variabili, per tutte le combinazioni.
- Di seguito è mostrata la tabella della verità di una funzione *f* di tre variabili:
- *A B C* *f*
- 0 0 0 0
- 0 0 1 1
- 0 1 0 0
- 0 1 1 1
- 1 0 0 0
- 1 0 1 1
- 1 1 0 1
- 1 1 1 0

Funzioni booleane

- E' immediato scrivere l'espressione booleana della funzione *f* partendo dalla tabella della verità. La funzione è data dalla OR delle combinazioni (somma canonica) per le quali *f* = 1:

  f = !*A* !*B* *C* + !*A* *B* *C* + *A* !*B* *C* + *A* *B* !*C*.
- Il metodo appena illustrato mostra come si giunge alla sintesi delle funzioni booleane. L'espressione appena scritta è l'espressione primitiva della funzione *f*. Applicando i teoremi dell'algebra booleana è possibile scrivere l'espressione più semplice della funzione *f*.
- I mintermini di una funzione vengono spesso identificati con il numero in base 10 corrispondente al valore binario del mintermine. (Es., a!b!c = m_4) e la funzione indicata come somma dei mintermini $f=\Sigma\, m_i$

Mappe di Karnaugh

- Una più eloquente ed utile rappresentazione di una funzione booleana è data dalla mappa di Karnaugh.
- Tutte le possibili combinazioni che un gruppo di variabili può assumere sono rappresentate in forma di matrice nella mappa.

Mappe di Karnaugh

- La mappa di Karnaugh può essere usata per definire una funzione. In ogni cella corrispondente alla combinazione delle variabili per cui la funzione è vera si pone '1'; dove la funzione è falsa si pone '0'. Essa da una definizione del tutto equivalente a quella data dalla tabella della verità.

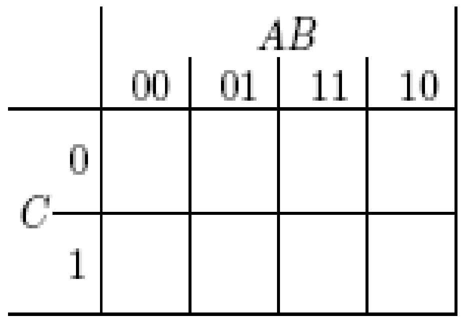
Minimizzazione delle funzioni booleane

- Si può effettuare la minimizzazione delle funzioni booleane mediante manipolazioni algebriche, utilizzando i teoremi dell'algebra di Boole, oppure mediante l'elaborazione delle mappe di Karnaugh.

Minimizzazione delle funzioni booleane

- **La mappa di Karnaugh** è una disposizione ordinata di celle, che contengono le combinazioni delle variabili in modo che nel passare da una cella ad una contigua cambi lo stato di una sola variabile.
- La mappa contiene una cella per ogni combinazione delle variabili, in modo da esaurire tutte le combinazioni possibili.
- Una mappa di 2 variabili deve contenere 4 celle, perché vi sono 2^2 combinazioni differenti delle due variabili. Una mappa di tre variabili deve contenere 2^3 celle; una mappa di *n* variabili deve contenere 2^n celle.
- Raffiguriamo la mappa di Karnaugh di tre variabili.

Minimizzazione delle funzioni booleane



Minimizzazione delle funzioni booleane

- Raffiguriamo il diagramma della mappa di Karnaugh per quattro variabili.
- Le variabili sono identificate sopra e a lato del diagramma. Le combinazioni delle variabili *A* e *B,* sopra in orizzontale, e delle variabili *C* e *D,* lateralmente in verticale, sono disposte secondo il codice di Gray di due variabili.

Minimizzazione delle funzioni booleane

$CD \backslash AB$	00	01	11	10
00				
01				
11				
10				

Minimizzazione delle funzioni booleane

- Consideriamo ora la struttura delle mappe di quattro variabili. Avendo ordinato le combinazioni, contenute nelle celle, secondo il codice di Gray, che è ciclico, i due bordi superiore ed inferiore della tabella risultano adiacenti e una sola variabile cambia stato nell'attraversamento del bordo. Pertanto a questo punto possiamo considerare la mappa come cilindrica, con i due bordi superiore ed inferiore coincidenti.
- Una considerazione analoga vale per i bordi destro e sinistro: anch'essi sono coincidenti. Ecco allora che la mappa di Karnaugh, che disegnamo in forma di matrice piana, è in realtà in forma di una superficie toroidale, senza bordi. Di conseguenza, qualsiasi cella ha una cella contigua su ognuno dei suoi quattro lati.

