Dipartimento di Fisica Sperimentale
Laboratorio di Elettronica
parte III

D.Gamba,P.P.Trapani

April 13, 2004

1 Logica cablata e logica programmabile in tecnologia CMOS

Con il termine logica di un sistema si intende la funzione che esso svolge. I multiplexer, i demultiplexer, i decoder, gli encoder, i contatori etc.. hanno una logica cablata, fissata una volta per tutte nel circuito che collega i componenti, gate in questo caso. Si parlera' di logica pogrammabile invece quano il comportamento del circuito puo' essere modificato senza cambiarne i collegamenti, ma solo il programma che lo governa.
RIcordare che gli IC CMOS sono molto sensibili all'elettricita' statica. Fare attenzione quando si maneggiano i chip a non mettere in corto i pin, scaricare a terra la propria elettricita' statica prima di maneggiarli toccando un ground della basetta.**Contollare sempre i collegamenti** in modo che le alimentazioni siano connesse sui pin corretti e siano nei range permessi, prima di dare tensione ai circuiti.

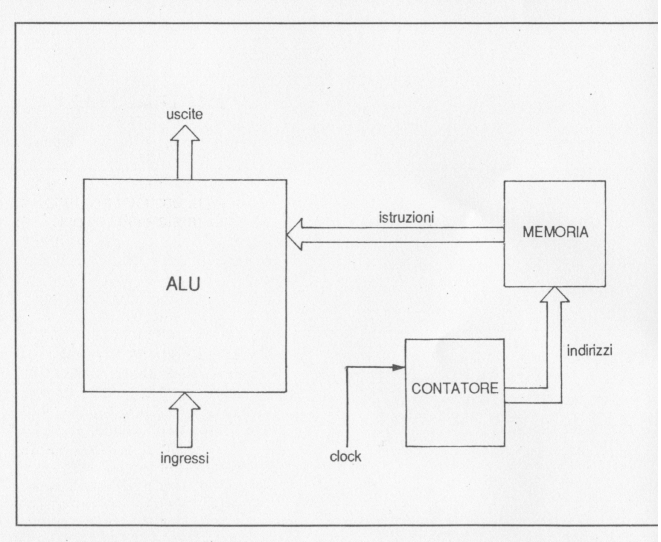


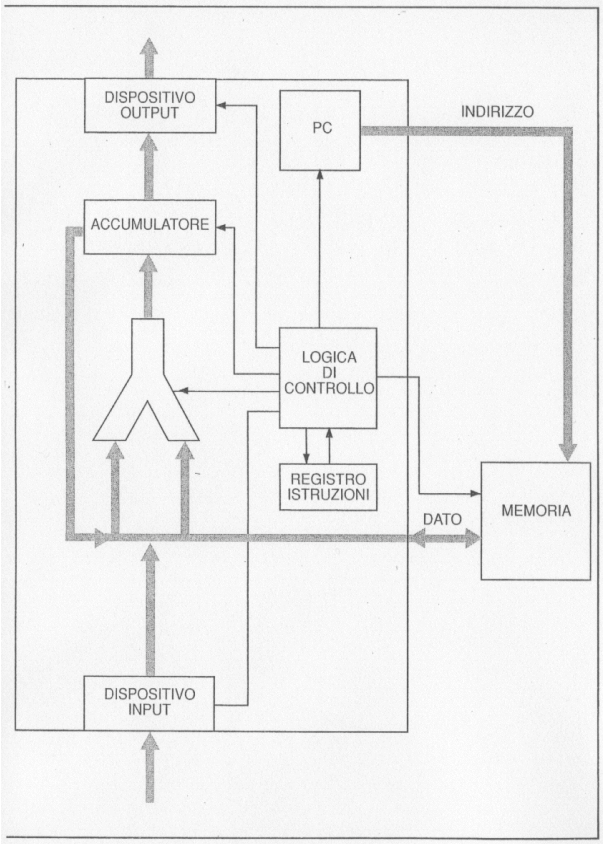
Figure 1: Schema di logica programmata

Nella fig. 1.1 e' mostrato un sistema logico in cui una ALU opera su dati provenienti dall'esterno. La memoria contiene le *istruzioni*, (indicazioni sulle istruzioni da svolgere sui dati) per la ALU.Un contatore, (*program counter*), fornisce alla memoria gli indirizzi in cui sono memorizzate le istruzioni. Canbiando programma registrato in memoria, senza cambiare il circuito, si puo' modificare la logica del sistema.

1.1 Controllore ad 1 bit, MC14500B

Per questo Lab. si e' scelto un componente un po' particolare: il 14500 che e' un dispositivo programmabile ad un solo bit, con un set di istruzioni limitato. Lo scopo e' di costruire un sistema programmabile completo, in grado di implementare funzioni logiche anche complesse. Lo schema

del componente, fig. **??**, mostra i circuiti base composti da una Logic Unit, che ha le stesse funzioni pur se limitate di una ALU, da un registro Accumulatore ad un bit e da un FF per gestire l'I/O.



2 *Struttura di un dispositivo programmabile.*

HARDWARE E SOFTWARE

ıine *hardware* - da *hard* = duro e *ware* = merce, articoli - significa ferramenta, ar-
li ferro. Usato inizialmente in forma scherzosa questo vocabolo è entrato nell'u-
rente per definire le componenti fisiche di un computer.
ınto essenziale dell'hardware di un computer sono i circuiti elettronici digitali: dal-
J, unità centrale di processo, alle memorie, dal generatore di clock ai dispositi-

Da *soft* (= morbido) è stato coniato, per analogia, il termine *software* per de
che serve al funzionamento di un elaboratore ma non ha tangibilità fisica: i p
Come abbiamo visto nel capitolo 4, si usa il termine *firmware* per i progran
ti) residenti in chip di memoria non volatile.

5.3 CONTROLLORE INDUSTRIALE 14500

Abbiamo scelto, per il passaggio dalla logica cablata a quella programmab
ponente un po' particolare: il 14500 è un dispositivo programmabile ad un s
un set di istruzioni molto limitato. Ma proprio per queste sue caratteristiche
sta ad affrontare, nello spazio di un solo capitolo, sia le problematiche har
quelle software di un sistema programmabile. In Jack Book 4 le stesse ten
no affrontate con riferimento al più diffuso microprocessore a 8 bit, lo Z80; n
punto ad esso è dedicato un intero volume.
Inoltre - senza superare la complessità circuitale che avete imparato ad aff
precedenti Jack Book - potrete realizzare un sistema programmabile comp
zionante.

In *Data 23* abbiamo riportato - dal Data Book della Motorola - le specifiche
componente. Notate intanto, sulla prima pagina, la scritta *"Detailed operati
plications are given in the «MC14500B Industrial control unit» handbook"*:

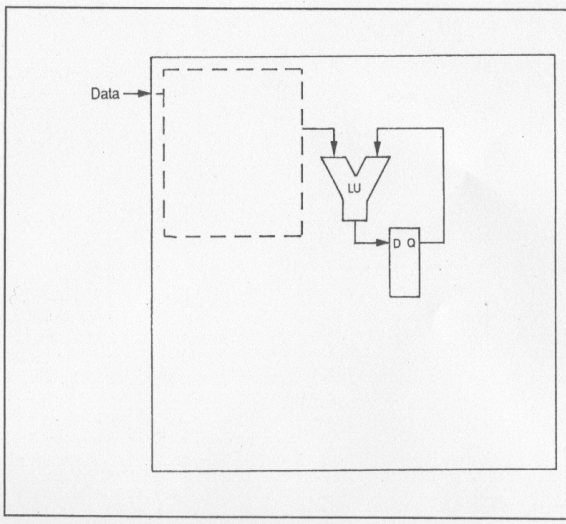
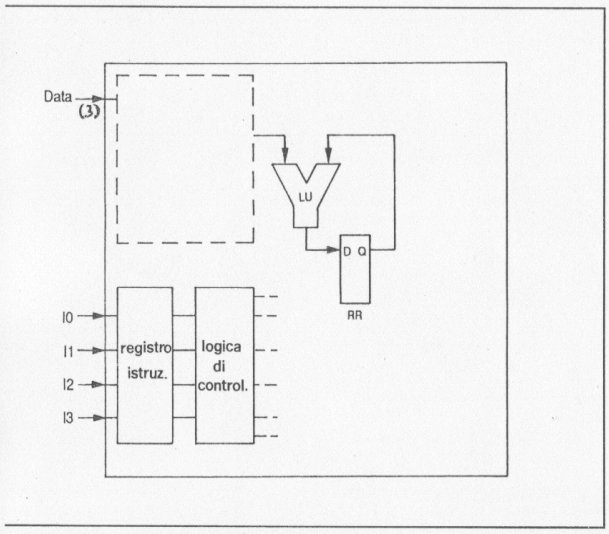


Fig. 5.3 *Nello schema a blocchi del 14500 si riconoscono un'unità logica e*
accumulatore. L'unità logica agisce su un dato di un solo bit.



