LABORATORIO DI ARCHITETTURA DEI CALCOLATORI

lezione n° 2

Prof. Rosario Cerbone

rosario.cerbone@libero.it

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

a.a. 2006-2007

Richiami di Algebra di Commutazione

- In questa lezione vengono richiamati i concetti fondamentali di Algebra di commutazione

Algebra di commutazione

- Il comportamento di un dispositivo digitale può essere rappresentato per mezzo di una trasformazione di numeri binari in numeri binari ottenibile applicando un insieme di operatori.
- A tale scopo è necessario definire un modello matematico che permetta di rappresentare insiemi di numeri binari e funzioni che li mettono in relazione.
- Tale modello è chiamato *Algebra di Commutazione* ed è definito come segue.

Algebra di commutazione

Un'***Algebra Booleana*** è una quintupla

$$B = (A, +, *, 0, 1)$$

dove:

- A è l'alfabeto, ovvero l'insieme su cui vengono eseguite le operazioni;
- $+$ e $*$ sono due operazioni che agiscono sugli elementi di A;
- 0 è l'elemento neutro per $+$;
- 1 è l'elemento neutro per $*$.

Algebra di commutazione

- Un'***Algebra di Commutazione*** è un Algebra Booleana dove l'alfabeto *A* è composto solo dai due elementi *0* e *1*.
- L'operazione *,detta *AND*, vale *1* se e solo se applicata a due valori uguali a *1*, altrimenti vale *0*.
- L'operazione *+*, detta *OR*, vale *0* se applicata a due valori uguale a *0*, altrimenti vale *1*.
- Oltre ad *AND* e *OR* è definita l'operazione *NOT*, che vale *1* se applicata al valore *0* e *0* se applicata al valore *1*.

Algebra di commutazione

- Una generica ***funzione*** *dell'Algebra di Commutazione* (*booleana*) può essere vista come una mappa

$$f: B^n \rightarrow B^m$$

  che trasforma numeri binari a *n* bit in numeri binari a *m* bit.
- Una funzione booleana è *completamente specificata* se il valore dell'uscita è specificato per ogni valore degli ingressi.
- Una ***variabile*** *di commutazione* è una variabile *x* che può assumere solo uno dei valori di *A* in modo che:
  - $x = 0$ se e solo se $x \neq 1$;
  - $x = 1$ se e solo se $x \neq 0$.

Algebra di commutazione

- Le operazioni *AND*, *OR* e *NOT* sono solitamente denotate utilizzando la seguente notazione:
- *z = AND (a ,b)* diventa *z = a * b*
- *z = OR (a ,b)* diventa *z = a + b*
- *z = NOT (ab)* diventa *z =* ā

Tabelle delle verità

- Una funzione booleana può essere descritta per mezzo di una ***tabella delle verità*** che assegna ad ogni combinazione dei valori di input i corrispondenti valori agli output.
- Per ogni funzione esiste un'unica tabella delle verità che la rappresenta e viceversa.
- Tuttavia per ogni funzione esistono infinite espressioni per rappresentarla.

Tabelle delle verità

- Le seguenti sono le tabelle di verità per le operazioni elementari e per altre funzioni particolarmente usate per descrivere circuiti digitali:
  - AND
  - OR
  - NOT
  - XOR
  - NAND
  - NOR
  - XNOR
- Gli insiemi di operatori (*AND, OR, NOT*), (*AND, NOT*), (*OR, NOT*), (*NAND*), (*NOR*) sono *funzionalmente completi* ovvero permettono di realizzare qualsiasi funzione booleana.

Teoremi e identità degli operatori elementari

- Gli operatori elementari *AND* e *OR* sono caratterizzati dalla seguente serie di identità e teoremi che possono essere dimostrati per induzione matematica:

Teoremi e identità degli operatori elementari

	AND	*OR*
Identità	$1 * x = x$	$0 + x = x$
Elemento nullo	$0 * x = 0$	$1 + x = 1$
Idempotenza	$x * x = x$	$x + x = x$
Inverso	$x * \bar{x} = 0$	$x + \bar{x} = 1$
Commutativa	$x * y = y * x$	$x + y = y + x$
Associativa	$(x * y) * z = x * (y * z)$	$(x + y) + z = x + (y + z)$
Assorbimento	$x * (x + y) = x$	$x + (x * y) = x$
Distributiva	$x * (y + z) = x * y + x * z$	$x + (y * z) = (x + y) * (x + z)$
De Morgan	$\overline{x * y} = \bar{x} + \bar{y}$	$\overline{x + y} = \bar{x} * \bar{y}$

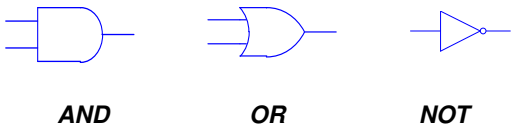
Teoremi e identità degli operatori elementari