Meccanismo della semplificazione

- Si parte scrivendo la tabella della verità della funzione. Da essa si ricava quali sono le combinazioni vere e si pone un '1' nelle celle della mappa corrispondenti alle combinazioni vere. Ogni '1' collocato nella mappa corrisponde ad una combinazione presente nella somma canonica, espressione della funzione.
- Per come è stata costruita la mappa, a due '1' collocati in celle contigue corrispondono combinazioni che differiscono soltanto in una variabile: le rispettive combinazioni nella somma canonica si sommano secondo il teorema (in forma generalizzata)

$$term^*Y + term^*\,!Y = term\ ^*(Y + !Y) = term.$$

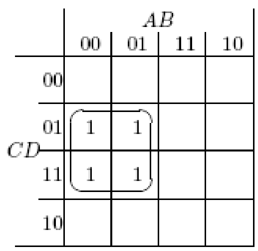
- La semplificazione delle funzioni avviene attraverso l'applicazione ripetuta del suddetto teorema. Inoltre le ridondanze sono automaticamente eliminate.

Meccanismo della semplificazione

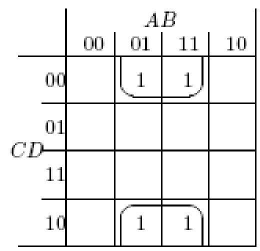
- Elenchiamo qui di seguito le regole da seguire per individuare i gruppi di celle rilevanti per costruire l'espressione semplificata di una funzione.
- \- i gruppi possono contenere 1, 2, 4, 8 o in generale 2^n celle
- \- i gruppi non possono includere celle contenenti uno '0'
- \- i gruppi possono essere orizzontali o verticali, ma non diagonali: i gruppi sono quindi in forma di rettangoli o di quadrati
- \- ogni gruppo deve essere il più largo possibile, cioè deve contenere quanti più '1' possibile
- \- ogni cella contenente un '1' deve appartenere ad almeno un gruppo
- \- i gruppi si possono sovrapporre
- \- le celle che si trovano sui bordi possono venir raggruppate con quelle corrispondenti dal lato opposto (ricordiamoci del "toroide")
- \- i gruppi devono essere nel minor numero possibile senza contraddire alcuna delle regole elencate precedentemente.

Meccanismo della semplificazione

- Seguono alcuni esempi di semplificazione con mappe di quattro variabili.



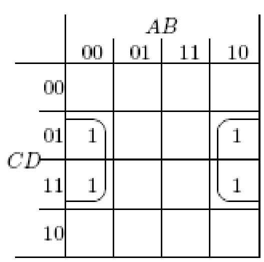
$\bar{A}D$



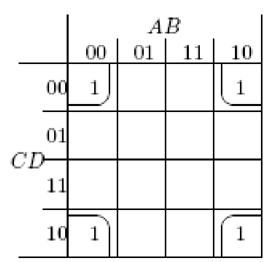
$B\bar{D}$

Meccanismo della semplificazione

- Seguono alcuni esempi di semplificazione con mappe di quattro variabili.



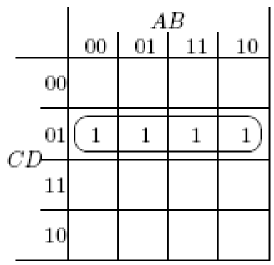
$\bar{B}D$



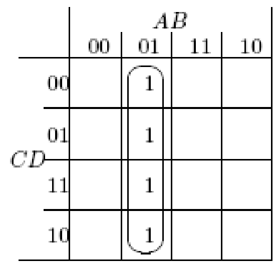
$\bar{B}\bar{D}$

Meccanismo della semplificazione

- Seguono alcuni esempi di semplificazione con mappe di quattro variabili.



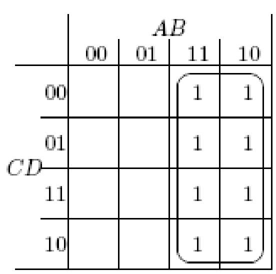
$\bar{C}D$



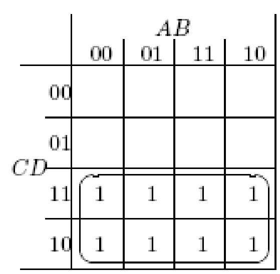
$\bar{A}B$

Meccanismo della semplificazione

- Seguono alcuni esempi di semplificazione con mappe di quattro variabili.