.4 *Quattro sono i pin per l'ingresso delle 16 istruzioni.*

nisce un intero libro relativo a questo componente.
ck diagram è una rappresentazione, a grandi blocchi, dello schema interno. Un
o, dovrebbe aver attirato la vostra attenzione: quello a Y, che conoscete come
Qui la A manca, potete immaginare che LU stia per *Logic Unit*, unità logica. An-
olo uno sguardo superficiale alla tabella delle istruzioni dovrebbe, del resto, aver-
strato la presenza di istruzioni quali AND e OR e l'assenza di operazioni aritme-
.

n unico pin denominato *Data*. Nella presentazione del componente, inoltre, è det-
iaramente che si tratta di un *single bit processor*, di un processore a un solo bit.
ita dell'unità logica entra in un blocco denominato RR, *result register*, registro del
ato, che, a sua volta, funziona come ingresso per l'altro ramo dell'unità logica. RR
nque l'accumulatore, realizzato con un semplice flip flop di tipo D.
g. 5.3 abbiamo evidenziato il pezzo dello schema a blocchi analizzato: fra il pin 3,
sso di Data, e l'ingresso della LU abbiamo per ora lasciato un blocco tratteggiato.

iamo ad analizzare le 16 istruzioni. Per codificarle in binario occorrono quattro bit:
dovrebbe esservi difficile identificarli in

I0, pin 7, *LSB instruction word* ovvero bit meno significativo della parola istruzione
I1, pin 6
I2, pin 5
I3, pin 4, *MSB instruction word*, bit più significativo dell'istruzione.

Come vedete questi pin sono gli ingressi di un blocco denominato INST R
ction register o registro delle istruzioni. Il blocco vicino al registro delle istru
te interpretarlo come logica di controllo del sistema: si tratta di un circuito
difica le istruzioni e invia i segnali necessari per realizzarle a tutti gli altri e
sistema (Fig. 5.4).

Analizziamo il set delle istruzioni (quarta pagina del data). Una prima colon
bella riporta l'*instruction code*, il codice dell'istruzione, una seconda, *mner*
me dell'istruzione in una forma per noi più semplice da ricordare; la terza in
ne svolta da quell'istruzione.
L'istruzione di codice

0000

è denominata NOP0, ovvero *No OPeration 0*, con l'ultima 0 che si riferisce a
10). La scritta

RR → RR

si presta facilmente ad essere interpretata come: il contenuto di RR (*res*
va in RR (che dunque non si modifica); mentre quella

Flag 0 → ∏

indica evidentemente un impulso sull'uscita flag 0.
Se guardate l'ultima istruzione, codice

1111

nome mnemonico NOPF - *No OPeration F* - vedete che è simile: nessun ca
to nei registri e un impulso su flag F (pin 9).

L'istruzione con codice

0001

load result register, carica il registro RR. Come vedete il valore presente
ta viene posto in RR:

Data → RR

Il nome attribuito a quest'istruzione LD sta, evidentemente per *LoaD*.
L'istruzione successiva, LDC - *LoaD Complement* - carica in RR il compl
dato:

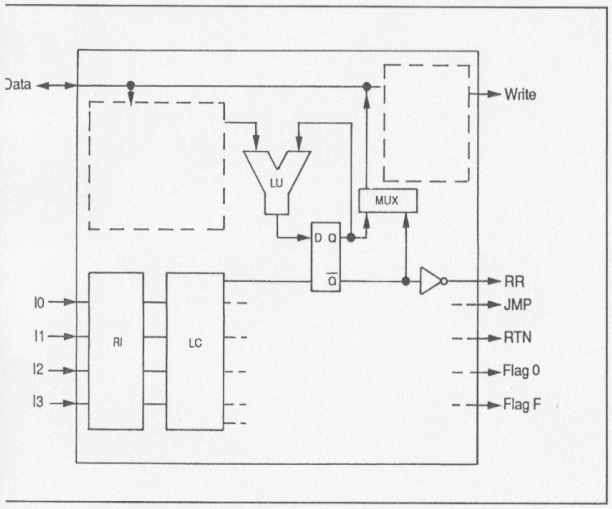
$\overline{\text{Data}}$ → RR

L'istruzione AND pone in RR il risultato del prodotto logico fra RR e Data

RR · Data → RR

ANDC pone in RR il risultato del prodotto fra RR e il complemento di Dat

RR · $\overline{\text{Data}}$ → RR



.5 Attraverso un multiplexer, il dato contenuto in RR o il suo negato possono arrivare al pin gresso e uscita). Le istruzioni STO e STOC inoltre mandano alto write.

one in RR il risultato della somma logica fra RR e Data:

$$RR + Data \rightarrow RR$$

pone in RR il risultato della somma logica fra RR e il complemento di Data:

$$RR + \overline{Data} \rightarrow RR$$

R esegue un exclusive NOR: in RR va un 1 se RR e Data sono uguali, uno 0 se diversi.

- per *STOre* - manda (immagazzina) il contenuto di RR sul pin Data, mentre STOC da sul pin Data il complemento del contenuto di RR. Sia STO che STOC inviano e un impulso su write (pin 2).

caso, prima di proseguire, di tornare allo schema a blocchi. Il registro RR ha due e: Q e Q negato. Q riporta il dato all'ingresso dell'unità logica. Da Q negato, attrso una NOT il valore di RR arriva al piedino 15 (Fig. 5.5).
llelamente Q e Q negato risultano gli ingressi di un multiplexer attraverso il quano o l'altro - a seconda dell'istruzione STO o STOC - arrivano al pin 3 (Data).
omplessa la strada attraverso cui viene azionato il segnale di write (pin 2) con il viene comunicata all'esterno la trasmissione del dato sul pin 3.

lteriore istruzione è IEN, *Input ENable*: abilita l'ingresso. Il valore presente sul pin

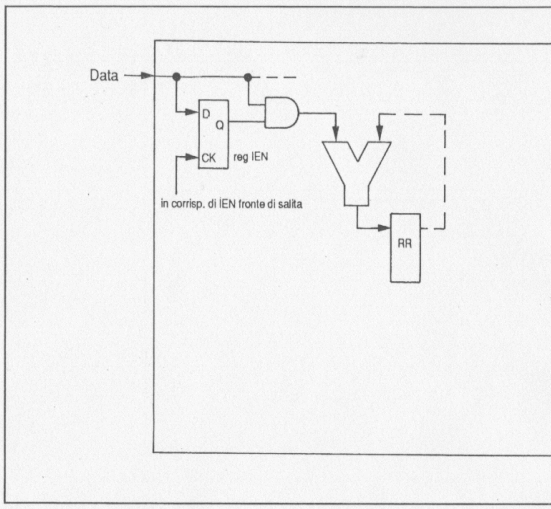


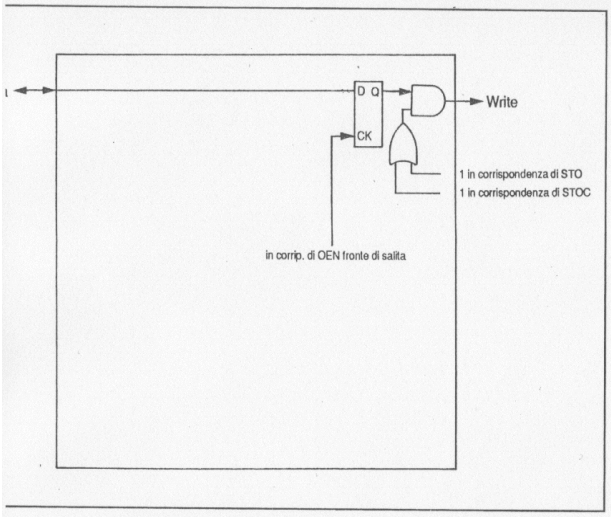
Fig. 5.6 *Una porta AND abilita o blocca il passaggio del dato, presente sul pin 3, ve dell'unità logica. La AND è comandata da un flip flop.*

Data viene trasferito nel registro IEN (il registro e l'istruzione hanno lo s ma sono due cose distinte). Analizziamo il percorso del dato fra il pin 3 e l'i l'unità logica (Fig. 5.6). Il pin 3 è collegato ad un ingresso di una AND che me gate: il segnale passa solo se il secondo ingresso è a "1". Ma il sec so della AND proviene da un flip flop di tipo D: il flip flop IEN. L'ingresso to a Data; l'ingresso C - clock - è comandato, par di capire, dall'istruzio In corrispondenza di tale istruzione il flip flop acquisisce il dato, presente D, lo memorizza su Q e lo conserva fino ad una nuova istruzione IEN. S sente al momento dell'istruzione IEN è "1" allora

$$Q = 1$$

e l'ingresso è abilitato: Data può passare attraverso la porta AND e pres gresso della LU.
Se invece l'uscita Q del flip flop IEN è "0" il segnale non passa dal pin 3 gresso è disabilitato).

