

Architettura degli elaboratori I
Modulo B,
A.A. 1999-2000

RICHIAMI SU ALCUNI ARGOMENTI SVOLTI NEL MODULO A

Scheda A

Il multiplexer (selettori di dati).

Un multiplexer serve a convogliare in un unico canale, informazioni provenienti da più sorgenti. La selezione della sorgente prescelta è affidata al codice binario presente sulle linee di selezione.

Consideriamo per semplicità un multiplexer "da quattro ad uno" che richiede due linee di selezione (figura 1).

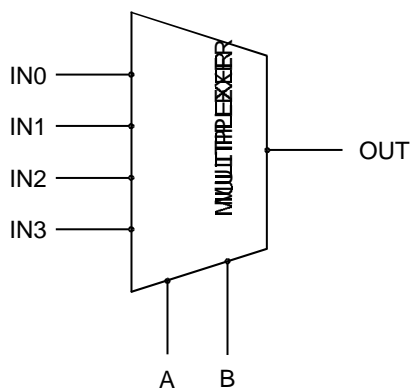


Fig. 1

La corrispondente tabella di verità è in figura 2:

B	A	COU	COU*	COU**
0	0	IN 0	1	0
0	1	IN 1	1	0
1	0	IN 2	1	0
1	1	IN 3	1	0

Fig. 2

IN0	IN 1	IN 2	IN 3	B	A	COU
0	X	X	X	0	0	0
1	X	X	X	0	0	1
X	0	X	X	0	1	0
X	1	X	X	0	1	1
X	X	0	X	1	0	0
X	X	1	X	1	0	1
X	X	X	0	1	1	0
X	X	X	1	1	1	1

Fig. 3

Dove le colonne OUT* ed OUT** si riferiscono ad una logica elettronica in cui lo stato logico "di riposo" è "0" o "1", rispettivamente. Se non si vuole lasciare adito a dubbi ed esprimere compiutamente il problema logico si può adottare la tecnica della figura 3:

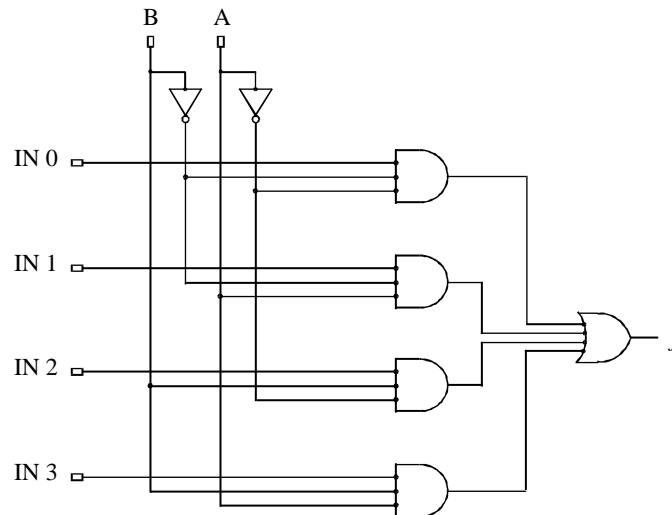


Fig. 4

La rete combinatoria che ne risulta è in figura 4.

Una alternativa più compatta e più moderna può essere la forma di figura 5 in cui si fa uso di buffer "tristate":

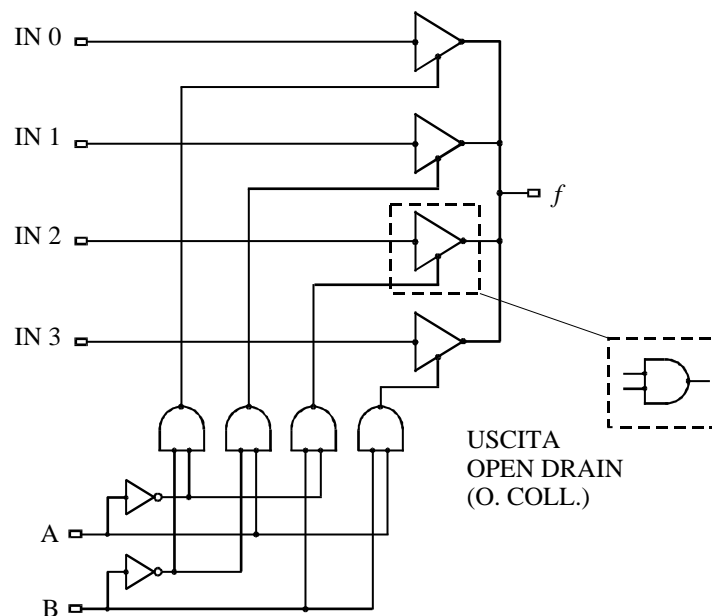


Fig. 5

La parte inferiore della rete non è altro che una decodifica dei due bit di indirizzo che, se realizzata con funzioni AND a tre ingressi, invece che a due, permette di inserire un ingresso di abilitazione che nello stato logico "0" disabilita tutti i buffer tristate.