- Ai teoremi riportati nella tabella precedente va aggiunto il seguente *teorema di Shannon*:
- $f(x_1 \ldots x_n) = x_1 * f(1,x_2,\ldots,x_n) + \overline{x}_1 * f(0,x_2,\ldots,x_n)$
- $f(x_1 \ldots x_n) = (x_1 + f(0,x_2,\ldots,x_n)) * (\overline{x}_1 + f(1,x_2,\ldots,x_n))$

Porte logiche

- Data una qualunque funzione booleana è possibile descrivere il ***circuito combinatorio*** che essa rappresenta per mezzo di un insieme di ***porte logiche***.
- Ogni porta logica corrisponde ad un operatore booleano e viene realizzata connettendo opportunamente dispositivi elettronici chiamati *transistor*.
- Le porte logiche rappresentano i valori *0* e *1* come una differenza di potenziale
  - 0 Volt per *0*
  - 3 Volt per *1*

Porte logiche



AND ***OR*** ***NOT***

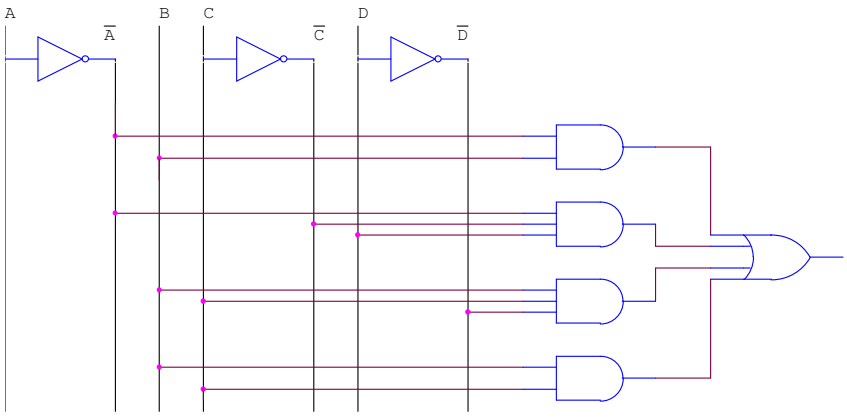
NAND ***NOR*** ***XOR*** ***XNOR***

Porte logiche

- Collegando opportunamente porte logiche è possibile realizzare qualunque funzione booleana. Ad esempio la funzione:
- $L = \overline{A}*B + \overline{A}*\overline{C}*D + B*C*\overline{D} + B*C$
- può essere realizzata dal seguente circuito:

Porte logiche

$L = \overline{A}*B + \overline{A}*\overline{C}*D + B*C*\overline{D} + B*C$



Forme canoniche

- Ogni funzione booleana può essere rappresentata mediante due distinte forme canoniche. Le seguenti definizioni sono necessarie per comprendere appieno il significato delle forme canoniche.
- Data una funzione booleana $y = f(x_1 \ldots x_n)$:
- - Un ***letterale*** è una coppia (variabile, valore). Ogni variabile *x* ha 2 letterali (*x, 0*) e (*x, 1*) che vengono indicati rispettivamente come $\bar{x}$ e *x*.
- - Un ***implicante*** di *f* è un prodotto di letterali $P = x_i * \ldots * x_j$, dove $1 \leq i, j \leq n$, tale che se $P = 1$ allora $f = 1$.
- - Un ***mintermine*** di *f* è un implicante in cui tutte le variabili compaiono come letterale.
- - L'***On-set*** di *f* è l'insieme dei mintermini. Dal punto di vista della tabella delle verità, l'on-set di *f* è l'insieme delle possibili combinazioni di valori di ingresso tali per cui z = 1.

Forme canoniche

- - Un ***maxtermine*** di *f* è un punto dello spazio B*n* dove la funzione vale 0.
- - L'***Off-set*** di *f* è l'insieme dei maxtermini. Dal punto di vista della tabella delle verità, l'off-set di *f* è l'insieme delle possibili combinazioni di valori di ingresso tali per cui z = 0.
- - La ***copertura*** di una funzione *f* è un insieme di implicanti che coprono tutti i mintermini (o i maxtermini) della funzione.

Forme canoniche

- **La *1° forma canonica di copertura di una funzione f***, detta *somma di prodotti*, è data da:

$$f(x_1 \ldots x_n) = m_1 + \ldots + m_k$$

- dove m_j sono i mintermini contenuti nell'on-set di *f*.
- Ogni mintermine è realizzato da una porta *AND* a *n* ingressi.
- Le uscite delle porte che rappresentano i mintermini sono collegate tramite una porta *OR* a *k* ingressi.