L'istruzione OEN , *Output ENable*, abilita l'uscita: il dato presente sul pi sferito nel registro OEN, anche questo un flip flop di tipo D cui il coman fornito in corrispondenza all'istruzione OEN (non confondere istruzione e che se hanno lo stesso nome). Attraverso la OR alla AND che comanda te giunge un "1" in corrispondenza dell'istruzione STO o STOC. Se l'usci si trova ad "1" (l'uscita è stata abilitata), la AND fornisce al pin di write un s Se il Q di OEN è a "0" write resta basso (Fig. 5.7). L'abilitazione dell'uscita



.7 Le istruzioni STO e STOC mandano alto write solo se in precedenza l'uscita è stata ıta per mezzo di un'istruzione OEN con un "1" in ingresso.

rda il fatto che il contenuto di RR arrivi al pin 3 ma il fatto che write, possa anda-
o.
zione: per abilitare ingresso o uscita non basta dare l'istruzione IEN (o OEN), oc-
che in corrispondenza di tale istruzione sul pin 3 sia presente un "1". In altre pa-
EN e OEN servono sia per l'abilitazione, che per la disabilitazione di ingresso e
a, secondo questa tabella:

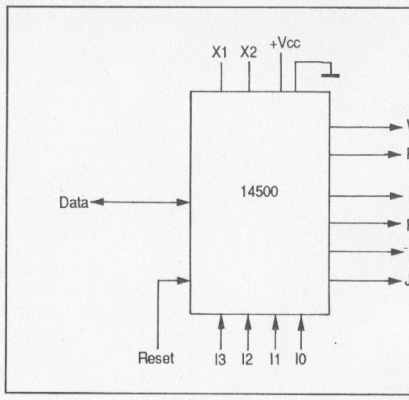
istruzione	pin 3	effetto
IEN 0	ingresso	disabilitato
IEN 1	ingresso	abilitato
OEN 0	uscita	disabilitata
OEN 1	uscita	abilitata

ngono da analizzare tre istruzioni:

- *JuMP* ovvero salta - fornisce un impulso sul pin 12, JMP

- *ReTurN* ovvero ritorna - da un analogo impulso sul pin 11, e provoca la non ese-
ne dell'istruzione successiva

Fig. 5.8 *Il 14500 disegnato come blocco: si tratta di un integrato a 16 pin.*



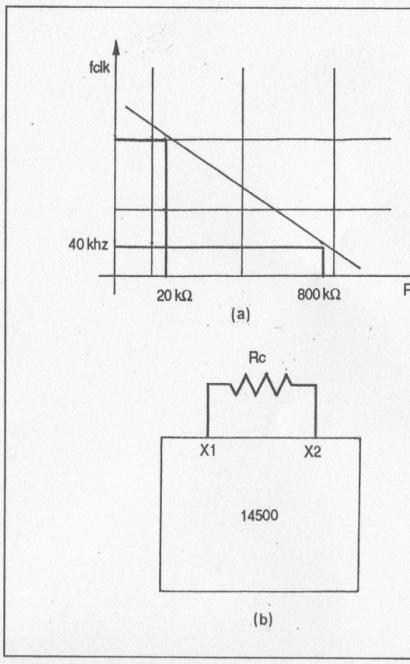
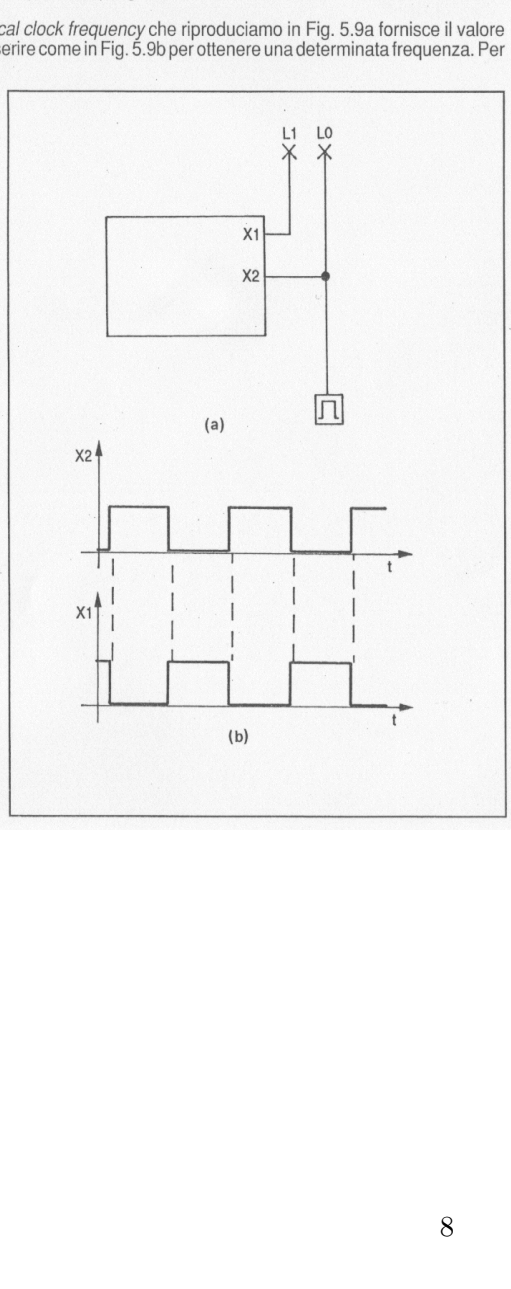


Fig. 5.9 *Il 14500 ha un oscillatore interno. Per mezzo di una resistenza esterna, collegata fra X1 e X2, si può scegliere la frequenza di clock.*

Z - *SKip Zero*, salta se zero, provoca il salto dell'istruzione successiva se

RR = 0

biamo così considerato tutte le istruzioni e quasi tutti i piedini. Di quelli rimanenti g. 5.8 - due sono quelli dell'alimentazione, V_{DD} da collegare al positivo dell'alimen- one e V_{SS} da collegare al negativo, o a massa. Il piedino 1 è il *chip reset*, quello che etta, azzera, il sistema: pone a zero RR e i registri IEN e OEN.
e X2, collegati al blocco OSC, oscillatore, sono rispettivamente *oscillator output* e *illator input*. Un sistema come questo ha naturalmente bisogno di un clock per sin- nizzare le diverse operazioni dei suoi blocchi interni, ma anche per sincronizzar- on quelle del mondo esterno. Avrete notato fra le caratteristiche del componente critta *on chip clock (oscillator)*: significa che il 14500 è fornito internamente di oscil- re per il clock.
agramma *Typical clock frequency* che riproduciamo in Fig. 5.9a fornisce il valore esistenza da inserire come in Fig. 5.9b per ottenere una determinata frequenza. Per



5.10 *Se invia- un clock a X2, X1 troviamo il nale comple- ntare.*

avere 1 Mhz, per esempio, useremo un resistore da 20 KΩ; per avere mo 800 KΩ.
E' possibile utilizzare anche un segnale di clock esterno, da collegar sto caso su X1 si ritrova il clock invertito: potete verificarlo collegando p come in Fig. 5.10.

5.4 SISTEMA DI NUMERAZIONE ESADECIMAL

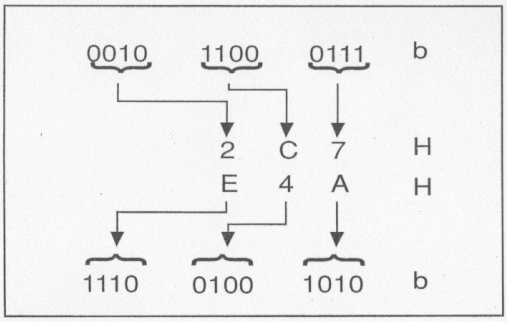
Avrete notato che, nel set di istruzioni - il codice binario dell'istruzione è un numero decimale corrispondente al numero binario codice dell'istruz cimale 9. Per l'istruzione IEN, binario 1010, invece del decimale 10 vi è così per le seguenti fino a F. Si tratta di un particolare codice, detto e

Per i calcolatori il sistema di numerazione binario è la soluzione miglio l'uso di tale sistema è piuttosto disagevole: è facile commettere errori quenze di "0" e di "1". E' già disagevole esprimere in binario un byte, fi dirizzo di memoria di 16 bit (microprocessore Z80) o di 20 bit (microp 8086 e 8088) o addirittura di 32 bit (Motorola 68000, Intel 80386).
Peraltro la traduzione da decimale a binario e viceversa non è agevole decimale ha i grossi vantaggi di un minor numero di cifre rispetto al bina mediata traducibilità.
Ad ognuna delle sedici possibili configurazioni di un nibble (4 bit) corris bolo del sistema esadecimale, Fig. 5.11.

binario	esadecimale	deci
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Fig. 5.11 *Il codice esadecimale permette di rappresentare quattro bit di un numero binario con una sola cifra.*

5.12 *Conver-*
ıe da binario in
decimale e da
decimale in bi-
o.



uattro cifre binarie corrisponde dunque sempre una cifra esadecimale e viceversa. La conversione da binario a esadecimale e da esadecimale a binario è molto semplice, come mostrano gli esempi di Fig. 5.12.
:he quello esadecimale è un sistema di numerazione di tipo posizionale, basato e potenze di 16, come mostrato in Fig. 5.13.
evitare confusioni i numeri binari devono essere contrassegnati con la lettera "b", lli esadecimali con una "H".
ì

10

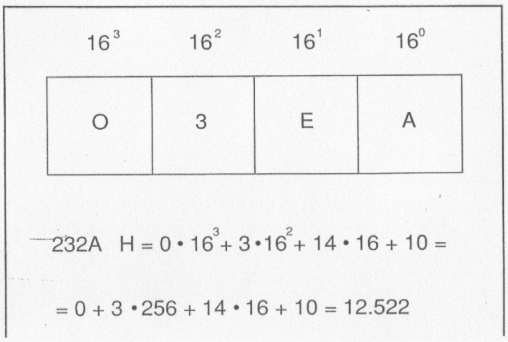
ıifica "dieci" (decimale),

10 b

ıifica "uno-zero" binario (ovvero 2 decimale),

10 H

ıifica "uno-zero" esadecimale (ovvero 00010000 binario e 16 decimale).