Un'altra possibile alternativa è quella di usare delle funzioni AND, con uscita Open Collector (OC), se si usa una logica bipolare, oppure, Open Drain (OD), se si usa una logica MOS.

Si tratta in entrambi i casi di circuiti che impongono solo lo stato logico "basso" (0 Volt). Lo stato logico "alto" è affidato ad un resistore di "pull-up", aggiunto esternamente, che svolge nei confronti delle uscite degli AND la funzione che in un sistema meccanico svolge una molla di richiamo. L'azione del resistore è "elastica" e si adatta ad un eventuale livello basso. Un interruttore chiuso verso l'alimentazione quando incrocia l'azione con uno chiuso verso massa crea un "corto circuito" che danneggia le uscite dei circuiti integrati.

Quando sono collegate fra loro uscite OD, se nessuna delle funzioni impone un livello "basso" il resistore tiene la linea "alta".

In questo caso perché i quattro AND funzionino da OR (la struttura si chiama "wired OR") bisogna usare necessariamente la logica "negata".

Il tipo di situazione circuitale è riportato nella figura 6.

La struttura che ne viene fuori ha un ruolo fondamentale nella gestione dei trasferimenti dati attraverso quelle vere e proprie "autostrade" di collegamento tra le varie parti di un sistema che sono i "bus".

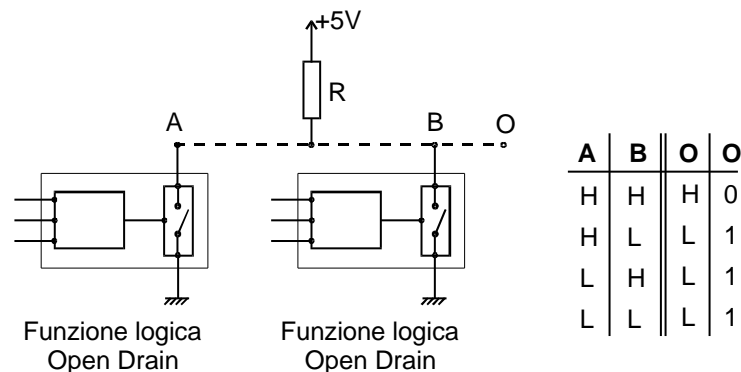


Fig. 6

Come vedremo, infatti, questo è l'unico modo corretto per la prima interazione logica tra unità tra loro completamente "asincrone".

Scheda B

Il demultiplexer:

IN	B	A	COU10	COU11	COU12	COU13
0	0	0	0	0	0	0
1	0	0	1	0	0	0
0	0	1	0	0	0	0
1	0	1	0	1	0	0
0	1	0	0	0	0	0
1	1	0	0	0	1	0
0	1	1	0	0	0	0
1	1	1	0	0	0	1

Figura 7

Le considerazioni fatte per il multiplexer si possono ripetere quasi perfettamente per la sua funzione duale, il demultiplexer.

Anche in questo caso la tabella di verità si può scrivere, in maniera non equivoca, nella forma di figura 7 dalla quale si perviene alla rete combinatoria di figura 8.

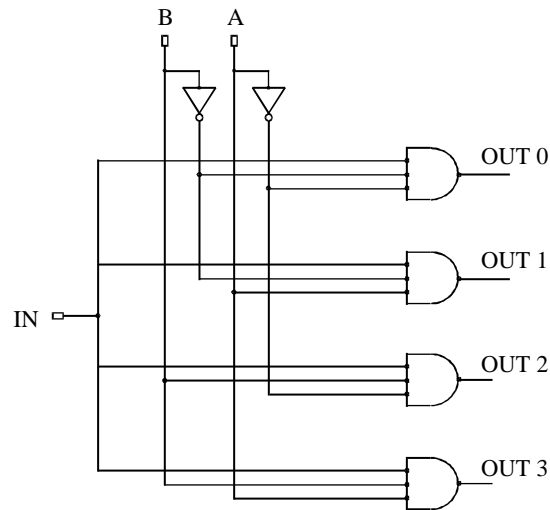


Fig. 8

Anche in questo caso esiste un'alternativa compatta che mette in evidenza che la combinazione dei bit di selezione viene decodificata per abilitare la porta AND di uscita per i segnali che arrivano dall'unico ingresso. La struttura è mostrata in figura 9.

