
LABORATORIO DI ARCHITETTURA DEI CALCOLATORI

Prof. Rosario Cerbone

rosario.cerbone@libero.it

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

a.a. 2007-2008

Obiettivo

- I due moduli integrati (Architettura dei Calcolatori e Laboratorio di Architettura dei Calcolatori, 6+6 CFU, **esame unico**) hanno l'obiettivo di illustrare gli aspetti fondamentali dell'organizzazione e della architettura dei moderni calcolatori elettronici. Il corso di Laboratorio di Architettura, in particolare, tratta della progettazione digitale di reti combinatorie e sequenziali e di sviluppo di programmi assembly.
-

Orario lezioni

Giovedì	11,00-14,00	Laboratorio	
Venerdì	14,00-16,00	Aula	

Ricevimento

Giovedì	14,00-14,30	
---------	-------------	--

Programma

- **Progettazione digitale**
 - **Linguaggio assembly**
 - **Sviluppo di programmi assembly**
 - **Approfondimenti**
 - **Processori Pentium, PowerPC e Athlon**
-

Materiale didattico

- Le presentazioni multimediali (formato .pdf) di tutte le lezioni saranno disponibili sul sito internet del corso.
 - Emulatori software.
-

Testi consigliati

- W. Stallings - Architettura e organizzazione dei calcolatori (progetto e prestazioni). Pearson Italia, 2004 (traduzione italiana della sesta edizione).

- Testi di approfondimento:
 - G. Bucci - Architettura dei calcolatori elettronici: fondamenti. Mc Graw-Hill Italia, 2005.

 - Franco Fummi, Mariagiovanna Sami, Cristina Silvano Progettazione Digitale (Edizione 1) McGraw-Hill

Modalità d'esame

- L'esame consisterà nella prova di Architettura dei Calcolatori e nella prova di Laboratorio.
- Quest'ultima prevederà la realizzazione di circuiti combinatori e sequenziali e di programmi assembler che implementano un algoritmo dato.

REGISTRAZIONE

- Indicare chiaramente i vostri dati.
- Importantissimo l'indirizzo e-mail.
- Le registrazioni si accettano fino a giovedì 8 novembre.

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	$\overline{x * y} = \bar{x} + \bar{y}$	$\overline{x + y} = \bar{x} * \bar{y}$

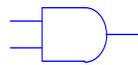
Teoremi e identità degli operatori elementari

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

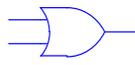
Porte logiche

- 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*

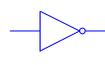
Porte logiche



AND



OR



NOT



NAND



NOR



XOR



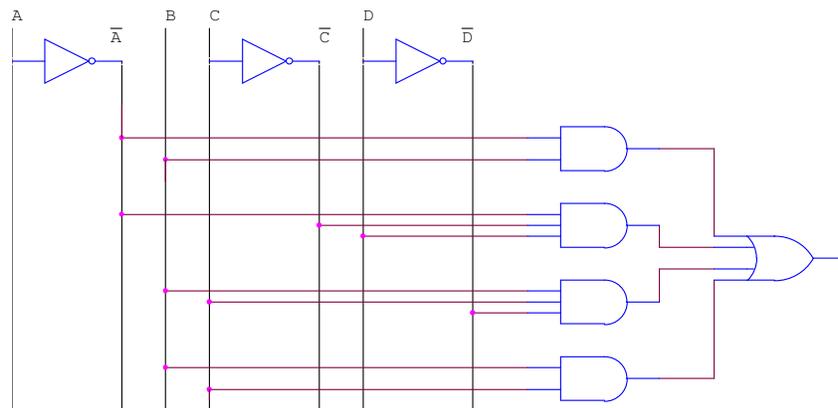
XNOR

Porte logiche

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

Porte logiche

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



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 $y = f(x_1, \dots, x_n)$:
- - 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 $P = x_1^{*1} \dots x_j^{*j}$, 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:

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

- 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:

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

- 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

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

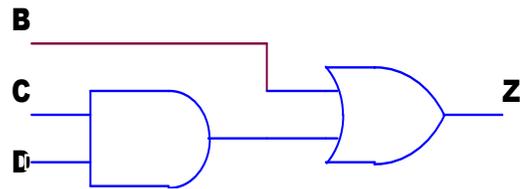
Esercizio (mappa di Karnaugh))

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

$$Z = B + CD$$

Esercizio (realizzazione)

$$Z = B + CD$$



Descrizioni di circuiti combinatori tramite SIS

- Concetti fondamentali per utilizzare SIS, un programma che consente di analizzare, ottimizzare e sintetizzare reti logiche.
- SIS permette di memorizzare, analizzare, ottimizzare e sintetizzare reti logiche in vari formati.
- Il formato *blif* è utilizzato per rappresentare circuiti combinatori rappresentati da tabelle delle verità.