. 5.13 *Quello*
decimale è un
ema di numera-
ıe posizionale
ato sulle poten-
di 16

Esercitazione 5.A

Progettate un circuito per verificare sperimentalmente le informa
14500B tratte dalla lettura del data sheet.
Provate l'istruzione *Load*.

Per provare il componente la cosa più semplice è por
board collegando gli ingressi a switch e pulsanti e le u
Attraverso quattro switch selezioneremo l'istruzione
visualizzeremo lo stato del registro RR, l'uscita wri
flag: flag 0, flagF, RTN, JMP.
Ma come comportarci con la linea Data che è bidirez
gresso la comanderemo tramite uno switch, in uscit
zeremo su di un LED, ma ingresso e uscita dovranr
parati. Si può farlo per mezzo di un componente thre
in Fig. 5.14, per esempio il 74HC244 già usato in esere
oppure un 74126 (Jack Book 1, esercitazione 4.C): p
istruzioni in cui viene acquisito un dato dal pin 3 abilit
state

S5 = 1

Lo stato di S4 verrà visualizzato sul LED 8, ma ciò no
In corrispondenza delle istruzioni in cui il dato esce
remo

S5 = 0

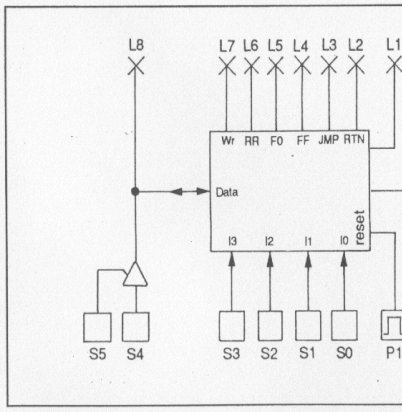


Fig. 5.14 *Per provare il componente possiamo usare sw*
linea di ingresso deve essere separate da quella di uscit

in modo che la linea di ingresso sia in alta impedenza: sul LED 8 arriva l'uscita.

Per poter provare istruzione per istruzione e vederne gli effetti ci è utile fornire manualmente gli impulsi di clock. Abbiamo pertanto collegato X2, *oscillator input*, a un pulsante e a un LED; X1 ad un altro LED. (Attenzione: nei diagrammi temporali il clock di riferimento è X1, non X2. Quando premete P1 provocate un fronte di salita di X2, dunque un fronte di discesa di X1. Quando lo rilasciate: fronte di discesa per X2 e di salita per X1).
Un ulteriore pulsante serve per il reset.

Se avete montato il circuito cominciate col dare un reset: quale possa essere lo stato iniziale dei registri è casuale, dopo il reset essi si trovano a 0. Ponete ora

S5 = 1

per abilitare il three state e

S4 = 1

per fornire un "1" in ingresso. Tramite gli switch S0-S3 predisponete l'istruzione LD. Osservando i LED premete P1 (fronte di discesa di X1) poi lasciatelo (fronte di salita di X1).
Cos'è successo? Il LED 8 è sempre stato acceso perchè direttamente collegato all'ingresso alto. I LED 0 e 1 si sono accesi e spenti in modo alternato in corrispondenza degli ingressi di clock. Gli altri LED sono rimasti tutti spenti. E' giusto? Forse vi aspettavate che si accendesse il LED 6, corrispondente ad RR: l'istruzione LD carica in RR il dato presente all'ingresso, e il dato era "1". Ma l'ingresso non era stato abilitato.
Lasciando sempre

S5 = S4 = 1

selezionate l'istruzione IEN, fornite un fronte di discesa e uno di salita a X1. I LED non segnalano alcuna novità, ma l'ingresso dovrebbe essere stato abilitato. Ripetete le operazioni per caricare "1" in RR:

S5 = S4 = 1

istruzione LD, discesa e salita di X1. Ecco che il LED 6 si dovrebbe accendere. Ponete

S4 = 0

e ripetete l'istruzione LD. In RR viene caricato uno "0" e LED 6 si spegne.
Notate l'ordine in cui vi abbiamo indicato di eseguire le operazioni:

- impostazione dato
- impostazione istruzione
- fronte di discesa di X1
- fronte di salita di X1
- impostazione dato
- ...

Non è l'unica sequenza possibile. Potete provarne
evitare di procedere per tentativi occorre rifarsi ai di
porali.

5.5 FETCH ED EXECUTE DELLE ISTRUZIONI

Considerate per prima la *timing waveform* che si riferisce alle istruzioni L
ANDC, OR, ORC, XNOR e IEN. In Fig. 5.15 l'abbiamo riprodotta aggiu
si: in ordinate c'è sempre il tempo, in ascisse la tensione che può assu
livelli alto e basso.
Il primo diagramma è quello del clock X1 (*output*), che scandisce i tempi
chi costituenti il 14500: gli scalini indicano chiaramente quando il clock
e viceversa.
I diagrammi di RR e di Data sono caratterizzati da una doppia linea: le "
istanti in cui il valore cambia (o può cambiare), indipendentemente ch
a "1" o da "1" a "0". In alcuni tratti la presenza di una linea unica indica
lore basso o alto.
Il secondo diagramma è relativo ai quattro bit dell'istruzione. Questi b

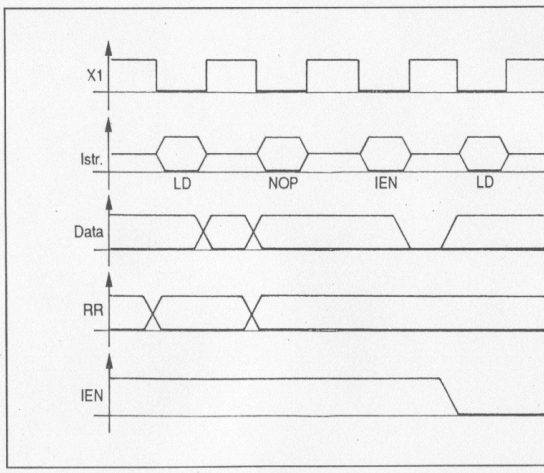
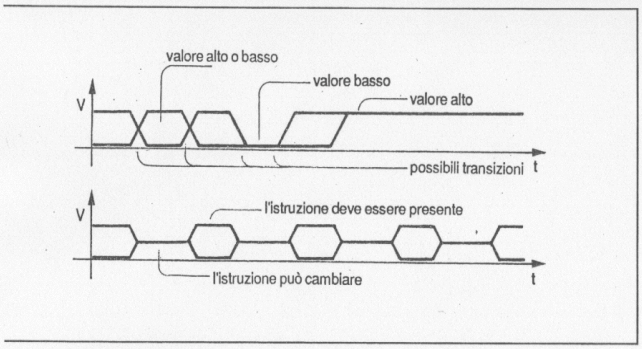


Fig. 5.15 *Riproduciamo il diagramma temporale relativo all'istruzione NOP. Co*



g. 5.16 *Alcuni dei simboli utilizzati nei diagrammi temporali del 14500.*

all'esterno, non è il 14500 a fissarne il valore: ciò che conta è che l'istruzione sia pre-
ente in un determinato intervallo di tempo, quello indicato in Fig. 5.16.

e l'istruzione deve essere presente in corrispondenza del fronte di discesa del clock
ò significa che è in tale istante che viene letta, o - come si dice - che ne viene effet-
ıato il *fetch* (cattura).
upponiamo si tratti di una LD: il dato che viene acquisito non è quello presente sul-
stesso fronte di discesa, ma quello presente sul successivo fronte di salita. Il da-
può cambiare (non è obbligatorio che cambi) prima di tale fronte. Si dice che l'*exe-
ute* (l'esecuzione) dell'istruzione avviene sul fronte di salita.
eggiamo la fig. 5.17 che riproduce il timing di Load. Sul fronte di discesa di X1 vie-
e effettuato il fetch dell'istruzione. Nel caso questa sia una Load, se IEN è alto, allo-
ı sul fronte di salita di X1 il dato presente sul pin 3 viene caricato nel registro RR.