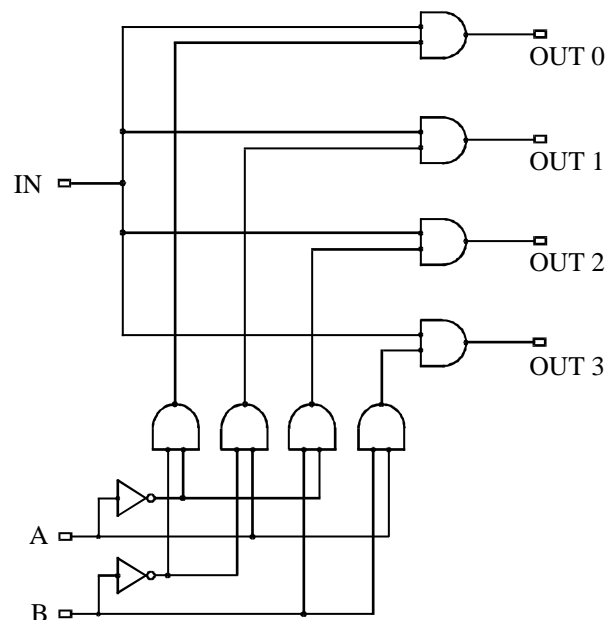


Fig. 9

Anche in questo caso è possibile sostituire gli AND di uscita con "*driver tristate*" ed inserire un ingresso di abilitazione nel decodificatore. In mancanza di abilitazione tutte le uscite saranno in "*alta impedenza*".

Scheda C

I flip-flop come macchine a stato:

Flip_flop Set/Reset

Abbiamo visto che il flip-flop Set/Reset è perfettamente riconducibile ad un automa di Moore col diagramma degli stati (diagramma a bolle) di figura 10.

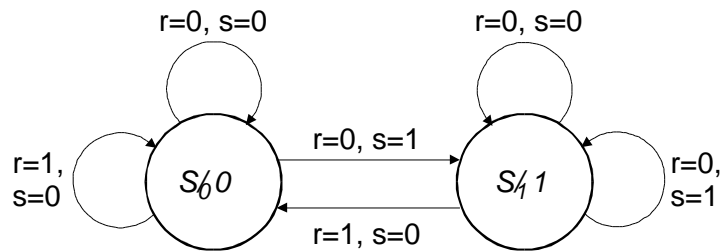


Fig. 10

La corrispondente tabella di flusso è mostrata in figura 11.

		<i>r, s</i>			
		0 0	0 1	1 1	1 0
<i>S</i>	<i>S</i> ₀	<i>S</i> ₀	<i>S</i> ₁	-	<i>S</i> ₀
	<i>S</i> ₁	<i>S</i> ₁	<i>S</i> ₁	-	<i>S</i> ₀

Fig. 11

Se si assegna ai due stati una codifica binaria coerente con le uscite, si ricava la seguente tabella delle transizioni che è anche mappa di Karnaugh.

<i>y (Q)</i>		<i>r, s</i>			
		0 0	0 1	1 1	1 0
<i>y' (Q')</i>	0	0	1	-	0
	1	1	1	-	0

Fig. 12

Con le riduzioni indicate in figura si ricava l'espressione dell'unica funzione che genera sia il prossimo valore logico dell'uscita, Q' , che il valore logico dell'unica variabile interna, y' , che caratterizza il prossimo stato:

$$Q' = \bar{R} ? S + \bar{R} ? Q = \bar{R} ? (S + Q)$$

Se si applica il teorema di De Morgan all'ultima espressione si ottiene:

$$Q' = \overline{R} \cdot (S + Q) = \overline{R + (\overline{S + Q})}$$

La realizzazione della rete sequenziale sincrona di un flip-flop Set/Reset è nella figura seguente:

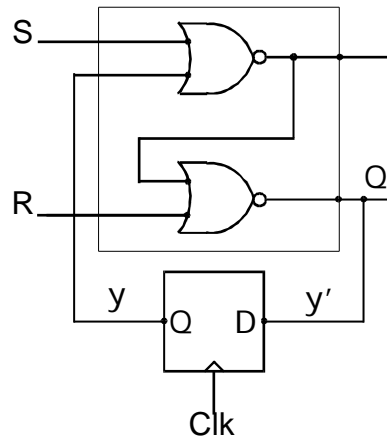


Fig. 13

E' evidente che si tratta dello stesso circuito del flip-flop asincrono in cui è ora inserito il flip-flop D sincrono. Il simbolo sintetico è:

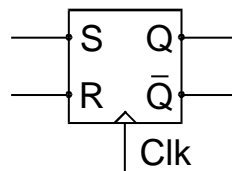


Fig. 14

Flip-flop "D"