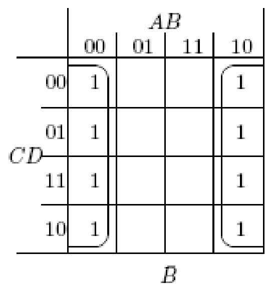
A

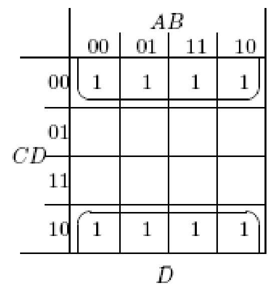


C

Meccanismo della semplificazione

- Seguono alcuni esempi di semplificazione con mappe di quattro variabili.





Condizioni di indifferenza e ridondanze

- Può accadere che nella tabella della verità di una funzione compaiano condizioni in cui per una qualche combinazione il valore della funzione può essere indifferentemente '1' o '0'. Questa condizione viene indicata nella tabella con il simbolo Φ. Tale simbolo appare anche nella mappa di Karnaugh.
- Può accadere inoltre che certe combinazioni non si verifichino mai (ad es., quando usiamo il codice BCD sei delle sedici combinazioni non si verificano mai): queste combinazioni sono dette ridondanze.
- Nella tabella della verità in corrispondenza della ridondanza viene posto il simbolo X. Tale simbolo appare anche nella mappa di Karnaugh.

Condizioni di indifferenza e ridondanze

- Sia la condizione di indifferenza sia la ridondanza sono utili nella minimizzazione delle espressioni delle funzioni. Basta scegliere uguali a '1' le condizioni di indifferenza che consentono una semplificazione.
- Le ridondanze non si verificano mai, quindi possiamo attribuire loro, dove è utile, il valore fittizio '1' unicamente per consentire la semplificazione.

Condizioni di indifferenza e ridondanze

- La tabella della verità seguente illustra le considerazioni fatte.
- $A\ B\ C\ f_1\ f_2$
- 0 0 0 0 Φ
- 0 0 1 1 Φ
- 0 1 0 Φ 0
- 0 1 1 Φ 1
- 1 0 0 0 Φ
- 1 0 1 0 Φ
- 1 1 0 X X
- 1 1 1 X X

Condizioni di indifferenza e ridondanze

- Disegniamo le mappe di Karnaugh per f_1 e f_2 e ricaviamo le funzioni semplificate.

		AB			
		00	01	11	10
C	0		ϕ	$\times$	
	1	1	ϕ	$\times$	

f_1

		AB			
		00	01	11	10
C	0	ϕ		$\times$	ϕ
	1	ϕ	1	$\times$	ϕ

f_2

Condizioni di indifferenza e ridondanze

- Risultano:
- f_1 = !A C; f_2 = C.
- Non considerando le condizioni di indifferenza e le ridondanze si avrebbe:
- f_1 = !A !BC; f_2 = !ABC.

Logica Combinatoria

- una rete combinatoria è un circuito logico avente n ingressi (x_1,x_2,…,x_n) ed m uscite (y_1,y_2, …,y_m), ciascuno dei quali assume valori binari (0/1), tale che a ciascuna combinazione degli ingressi corrisponde un'unica combinazione delle uscite.
- da un punto di vista logico, ogni uscita può essere definita come una funzione booleana degli ingressi $y_i = y_i(x_1,x_2,\ldots,x_n)$.
- ad ogni istante, il valore delle uscite dipende unicamente dal valore assunto dagli ingressi nello stesso istante.

Logica Combinatoria

- La procedura per progettare una rete logica combinatoria passa attraverso i seguenti stadi:
- 1: definizione completa e univoca del problema da risolvere
- 2: analisi del problema, con individuazione delle variabili d'ingresso e delle funzioni di uscita
- 3: scrittura della tabella della verità di ogni funzione
- 4: sintesi delle funzioni e loro semplificazione con le mappe di Karnaugh
- 5: disegno della schema logico della rete.

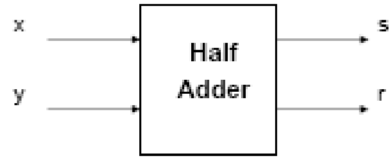
Blocchi logici funzionali

- Diamo qualche esempio di blocco logico funzionale, elemento costitutivo di schemi logici complessi.