uanto detto vale per l'istruzione LD ma anche per LDC, per le operazioni logiche e
er l'istruzione IEN: per tutte queste il fetch avviene sul fronte di discesa e l'execute
ıl fronte di salita. Di conseguenza l'istruzione deve essere presente nel momento del
onte di discesa e il dato in quello del fronte di salita.
termine "nel momento", naturalmente, è un po' vago. In Fig. 5.18 abbiamo eviden-
ato le indicazioni dei tempi (i valori sono nella tabella delle switching characteristics).
istruzione deve essere presente un tempo

$$t_{su(I)}$$

etup time dell'istruzione, prima del fronte di discesa. Nel caso peggiore e con alimen-
zione a 5 V

$$t_{su(I)} = 400 \text{ ns}$$

istruzione deve permanere per un tempo

$$t_{h(I)}$$

Fig. 5.17 *Sul fronte di discesa di X1 viene letta l'istruzione. Se si tratta di una LD il comportamento successivo dipende dallo stato del flip flop IEN. In questo caso, poichè IEN è a "1", sul successivo fronte di salita, il dato presente in ingresso viene trasferito in RR.*

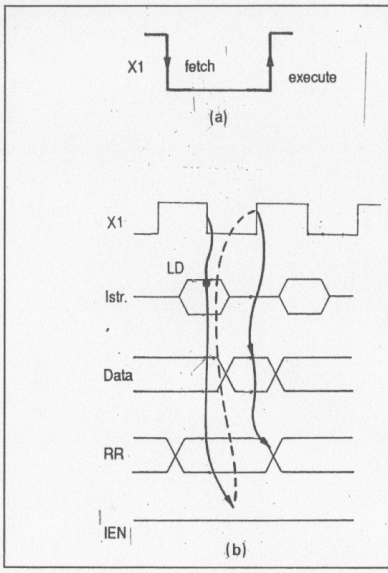
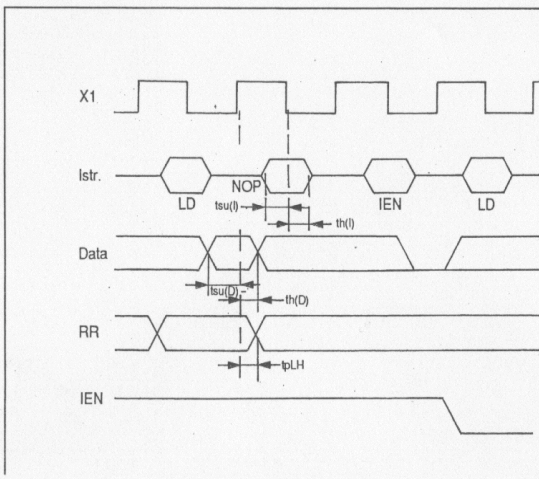


Fig. 5.18 *I tempi indicati in figura devono essere cercati nella tabella switching characteristics.*



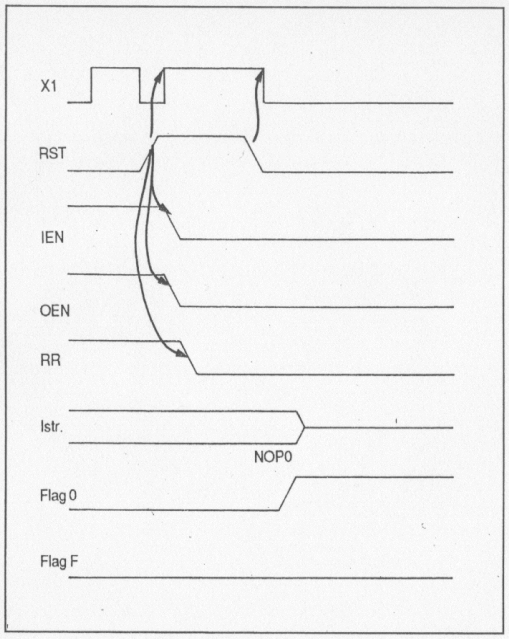


Fig. 5.21 *Un segnale di reset azzera i registri e manda alto X1.*

discesa (è sicuramente valido fino a quel punto). Dunque, in questo caso l'*execute* inizia sullo stesso fronte del *fetch*.
Nel timing che stiamo considerando dopo una prima istruzione STO o STOC ne viene incontrata ed eseguita una seconda, poi c'è una NOP (i registri non cambiano) e quindi una OEN che, essendo Data a "0", pone il flip flop OEN a zero. Infine c'è una nuova STO che non ha effetto poichè trova OEN basso.

Eccoci alle istruzioni NOP0 e NOPF, che non toccano i registri RR, IEN e OEN. In questo diagramma viene illustrato anche l'effetto di un segnale di reset (Fig. 5.21): manda bassi RR, IEN e OEN, ma ha effetto, come vedete, anche sul clock X1. Dopo l'arrivo del reset X1 va alto e resta tale fino a quando resta alto reset (più un ritardo t_{pHL}), quindi riparte con un nuovo fronte di discesa, dunque in grado di leggere un'istruzione.
L'istruzione NOP0 (Fig. 5.22) viene letta sul fronte di discesa e immediatamente viene mandato alto il flag 0, che resta tale fino al successivo fronte di discesa. Analogamente per NOPF e flag F. Se

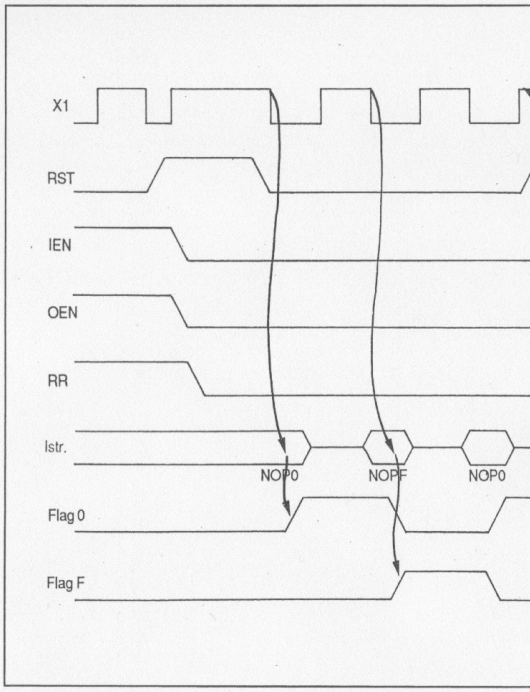


Fig. 5.22 *NOP0 e NOPF mandano alto il relativo flag su*
Il flag torna basso sul fronte di discesa successivo.

ne subito abbassato.

Infine il timing di SKZ, JMP e RTN. Anche queste is
cano i registri RR, IEN e OEN. Sul fronte di discesa
(salta se zero) verifica lo stato di RR; se questo è a
to un flip flop interno, di nome SKP (Fig. 5.23). Su
sa del clock, a quanto pare, viene controllato lo st
flop: se è basso l'istruzione viene letta, altrimenti,
caso, ignorata.
L'istruzione JMP (salta) sembra analoga a NOP
che il flag su cui agisce è JMP. L'istruzione RTN
alti, sul fronte di discesa del clock, sia il flag RTN
terno SKP, che non fa prendere in considerazione
cessiva. Se arriva un reset mentre è in corso un'
il relativo flag è subito mandato basso

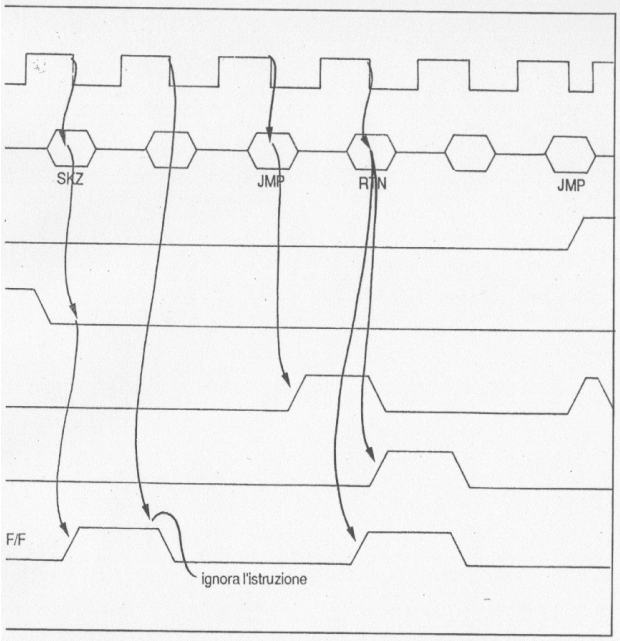


Fig. 5.23 *Le istruzioni SKZ e RTN mandano alto un flip flop SKP che fa sì che la successiva istruzione venga ignorata.*

ercitazione 5.C

ella quarta pagina del data sheet del 14500 vi è una figura intitolata *outline of typical organization for a MC14500B based system*, ovvero schema di un tico sistema basato sul 14500. Partite da essa per progettare un sistema programmabile.
vete a disposizione, in *Data 24* e *Data 25*, i fogli delle specifiche dei componenti,14512 e 14599, citati nello schema.