Se torniamo al diagramma a stati di partenza, per ricavare il flip-flop di tipo "D", cioè un flip-flop con un solo ingresso, basta porre:

$$S = D \quad R = \overline{D}$$

$$Q = D \cdot (D + Q) = D + D \cdot Q = D$$

Il diagramma si riduce a quello mostrato in figura 15

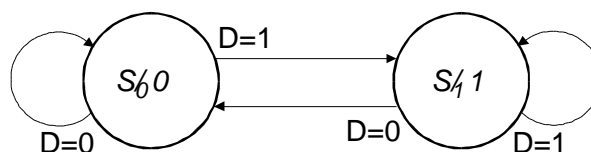


Fig. 15

Anche la tabella di flusso e quella di transizione si semplificano, assumendo la forma di figura 16.

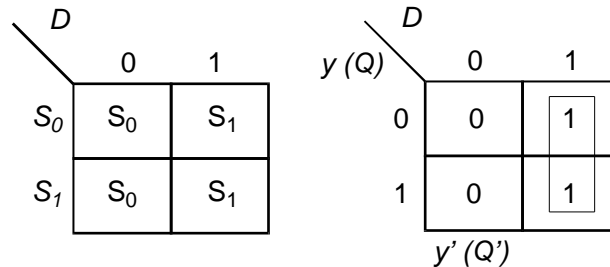


Fig. 16

La funzione d'uscita e quella del prossimo stato si riducono, com'era inevitabile, all'identità $y'=Q'=D$, la rete combinatoria diventa un semplice collegamento e la rete sequenziale coincide col FF di tipo D che memorizza il valore della variabile di stato nel prossimo stato.

Se invece vogliamo ottenere lo stesso flip-flop da quello Set/Reset nella sua versione sincrona, basta inserire sugli ingressi S e R le due uscite di un "demultiplexer" ad un solo bit il cui ingresso è la variabile D.

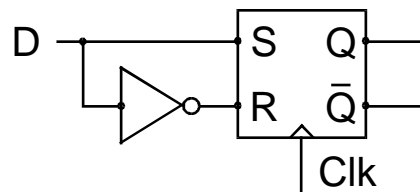


Fig. 17

Flip-flop da conteggio ("T").

Per ottenere il flip-flop da "conteggio" detto anche tipo "T" perché è caratterizzato dal fatto che commuta (in inglese *toggle*) ad ogni livello attivo (o ad ogni transizione attiva) del clock, il diagramma a stati secondo Moore deve essere quello di figura 18:

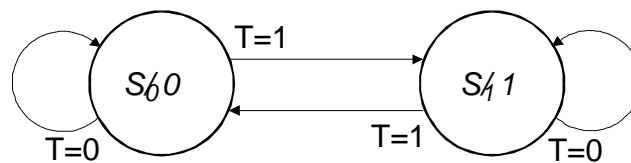


Fig. 18

Di conseguenza la tabella di flusso e tabella delle transizioni assumono la forma di figura 19

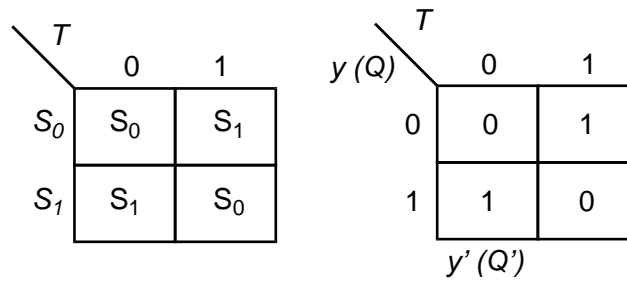


Fig. 19

Funzione d'uscita e funzione del prossimo stato coincidono e sono date da:

$$Q = T \bar{Q} + \bar{T} Q$$

La rete sequenziale corrispondente può essere messa oltre che nella consueta forma basata sul FF tipo D sincrono, anche in una forma alternativa che usa un FF Set/Reset:

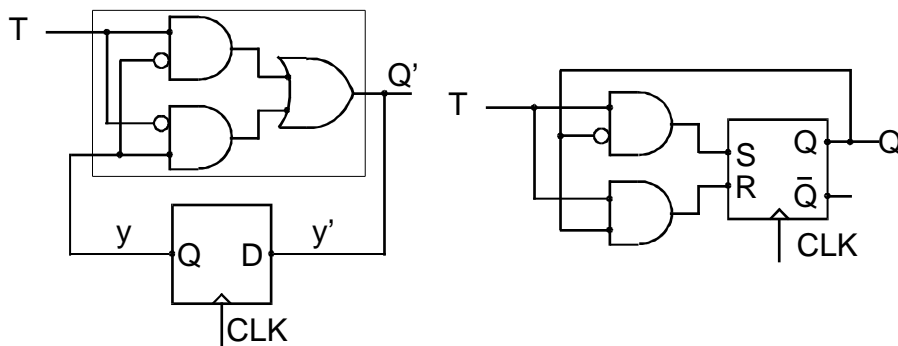


Fig. 20

L'ultimo tipo di flip-flop che resta è quello cosiddetto "J/K" che è anche il più duttile perché si presta ad essere trasformato con grande semplicità in uno degli altri tre. La sua tabella di transizione è in figura 21, il diagramma degli stati in figura 22.