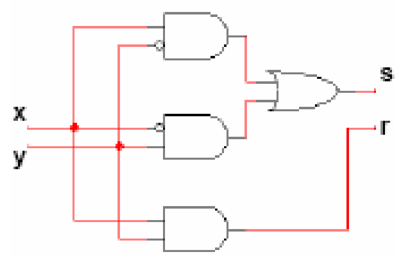
Sommatore

- Esegue l'addizione di cifre binarie fornendo in uscita la cifra somma e la cifra riporto. Sono possibili due schemi:
- – semiaddizionatore (half adder)
  - **2 cifre in ingresso**
- – addizionatore completo (full adder)
  - **2 cifre in ingresso + carry in ingresso**

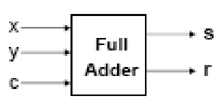
Half adder



x	y	s	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Full Adder



x	y	c	s	r
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

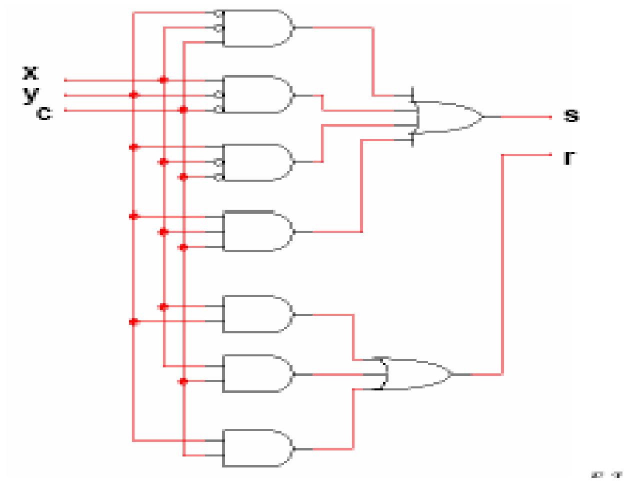
c \ xy	00	01	11	10
0		1		1
1	1		1	

s = !x!yc+!xy!c+xyc+x!y!c

c \ xy	00	01	11	10
0			1	
1		1	1	1

r = xy+yc+xc

Full Adder – sintesi diretta



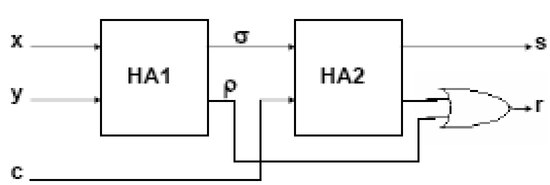
Full Adder – sintesi per decomposizione

s = !x!yc+!xy!c+xyc+x!y!c = (!x!y+xy)c+(!xy+x!y)!c

r = xy+yc+xc = xy+!xyc+xyc+xyc+x!yc = xy+(!xy+x!y)c

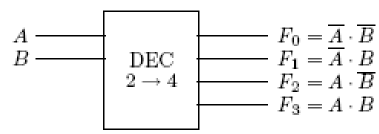
σ = !xy+x!y $\Rightarrow$ s = !σc + σ!c

ρ = xy   r = ρ + σc



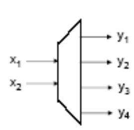
Il decoder binario

- Un decoder binario ha tante uscite quante sono le combinazione delle variabili d'ingresso.
- E' fatto in modo che sia attiva la sola uscita che corrisponde alla combinazione presente in ingresso.



Il decoder binario

- Rete combinatoria ad n ingressi ed a 2^n uscite. Per ogni combinazione degli ingressi, solo una uscita assume valore 1 mentre le altre sono uguali a 0.



decodificatore 1/4

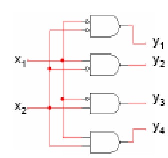
x_1	x_2	y_1	y_2	y_3	y_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

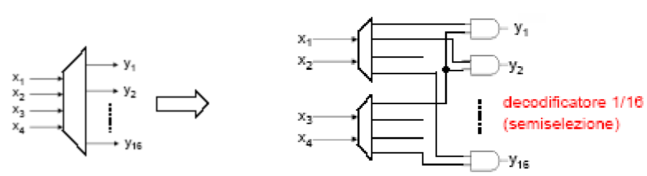
$y_1 = !x_1!x_2$

$y_2 = !x_1x_2$

$y_3 = x_1!x_2$

$y_4 = x_1x_2$



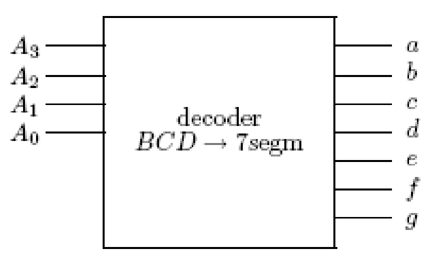


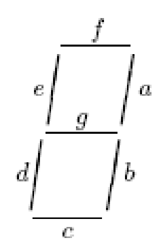
Il decoder BCD-decimale

- Esercizio:
- Scrivere la tabella della verità di un decoder BCD-decimale
- Minimizzare la funzione logica con le mappe di Karnaugh senza considerare ridondanze e indifferenze
- Come al punto 2 ma considerando le ridondanze e le indifferenze.
- Confrontare i risultati
- Eseguire la minimizzazione con SIS e confrontare i risultati con quanto ottenuto ai punti precedenti

Il decoder BCD - 7 segmenti

- Rete combinatoria ad 4 ingressi ed a 7 uscite. Serve per pilotare display numerici a 7 segmenti.