Analizziamo, in primo luogo, il 14512. Si tratta di un *8 channel data selector*, ovvero di un multiplexer ad 8 canali - Fig. 5.24a - con tre ingressi di indirizzo, C, B e A, un ingresso *disable* e un ingresso *inhibit*. Se

disable = 1

Data 24

MOTOROLA

MC1409
MC1459

8-BIT ADDRESSABLE LATCHES

The MC14099B and MC14599B are 8-bit addressable latches. Data is entered in serial form when the appropriate latch is addressed (via address pins A0, A1, A2) and write disable is in the low state. Chip enable must be high for writing into MC14599B. For the MC14599B the data pin is a bidirectional data port and for the MC14099B the input is a unidirectional write only port. The Write/Read line controls this port in the MC14599B.

The data is presented in parallel at the output of the eight latches independently of the state of Write Disable, Write/Read or Chip Enable.

A Master Reset capability is available on both parts.

- Serial Data Input
- Parallel Output
- Low Input Capacitance – 5.0 pF typical
- Master Reset
- Noise Immunity – 45% of V_{DD} typical
- Supply Voltage Range = 3.0 Vdc to 18 Vdc
- Capable of Driving Two Low-Power TTL Loads, One Low-Power Schottky TTL Load or Two HTL Loads over the Rated Temperature Range
- MC14099B pin for pin compatible with CD4099B
- Pin for pin compatible with Fairchild 4724.

CMOS M
(LOW-POWER COMPLEM
8-BIT ADDRESSAB
MC14599B WITH BIDIREC

L SUFFIX
CERAMIC PACKAGE
CASE 620

L SUFFIX
CERAMIC PACKAGE
CASE 726

MAXIMUM RATINGS (Voltages referenced to V_{SS})

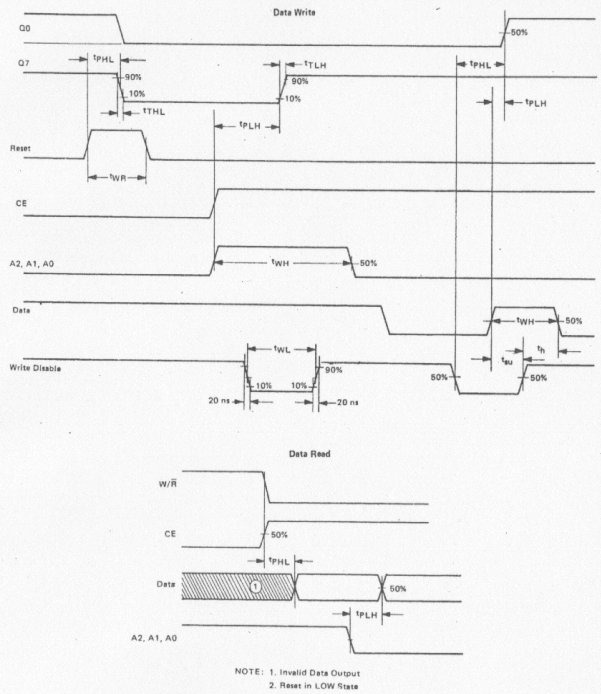
Rating	Symbol	Value	Unit
DC Supply Voltage	V_{DD}	-0.5 to +18	Vdc
Input Voltage, All Inputs	V_{in}	-0.5 to V_{DD} + 0.5	Vdc
DC Current Drain per Pin	I	10	mAdc
Operating Temperature Range AL Device	T_A	-55 to +125	°C
CL CP Device		-40 to +85	
Storage Temperature Range	T_{stg}	-65 to +150	°C

ORDERING INFO
MC14XXXB — Suffix
L Cera
P Plas
A Ext
Tem
C Lim
Tem

MC14099B
MC14599B
PIN ASSIGNMENT
PIN ASSIGNMENT

This device contains circu
the inputs against damage d
voltages or electric fields;
advised that normal precau
to avoid application of any
than maximum rated volta
impedance circuit. For prop
is recommended that V_{in}
constrained to the range
$V_{out} < V_{DD}$.
Unused inputs must alway
appropriate logic voltage le
V_{SS} or V_{DD}).

MC14599B
TIMING DIAGRAMS



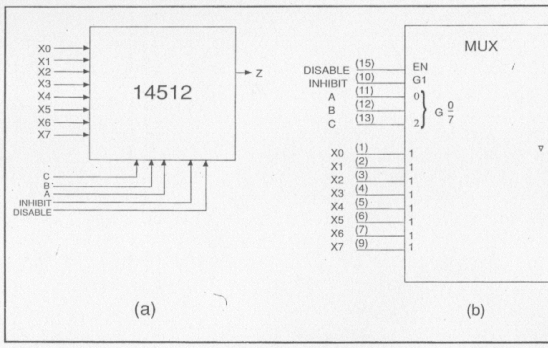


Fig. 5.24 *In (a) ingressi e uscite del multiplexer 14512 rappresentazione a norme IEC.*

comunque siano gli altri ingressi, l'uscita è in alta im

disable = 0 e inhibit = 1

comunque siano gli altri ingressi, l'uscita è bassa. S

disable = 0 e inhibit = 0

l'uscita è uguale all'ingresso selezionato. In Fig. 5.24 disegnato il componente secondo le norme IEC. Si hibit - attivo basso (nel senso che l'azione è perme basso, inibisce quando è alto) - svolga una funzione gresso di volta in volta selezionato.

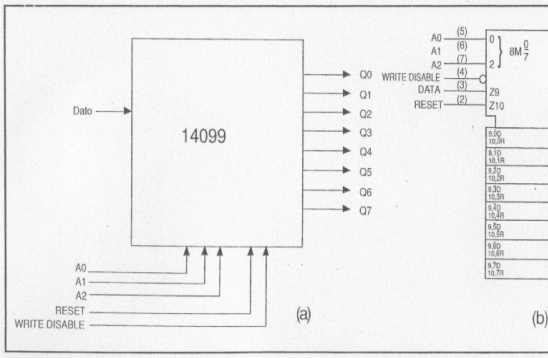


Fig. 5.25 *In (a) ingressi e uscite dell'addressable latch 140 rappresentazione a norme IEC.*

Passiamo a *Data 25*: 14099 e 14599 sono due *8 bit addressable latch*, ovvero latch indirizzabili ad 8 bit (vedi Jack Book 2, esercitazione 1.G). In Fig. 5.25a abbiamo ridisegnato il 14099. Se

write = 0 e reset = 0

l'uscita indirizzata - da A0÷A3 - riproduce l'ingresso, le altre uscite sono in situazione di memoria (mantengono il dato precedente). Se

write = 0 e reset = 1

l'uscita indirizzata riproduce l'ingresso mentre le altre vengono resettate. Se

write = 1 e reset = 0

tutte le uscite sono in condizione di memoria. Infine, se

write = 1 e reset = 1

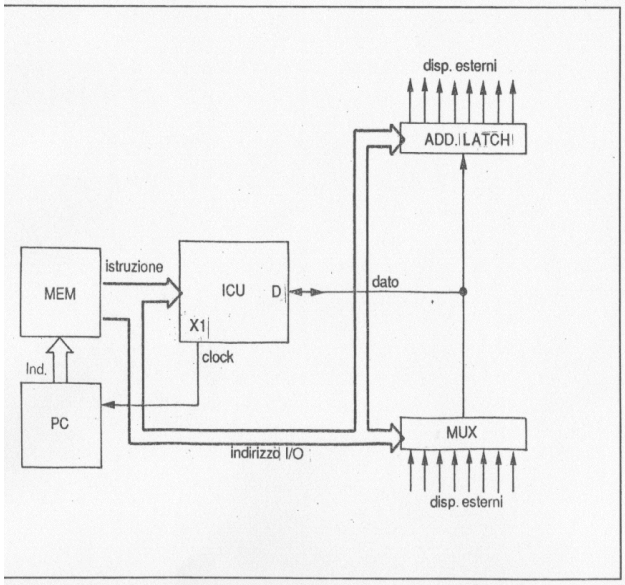


Fig. 5.26 *Un sistema programmabile basato sul 14500. La memoria contiene le istruzioni per il processore e i bit di selezione per i canali di ingresso e uscita.*

tutte le uscite vengono resettate.
Volendo disegnare il componente secondo le norme
ta Fig. 5.25b.
Noi useremo questo componente al posto del 1459
dati è bidirezionale e che ha anche un ingresso *chip*

Riproduciamo, in Fig. 5.26, lo schema proposto dal da
in forma graficamente diversa.
Per mezzo del multiplexer e dell'addressable latch l'u
ti del 14500 può essere collegata ad 8 linee di ingress
tante in uscita.
Il clock dell'ICU agisce su un contatore, chiamato con
gramma perchè la sua uscita indirizza la memoria ch
sequenza di istruzioni: si tratta di un sottoinsieme d
realizzato nel capitolo 3 (esercitazione 3.E).