J	K	Q'
0	0	Q
0	1	0
1	0	1
1	1	Q*

Fig. 21

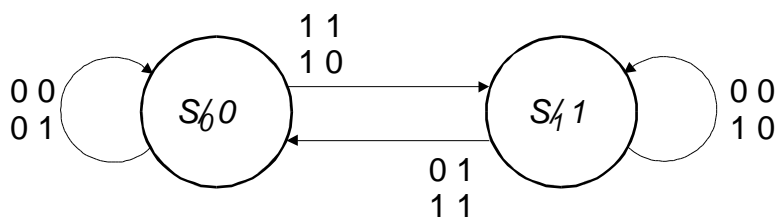


Fig. 22

Tabella di flusso e tabella di transizione sono in figura 23.

		J, K			
		0 0	0 1	1 1	1 0
S ₀		S ₀	S ₀	S ₁	S ₁
		S ₁	S ₀	S ₀	S ₁

		J, K			
		0 0	0 1	1 1	1 0
Q		0	0	1	1
		1	0	0	1

Fig. 23

Da cui si ricava la funzione d'uscita nel prossimo stato:

$$Q = J \bar{Q} + \bar{K} Q$$

Lo schema della rete sequenziale corrispondente è in figura 24.

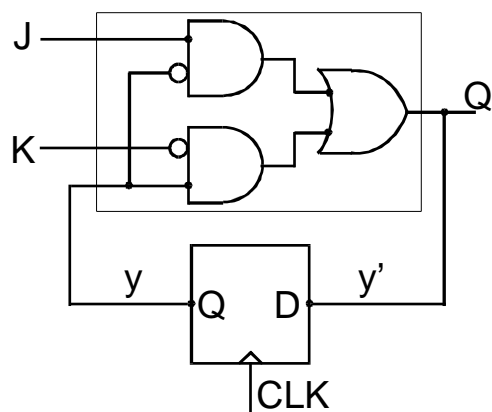


Fig. 24

Scheda D

Struttura delle memorie

Il chip di memoria

Una cella elementare di memoria può essere in linea di principio un qualsiasi elemento bistabile, anche asincrono.

Quando, però, si comincia a considerare la possibilità di costruire un banco di memoria di apprezzabile capacità, ci si rende conto che è **indispensabile usare flip-flop sincroni, dotati cioè di un ingresso di abilitazione, a livello o a transizione (edge-triggered).**

Una volta stabilito che si usano flip-flop sincroni è evidente che **la struttura ideale di una cella di memoria è un flip-flop di tipo D** perché la disponibilità di un solo ingresso per scrivere 0 od 1 è sicuramente la cosa più semplice.

Perché un flip-flop possa essere usato come cella di memoria deve essere equipaggiato con un minimo di "logica di servizio".

La struttura di una possibile cella di memoria è riportata in figura 25.

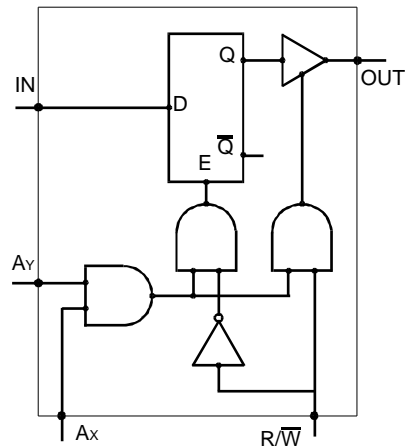


Fig. 25

Gli elementi aggiuntivi rispetto al flip-flop sono:

1. **Una doppia abilitazione, in assenza della quale la cella è paralizzata. Non vi si può scrivere, né vi si può leggere. L'uscita è nello stato di alta impedenza.** La doppia abilitazione consente la "*selezione a coincidenza*" che non è altro che un trucco per spezzare una matrice di selezione (decodifica) di grandi dimensioni (per esempio da 8 bit, in grado di selezionare una uscita su 256) in due da 4 bit (una uscita su 16). Il vantaggio, che è già molto grande con numeri piccoli, diventa enorme quando il numero di bit di indirizzo cresce ed è ancora maggiore se la selezione a coincidenza si usa su più dimensioni: tripla o quadrupla coincidenza.
2. **Una decodifica dello stato logico della linea R/W* che sceglie tra l'operazione di scrittura o quella di lettura.**
3. **Un driver tristate in uscita, che predispone la cella ad operare in parallelo con tante altre celle, senza problemi, purché una sola di esse sia abilitata in uscita.**