Descrizioni di circuiti combinatori tramite SIS

- Per descrivere una funzione booleana nel formato blif è necessario specificare le seguenti informazioni:
- - Nome della funzione tramite la keyword **.model**
- - Elenco degli input tramite la keyword **.inputs**
- - Elenco degli output tramite la keyword **.outputs**
- - On-set per ogni output tramite la keyword **.names**
- - Il file deve essere terminato dalla keyword **.end**

Descrizioni di circuiti combinatori tramite SIS

- Ad esempio il formato blif per rappresentare una porta AND a 3 ingressi è il seguente:
- `.model and`
- `.inputs x y z`
- `.outputs a`
- `.names x y z a`
- `111 1` *spazio obbligatorio prima dell'ultimo 1
- `.end`

Descrizioni di circuiti combinatori tramite SIS

- Il formato blif può essere letto tramite il comando **read_blif** dopo aver lanciato il programma tramite il comando **sis**.
- Il circuito può quindi essere simulato tramite il comando **simulate** seguito dai valori che devono essere assegnati agli ingressi.

Descrizioni di circuiti combinatori tramite SIS

- I passi da compiere sono i seguenti:
 1. Scrivere la descrizione in formato blif con un editor di testo (blocco note) e salvare il file come nome.blif
 2. Eseguire il programma sis
 3. Leggere il file con l'istruzione `read_blif nome.blif` (abbreviato `rl nome.blif`)
 4. Controllare se il file è stato letto correttamente con `write_blif` (abbreviato `wl`)
 5. Il comando `print_stats (ps)` stampa le informazioni sul circuito
 6. Provare il circuito con `simulate (sim)` seguito dai valori da assegnare alle variabili di ingresso separati da uno spazio. Es. `simulate 1 0 1`
 7. Con `write_eqn (we)` il programma ricava le equazioni del circuito che possono essere salvate su file con `write_eqn nome.blif`
 8. Per chiudere il programma: `quit`

Descrizioni di circuiti combinatori tramite SIS

- Altri comandi utili nell'utilizzo di sis sono i seguenti:
- - **help** fornisce l'elenco dei comandi disponibili
- - **help comando** fornisce aiuto per il comando indicato
- - **alias** fornisce l'elenco delle abbreviazioni attive
- - **print_stats** fornisce informazioni sul circuito, quali numero di input, output, elementi di memoria, letterali
- - **write_blif** stampa il file blif del circuito in memoria

Esercizi

- **Esercizio 1:** Descrivere in formato blif le seguenti funzioni booleane: *AND*, *OR*, *NOT*, *NAND*, *NOR*, *XOR*, *XNOR*.
- Simularne il loro comportamento con SIS.
- Ricavare l'espressione booleana con SIS.



Esercizi

- **Esercizio 2:** Descrivere in formato blif un sommatore binario ad un bit.
- Simularne il comportamento con SIS.
- Ricavare la funzione logica



Esercizi

- **Esercizio 3:** Scrivere la tabella delle verità di un circuito che conta quanti 1 sono presenti in una sequenza di 4 cifre binarie.
- Il circuito avrà quindi 4 ingressi e 3 uscite (al massimo è possibile avere quattro 1 in ingresso e in tal caso l'output del circuito deve essere il numero binario 100).
- Descrivere il circuito in formato blif e simularne il comportamento con SIS.



Soluzioni esercizio 1

Porta AND

- .model and
- .inputs x y
- .outputs z
- .names x y z
- 11 1
- .end

Soluzioni esercizio 1

Porta OR

- .model OR
- .inputs x y
- .outputs z
- .names x y z
- 01 1
- 10 1
- 11 1
- .end

Soluzioni esercizio 1

Porta NOT

- .model NOT
- .inputs x
- .outputs z
- .names x z
- 0 1
- .end

Soluzioni esercizio 1

Porta NAND

- .model NAND
- .inputs x y
- .outputs z
- .names x y z
- 00 1
- 01 1
- 10 1
- .end

Soluzioni esercizio 1

Porta NOR

- .model NOR
- .inputs x y
- .outputs z
- .names x y z
- 00 1
- .end

Soluzioni esercizio 1

Porta XOR

- .model XOR
- .inputs x y
- .outputs z
- .names x y z
- 01 1
- 10 1
- .end

Soluzioni esercizio 1

Porta XNOR

- .model XNOR
- .inputs x y
- .outputs z
- .names x y z
- 00 1
- 11 1
- .end



Soluzioni esercizio 2

Sommatore 1 bit

- .model full_adder
- .inputs x y cin
- .outputs s cout
- .names x y cin s
- 001 1
- 010 1
- 100 1
- 111 1
- .names x y cin cout
- 011 1
- 101 1
- 110 1
- 111 1
- .end



Soluzioni esercizio 3

Contatore di 1

x3	x2	x1	x0	z2	z1	z0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	1	1
1	1	0	0	0	1	0
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	1	0	0

Soluzioni esercizio 3

Contatore di 1

- .model counter_one
- .inputs x0 x1 x2 x3
- .outputs z0 z1 z2
- .names x0 x1 x2 x3 z0
- 0001 1
- 0010 1
- 0100 1
- 0111 1
- 1000 1
- 1011 1
- 1101 1
- 1110 1
- .names x0 x1 x2 x3 z1
- 0011 1
- 0101 1
- 0110 1
- 0111 1
- 1001 1
- 1010 1
- 1011 1
- 1100 1
- 1101 1
- 1110 1
- .names x0 x1 x2 x3 z2
- 1111 1
- .end



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