Considerate ora l'uscita della memoria: essa è collegata
I3 - I0 del 14500 ma anche con multiplexer e addres
bit che arrivano, in parallelo, a 14512 e 14099 sono,
quelli di indirizzo.
Utilizzando una memoria a 8 bit questi possono aver
loro attribuita in Fig. 5.27. Ogni cella di memoria con
un'istruzione costituita da due parti:

istruzione 14500 + indirizzo dato

per esempio

LD <indirizzo 4>

AND <Indirizzo 1>

STO <indirizzo 6>

Lo stesso indirizzo arriva contemporaneamente sia a

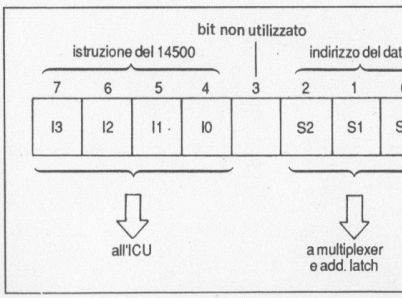


Fig. 5.27 *Ogni istruzione di questo sistema è formata selezionano l'istruzione dell'ICU e da altri 3 bit che selezio otto linee di ingresso e uscita*

che all'addressable latch: la selezione non deve però agire su di essi contemporaneamente. Solo nel caso di istruzioni STO e STOC l'indirizzo è riferito al dispositivo di uscita, nelle istruzioni LD, LDC, IEN, OEN, AND, ANDC, OR, ORC, XOR l'indirizzo è riferito alle istruzioni di ingresso. (Le istruzioni NOP0, NOPF, JMP , RTN, SKZ non agiscono sul dato, quindi non hanno bisogno di indirizzo). Posti

$$inhibit_{14512} = 0$$

$$reset_{14099} = 0$$

possiamo utilizzare i pin *disable* del 14512 e *write disable* del 14099. In corrispondenza delle istruzioni STO e STOC deve risultare

$$disable_{14512} = 1$$

(uscita del multiplexer in alta impedenza) e

$$write\ disable_{14099} = 0$$

(l'uscita selezionata dell'addressable latch copia il dato, le altre conservano memoria).
In occasione delle istruzioni LD, LDC, ... ORC, deve essere

$$disable_{14512} = 0$$

(l'uscita del multiplexer è uguale all'ingresso selezionato) e

$$write\ disable_{14099} = 1$$

(addressable latch in condizioni di memoria su tutte otto le uscite).

Poichè i segnali da inviare ai due dispositivi sono sempre l'uno complementare dell'altro può esser utilizzato un unico bit che entra "dritto" nel14512 e negato nel 14099. Il 14500 ha un pin che va alto solo in corrispondenza delle istruzioni di uscita: write. Ecco individuata la sua funzione: write serve per attivare la linea di ingresso o di uscita come in Fig. 5.28.
Write è basso anche in corrispondenza delle istruzioni NOP0, ...SKZ che non agiscono sul dato. Ma ciò non provoca problemi: in corrispondenza di tali istruzioni il multiplexer pone un dato sulla linea di ingresso (quello selezionato dai bit 0, 1 e 2 della memoria); ma il 14500 non lo legge.

Prendiamo in considerazione il clock. Possiamo usare l'oscillatore interno del 14500 e prelevare dal pin X1 il segnale da inviare in ingresso al contatore. La scelta della frequenza di clock e della modalità di collegamento richiedono precise valutazioni dei tempi di tutti i componenti: optiamo per il contatore 14040 e per la memoria 4016 (RAM statica da 2 k) già usati in esercitazione 3.E.
Dal data del 14500 apprendiamo che la sua *clock pulse width* -larghezza dell'impulso di clock- a 5 volt, è, nelle condizioni peggiori

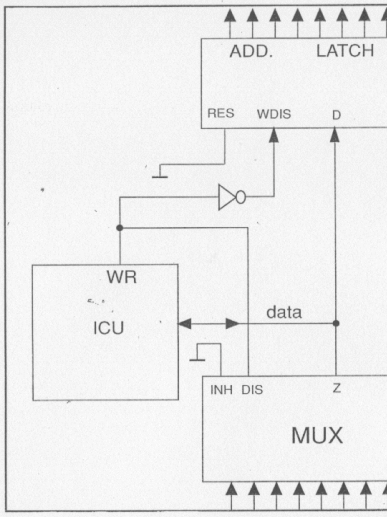


Fig. 5.28 *Il pin write serve per selezionare il dispositivo di uscita.*

Dalla timing in sesta pagina del data vedete che co teso il semiperiodo. Dunque il periodo del clock, pe portuno sia

$$T \geq 800\ ns$$

il che significa un clock di frequenza

$$f \leq 1{,}25\ Mhz$$

Dal data del 14040 ricavate che si tratta di una fr tabile da parte di quel contatore. Ma non è sufficie le timing waveform: ben precisi sono gli intervalli del clock in cui devono essere presenti istruzione Dobbiamo controllare che ciò avvenga. In esercitaz mo già considerato il blocco 14040 - 4016. Il 14040 te di discesa con un ritardo che, a 5 V, può essere

$$800\ ns$$

per l'uscita Q1, di

$$5\ \mu s$$

per l'uscita Q12. Dal momento in cui la RAM rice

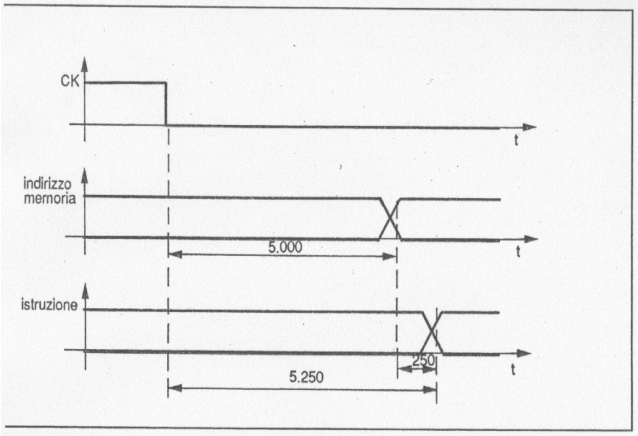


Fig. 5.29 *Dall'impulso al contatore al momento in cui questo fornisce l'indirizzo alla RAM intercorre un certo ritardo. Un ulteriore tempo è impiegato dalla RAM per fornire l'istruzione presente a quell'indirizzo.*

segnale di lettura a quello in cui fornisce il dato c'è un possibile ritardo di

250 ns

Dunque è

5.250 ns

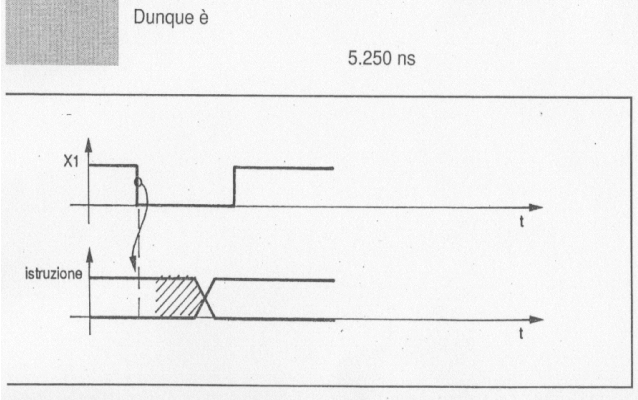


Fig. 5.30 *Se il fronte su cui il 14500 legge l'istruzione è lo stesso su cui si incrementa il Program Counter, allora il fetch avviene mentre non è ancora pronta la nuova istruzione e la precedente sta cambiando.*

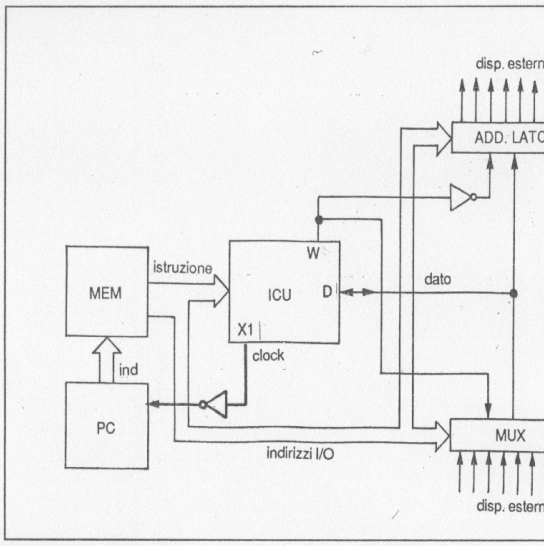


Fig. 5.31 *Il clock in uscita dall'ICU è negato prima di entra*
In questo modo i fronti di conteggio vengono sfasati.