Il passaggio da una singola "*cella*" in grado di registrare un solo bit ad una "*locazione*", in grado di ospitare un byte o una parola, si fa replicando la struttura elementare tante volte quanti sono i bit richiesti:

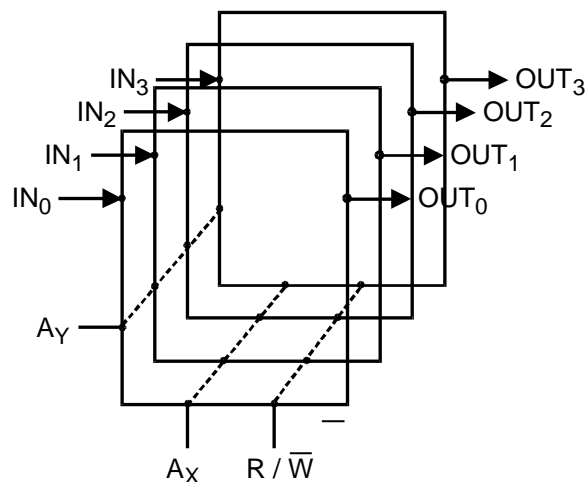


Fig. 26

E' importante osservare che, mentre ciascuna cella della locazione mantiene ingresso ed uscita indipendente, tutti i "servizi", linee di selezione e linea di operazione, sono in comune.

A questo punto per completare la struttura dell'elemento base di una memoria, il "chip", non resta che introdurre la "matrice di selezione". E' la struttura logica che decodifica i bit d'indirizzo, attivando una e soltanto una delle locazioni del chip.

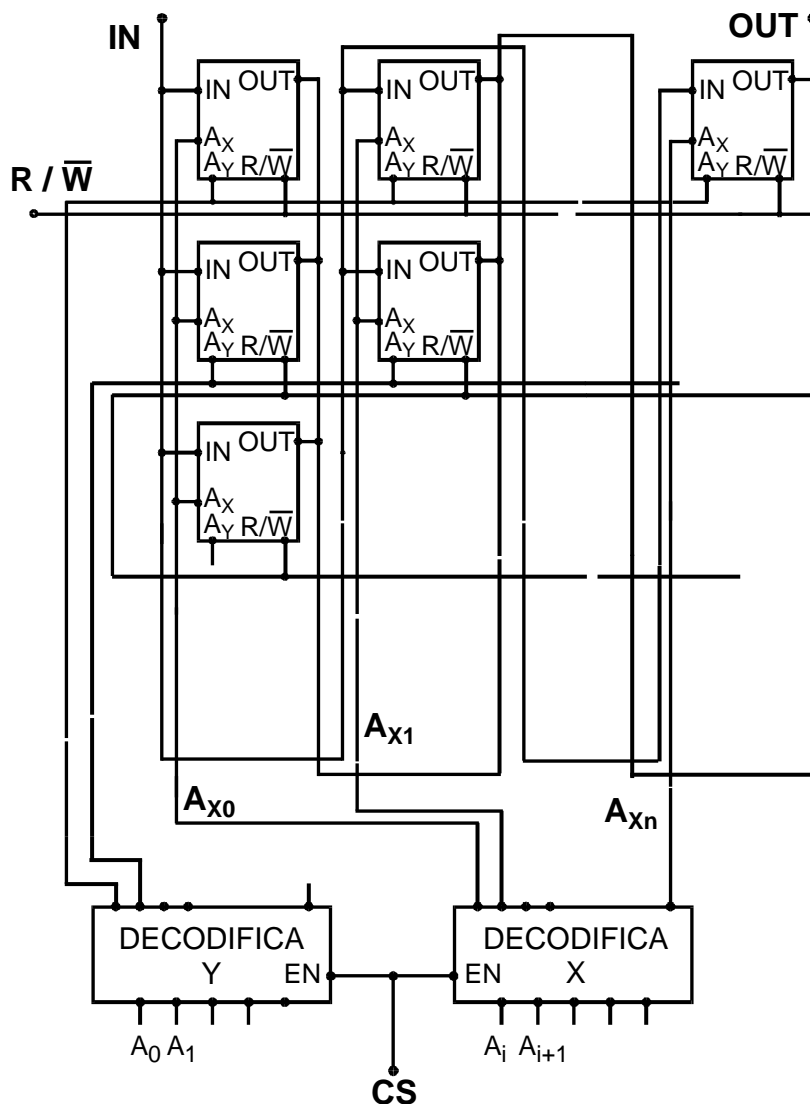


Fig. 27

Premesso che ciascuno dei blocchi della figura rappresenta una locazione da una sola cella ma potrebbe essere costituito anche da un certo numero di flip-flop rigidamente associati fra di loro con i "servizi" in comune. In tal caso le linee di uscita sarebbero tante quanti sono i bit della locazione.