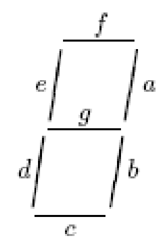




Il decoder BCD - 7 segmenti

- Tabella della verità

	A_3	A_2	A_1	A_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	1	1	0	0	0	0	0
2	0	0	1	0	1	0	1	1	0	1	1
3	0	0	1	1	1	1	1	0	0	1	1
4	0	1	0	0	1	1	0	0	1	0	1
5	0	1	0	1	0	1	1	0	1	1	1
6	0	1	1	0	0	1	1	1	1	1	1
7	0	1	1	1	1	1	0	0	0	1	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	0	0	1	1	1



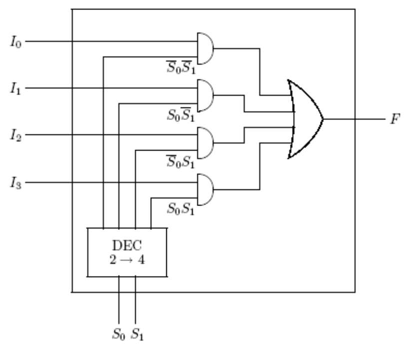
Il decoder BCD - 7 segmenti

- Esercizio:

1. Eseguire la minimizzazione con le mappe di Karnaugh sfruttando le ridondanze e le indifferenze.
2. Eseguire la minimizzazione con SIS.
3. Confrontare i risultati

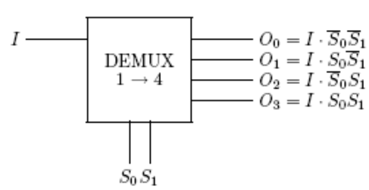
Il multiplexer binario

- Il multiplexer è la realizzazione interamente elettronica di un commutatore meccanico.
- Esso permette di selezionare un ingresso fra tanti e di inviare in uscita il suo stato. Il multiplexer si avvale di un decoder per effettuare la selezione degli ingressi. A fianco è rappresentato, in forma di blocco funzionale, un multiplexer a 4 ingressi (4 a 1).



Il demultiplexer

- Il demultiplexer svolge la funzione opposta rispetto al multiplexer: invia il segnale in ingresso su una delle possibili uscite selezionate da un decoder. A fianco è dato il blocco logico funzionale del demultiplexer 1 a 4 e sono scritte le espressioni logiche delle uscite.



Esercizio 2.1

- In un consiglio di amministrazione di quindici soci è richiesto un voto con maggioranza di 2/3 per approvare spese superiori a 100.000 €. I voti favorevoli sono conteggiati e convertiti in codice binario. Progettare e minimizzare un circuito che abbia uscita 1 se la maggioranza ha votato a favore della delibera.

Esercizio 2.2a

- Progettare e minimizzare un circuito che risolva il problema seguente:
- Una industria dolciaria ha stabilito di vendere le proprie gomme da masticare con un distributore automatico.
- Il prezzo di una confezione è di 0,20 €. Il distributore accetta monete da 10, 20 o 50 centesimi.
- Ogni moneta accettata aziona uno dei quattro interruttori che indicano la presenza di una moneta. Vi sono 2 interruttori riservati alle monete di 10 cent e si suppone che la prima moneta da 10 azioni sempre il primo interruttore e la seconda da 10 azioni sempre il secondo interruttore. Un terzo interruttore è riservato ai 20 centesimi e un quarto a 50 cent.
- Sono previsti 2 segnali di uscita:
  1. Comando per l'erogazione di una confezione di gomme quando sono stati inseriti almeno 20 cent.
  2. Indicazione del resto

Esercizio 2.2b

- Modificare il circuito precedente aggiungendo due uscite per l'erogazione del resto.
- Il resto può essere costituito da una moneta da 10, una moneta da 20 o da entrambe.
- Se un acquirente, per errore, inserisce 10 cent seguite da 50 cent, perde la prima moneta.