Forme canoniche

- **La *2° forma canonica di copertura di una funzione f***, detta *prodotto di somme*, è data da:

$$f(x_1 \ldots x_n) = M_1 * \ldots * M_k$$

- dove M_j sono i maxtermini contenuti nell'off-set di *f*.
- Ogni maxtermine è realizzato da una porta *OR* a *n* ingressi. Le uscite delle porte che rappresentano i maxtermini sono collegate tramite una porta *AND* a *k* ingressi.

Forme canoniche

- I circuiti in somma di prodotti o prodotto di somme sono circuiti a 2 livelli. Il numero di livelli si calcola in base al numero di porte logiche che i segnali in ingresso devono attraversare per propagare i corrispondenti valori alle uscite. In questo calcolo non vengono conteggiate le eventuali porte *NOT*.

Esercizio

- Per frequentare un certo corso di elettronica uno studente deve soddisfare le seguenti condizioni:
- 1_ aver superato almeno 20 esami **ed** essere uno studente di ingegneria in corso, **oppure**
- 2_ aver superato almeno 20 esami **ed** essere uno studente di ingegneria con il piano di studio approvato, **oppure**
- 3_ aver superato meno di 20 esami **ed** essere uno studente di ingegneria fuori corso, **oppure**
- 4_ essere in corso **ed** avere il piano di studio approvato, **oppure**
- 5_ essere uno studente di ingegneria **ed** avere il piano di studi non ancora approvato.

- Ricavare la funzione logica che minimizza le condizioni precedenti.

Esercizio (definizione variabili)

- Introduciamo le variabili logiche A, B, C, D, Z e definiamole nel seguente modo:
- A= lo studente ha superato almeno 20 esami
- B= lo studente è studente di ingegneria
- C= lo studente è in corso
- D= lo studente ha il piano di studi approvato
- Z= lo studente può frequentare il corso

Esercizio (tabella della verità)

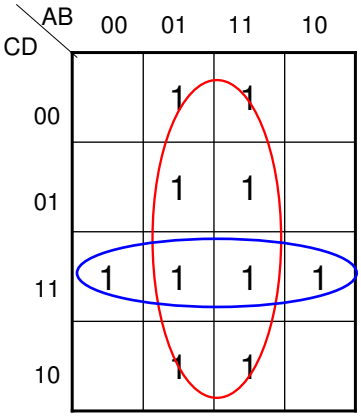
A	B	C	D	Z
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**1**
0	1	0	0	**1**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**1**
1	1	0	0	**1**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

1_ aver superato almeno 20 esami ed essere uno studente di ingegneria in corso, oppure
2_ aver superato almeno 20 esami ed essere uno studente di ingegneria con il piano di studio approvato, oppure
3_ aver superato meno di 20 esami ed essere uno studente di ingegneria fuori corso, oppure
4_ essere in corso ed avere il piano di studio approvato, oppure
5_ essere uno studente di ingegneria con il piano di studi non ancora approvato.

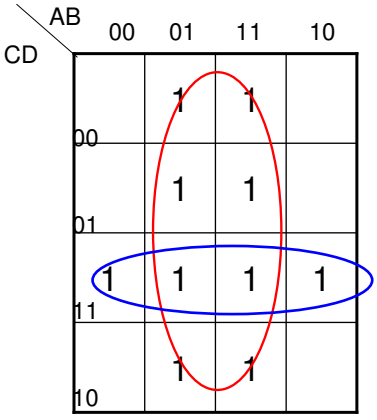
A= lo studente ha superato almeno 20 esami
B= lo studente è studente di ingegneria
C= lo studente è in corso
D= lo studente ha il piano di studi approvato
Z= lo studente può frequentare il corso

Esercizio (mappa di Karnaugh))

A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



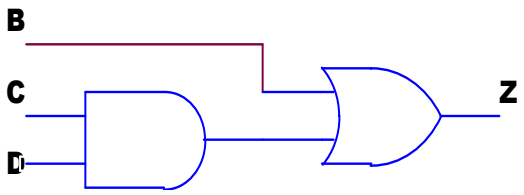
Esercizio (mappa di Karnaugh))



$Z = B + CD$

Esercizio (realizzazione)

$Z = B + CD$



Funzione AND

x	*y*	*z*
0	0	0
0	1	0
1	0	0
1	1	1

Funzione OR

x	*y*	*z*
0	0	0
0	1	1
1	0	1
1	1	1

Funzione NOT

x	*z*
0	1
1	0

Funzione XOR

x	*y*	*z*
0	0	0
0	1	1
1	0	1
1	1	0

Funzione NAND

x	*y*	*z*
0	0	1
0	1	1
1	0	1
1	1	0

Funzione NOR

x	*y*	*z*
0	0	1
0	1	0
1	0	0
1	1	0

Funzione XNOR

x	*y*	*z*
0	0	1
0	1	0
1	0	0
1	1	1