Gli elementi di una riga hanno in comune uno degli ingressi di abilitazione e quelli di una colonna hanno in comune l'altro.

Ovviamente non esiste nessun vincolo a che la matrice sia quadrata. il numero di righe, p può essere diverso da quello delle colonne, q .

Il concetto di "pagina di memoria"

Quando si deve progettare un sistema di memoria bisogna come prima cosa scegliere il tipo di chip che si vuole usare. **La capacità del sistema (numero di locazioni) non potrà che essere la capacità del chip base moltiplicata per una potenza di due.**

Il numero di locazioni del chip o dell'insieme di chip necessario per avere la lunghezza di parola richiesta, costituisce quello che si chiama "*pagina di memoria*" (figura 27 bis).

- **I bit d'indirizzo, a partire da quello meno significativo, necessari per individuare una locazione all'interno di questo blocco elementare, vengono inviati in parallelo a tutti i blocchi, indipendentemente dal loro numero.**
- **I bit in più vengono inviati ad una decodifica le cui uscite sono collegate al piedino di abilitazione del chip (chip select).**

Se il sistema di memoria appena descritto, è realizzato su un circuito stampato ed il contatto di abilitazione della decodifica dei bit più significativi dell'indirizzo è portato sul connettore, esso può costituire un "*board select*" e può essere usato (come il chip select) per moltiplicare la capacità complessiva della memoria che può essere estesa a due, quattro, otto o più piastre.

E' una sorta di gioco delle scatole cinesi. Si usa a ripetizione lo stesso artificio.

Scheda E

Tecniche alternative per la sintesi di una funzione booleana

La tecnica base per la sintesi di una funzione booleana è quella di estrarre dalla tabella di verità la forma canonica disgiuntiva o congiuntiva e, dopo averla minimizzata, implementarla con le funzioni logiche elementari disponibili.

Abbiamo già visto che esistono soluzioni alternative, quella che usa una decodifica ed un OR, quella che usa un multiplexer con un numero di bit di selezione pari alle variabili o ridotto di una unità.

Tutte queste tecniche hanno in comune la caratteristica che la funzione viene implementata direttamente in forma canonica.

Abbiamo visto infatti che, **qualora la funzione non sia in forma canonica le espressioni algebriche che non contengono tutte le variabili vanno scomposte in prodotti completi**

Tabella funzione (Look-up table)

Esiste un'altra tecnica di sintesi di una funzione booleana che è ancora più semplice e diretta di quelle viste, si applica registrando in una memoria con un adeguato numero di locazioni direttamente la tabella di verità.

Supponiamo di avere una funzione di 4 variabili, assegnata attraverso la sua tabella di verità. Se si dispone di una memoria, meglio se si tratta di una *Read Only Memory (ROM) programmabile (PROM)*, a 16 locazioni da 1 bit, basta **scrivere in ciascuna locazione, corrispondente ad una combinazione delle variabili interpretata come indirizzo, "0" o "1", secondo il contenuto della tabella di verità e la funzione è implementata.**

Ponendo sulle linee d'indirizzo una combinazione delle variabili ed attivando la linea di lettura si promuoverà in uscita il contenuto della cella corrispondente e, cioè 0 o 1 secondo la tabella di verità.

Nella figura 28 è rappresentata l'implementazione di una funzione con la tecnica della "look-up table"