dopo il fronte di discesa del clock che l'istruzione g
e la selezione ai dispositivi di input/output (Fig. 5.2
E' sul fronte di discesa di X1 che il 14500 cattura l
X1 viene utilizzato come ingresso del contatore nel r
il 14500 fa il fetch l'istruzione non è ancora presente
so mentre non vi è più nemmeno la precedente che
do (Fig. 5.30). Questa soluzione è da scartare.
Possiamo inviare al contatore il negato del clock con
Il clock del contatore risulta sfasato di un semiperi
quello su cui agisce il 14500. Un ulteriore ritardo è de
la porta NOT (per realizzare la quale dovrete usare
14011, oppure un inverter 14069).
Sul fronte su cui il contatore commuta fornendo l'inc
moria - vedi Fig. 5.32 - il 14500 sta eseguendo l'istr
dente. Quando arriva il fronte di discesa di X1 l'istru
pronta.
Come vedete dalla figura, fra il fronte di salita di X1
di un'istruzione valida all'ingresso dei pin I3 - I0 del
tercorrere un tempo

t = 5.350 ns

(un tempo maggiore occorre perchè sia presente s

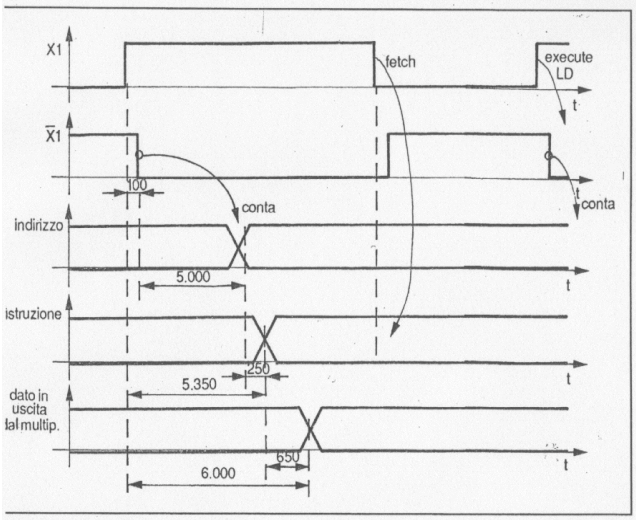


Fig. 5.32 *Il fronte su cui il contatore conta (dando così l'indirizzo alla memoria che, di conseguenza, fornisce l'istruzione) è sfalsato rispetto a quello su cui l'ICU effettua il fetch.*

gresso selezionato del multiplexer, ma questo interessa solo nel successivo *execute*).
Occorre dunque che il semiperiodo di clock sia superiore a 5.350 ns: possiamo dunque scegliere un clock di periodo

T = 20 µs

ovvero di frequenza

f = 50 khz

Si tratta di una frequenza molto minore di quella possibile per il 14500. Se volete realizzare un sistema programmabile più veloce dovete scegliere un altro program counter.
Qui accetteremo la conclusione cui siamo pervenuti, ricorrendo ad una resistenza, fra X1 e X2, di 500 kΩ.

Usare il negato di X1 come clock per il contatore fa sì che l'istruzione valida sia presente nel momento in cui il 14500 effettua il fetch. Dobbiamo controllare che tutto sia a posto anche ai fini dell'*execute*. Il dato in uscita dal multiplexer si presenta al pin 3 del 14500 con altri

650 ns

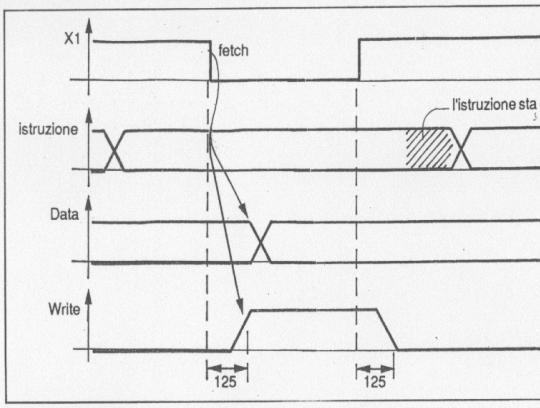


Fig. 5.33 *Subito dopo il fetch, RR viene inviato sul pin 3*

di ritardo rispetto al fronte di conteggio, ma ben in
to al fronte di discesa di x1. Quindi è tutto a posto
ni che accedono al dato in ingresso (LD, LDC, AN
Per le istruzioni che agiscono sui flag eventuali pr
gati a componenti collegati ai pin di flag, attualmer
ti. E per le istruzioni di uscita? La situazione è descr
subito dopo il fetch RR viene mandato sul pin 3 e w
te torna basso con

125 ns

di ritardo rispetto al fronte di salita di X1. Questo
co: se write fosse ancora alto mentre l'uscita della r
ziando a cambiare l'uscita potrebbe avvenire anch
errato. Il ritardo che la NOT aggiunge al tempo di pr
fronte di clock all'uscita dell'istruzione fa sì che sic
torni basso prima che al demultiplexer giunga u
zione.

5.6 PROGRAMMABLE LOGIC CONTROLLER

Per valutare i possibili campi di applicazione di un sistema programma
lo realizzato nella precedente esercitazione può essere di aiuto la pre
strial control unit handbook", della Motorola, che abbiamo riportato in
vi invitiamo a leggere.
In modo semplice ma efficace indica un'evoluzione della component
trolli industriali: dai relè ai componenti logici elettronici, ai PLC, *Progr*
Controller

Ma a volte è necessario: per esempio per scrivere il programma compilatore o inprete, oppure per risparmiare spazio di memoria o per rendere più veloce l'esecune.
r i PLC viene frequentemente usato un linguaggio ad alto livello di tipo grafico: si digna il circuito elettrico, a contatti, in grado di realizzare la funzione voluta.

r quanto riguarda il sistema che stiamo realizzando lo programmeremo, ovviamenin linguaggio macchina. Dobbiamo in primo luogo definire quale sia il suo linguagmacchina.
ni istruzione è formata da 4 bit (i più significativi) che contengono l'istruzione per :U e da 3 bit (i meno significativi) corrispondenti all' indirizzo di multiplexer e addresble latch.
sì, per esempio, se vogliamo caricare in RR il dato proveniente dall'indirizzo 2 (doaver abilitato l'ingresso) dovremo scrivere l'istruzione macchina:

0001 x010

'e al posto della x possiamo porre indifferentemente un "1" o uno "0").
programma scritto in questi termini risulta leggibile per la macchina ma ben poco ' noi. Si fa ricorso pertanto ad espressioni che corrispondano alle istruzioni del coe macchina ma siano per noi più facili da ricordare: codice mnemonico; per empio:

LD <2>

/ero

CARICA (load) DALL'INDIRIZZO 2

sercitazione 5.D

Compilate una tabella con tutte le istruzioni del sistema progettato. Accanto al codice macchina introducetene uno mnemonico.

Otto sono le possibili istruzioni LD:

LD <0>	load all'indirizzo 0
LD <1>	load all'indirizzo 1
LD <2>	load all'indirizzo 2
.....	.....
LD <7>	load all'indirizzo 7

Otto sono le possibili LDC, otto le possibili operazioni AND, e così via. In Fig. 5.35 abbiamo disegnato la tabella delle istruzioni. Nella colonna "mnemonico" trovate le diverse istruzioni; "i" è un indirizzo compreso fra 0 e 7. La colonna "operazione" indica il risultato dell'esecuzione dell'istruzione.
Nella colonna "codice" è indicata, bit per bit, l'istruzione macchina:

	MNEMONICO	OPERAZIONE	CODICE 7654 3 210
ISTRUZIONI DI I/O	IEN <i>	i ---> ff IEN	1010 X a
	OEN <i>	i ---> ff OEN	1011 X a
	LD <i>	i ---> RR	0001 X a
	LDC <i>	$\bar{i}$ ---> RR	0010 X a
	STO <i>	RR ---> EST i	1000 X a
	STOC <i>	$\overline{RR}$ ---> EST i	1001 X a
ISTRUZIONI LOGICHE	AND <i>	RR * i ---> RR	0011 X a
	ANDC <i>	RR * $\bar{i}$ ---> RR	0100 X a
	OR <i>	RR + i ---> RR	0101 X a
	ORC <i>	RR + $\bar{i}$ ---> RR	0110 X a
	XNOR <i>	se RR=Data, 1 --> RR	0111 X a
ISTRUZIONI DI SALTO E VARIE	SKIP	salta la successiva	1101 X XXX
	SKZ	se RR=0 salta la succ.	1110 X XXX
	NOP	nessuna operazione	0000 X XXX
			1100 X XXX
			1111 X XXX

Nota: l'effetto delle operazioni LD, LDC, AND, ANDC, OR, ORC, XNOR, è all'ipotesi IEN = 1; quello delle istruzioni STO e STOC all'ipotesi OEN = 1

Fig. 5.35 *Istruzioni del sistema programmabile che abb*
Le istruzioni del sistema non coincidono con quelle de
dell'istruzione l'indirizzo del dato.

ad "a" dovranno essere sostituiti i tre bit corrispon
zo. La "x" indica, al solito, un valore indifferente: no
re "vuoti" quei bit di memoria, ma non hanno effe
Nel sistema progettato i flag non sono utilizzati, no
ti a nulla. Dunque ai fini pratici le tre configurazioni
bella - corrispondenti alle istruzioni NOP0, NOPF,
fanno la stessa cosa: cioè niente. Le abbiamo inc
me di *No Operation*: nessuna operazione. (Alle vol
NOP può essere utile in fase di programmazione
L'istruzione SKZ fa sì che quella successiva non
se il contenuto di RR è zero. L'istruzione che abbia
SKIP fa sì che quella successiva venga sempre sal
de all'istruzione RTN del 14500 ma abbiamo pre
nome per indicarne l'effettiva, ed unica, funzione
lizzato).
Come vedete abbiamo suddiviso le istruzioni in gr