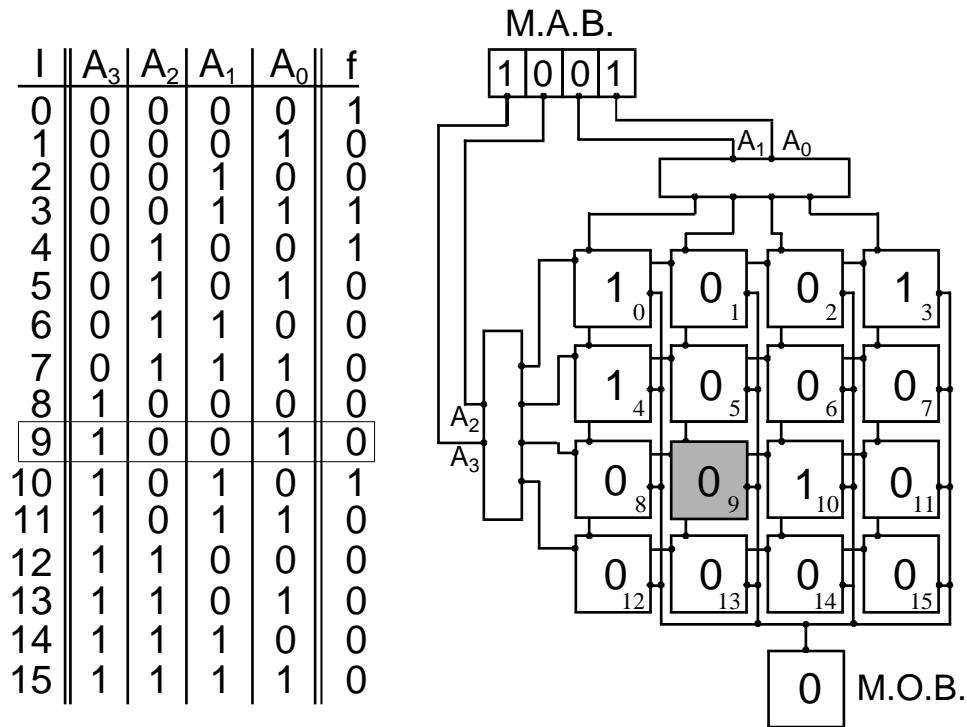


Fig. 28

Il registro Memory Address Buffer ricevendo la combinazione delle variabili d'ingresso $A_3 A_2 A_1 A_0$, indirizza la cella 9 e nel registro Memory Output Buffer compare il contenuto della locazione che è 0.

E' ovvio che l'ingresso del M.O.B. è collegato alla linea che collega tra loro i buffer tristate di uscita di tutte le locazioni mono-cella della memoria. La cosa è legittima perché il sistema di selezione garantisce che non saranno mai attivate contemporaneamente due celle.

Scheda F

Automi a funzionamento non completamente specificato.

Le funzioni del prossimo stato F e quella d'uscita G possono non essere definite in tutti i punti del dominio $Y \times X$, in altre parole possono esistere coppie di combinazioni delle variabili d'ingresso e di stato attuale dell'automata che non hanno definito il prossimo stato, la combinazione delle variabili d'uscita od entrambi.

In questi casi la definizione di "equivalenza" tra stati ed il processo di minimizzazione che ne è stato derivato debbono essere profondamente rivisti.

Per questo tipo di automa si introduce un concetto diverso che prende il posto dell'equivalenza.

Si parla di “compatibilità”.

Lo stato S_1 dell'automa A_1 si dice compatibile con lo stato S_2 dell'automa A_2 se per qualsiasi sequenza d'ingresso applicata ad A_1 nello stato S_1 ed ad A_2 nello stato S_2 si ottengono due sequenze d'uscita che coincidono ovunque entrambe sono specificate. In parole povere se in una delle sequenze d'uscita appare il segno “-“ che significa “non definito” esso è, ovviamente, compatibile sia con uno “0” che con un “1” nell'altra sequenza.

La compatibilità tra stati si indica col simbolo “ \approx ” e gode della proprietà riflessiva e di quella simmetrica ma non di quella “transitiva”.

Per rendersene conto basta far riferimento al semplice automa la cui tabella di flusso è riportata qui sotto.

Stato interno	Ingresso	
	0	1
A	A, 1	A, 1
B	A, -	A, -
C	A, 0	A, 0

Fig. 29

E' evidente, infatti, che, qualunque sia la sequenza d'ingresso, nello stato A la sequenza d'uscita è: 1111.... Mentre nello stato B è: -111.... ed in quello C è: 0111....

Se ne deduce che $A \approx B$ e $B \approx C$ ma A e C non sono tra loro compatibili.

Da ciò si deduce che se si costruiscono le “classi di compatibilità” tra gli stati dell'automa esse non sono necessariamente disgiunte, infatti nel caso dell'esempio, le due classi sono (A, B) e (B, C).

Se in un automa si sostituiscono tutti gli stati di una classe di compatibilità con uno di essi si ottiene un automa che può essere usato in sostituzione del primo ma solo a determinate condizioni. Se, infatti, si opera la trasformazione sulla tabella di flusso precedente si ottiene:

Stato interno	Ingresso	
	0	1
(A, B)	, 1	, 1
(B, C)	, 0	, 0

Fig. 30

Come si può facilmente verificare la sequenza di uscita dallo stato del nuovo automa è: 1111.... e quella da è 0111.... e la tabella è completamente specificata.

Per tenere conto di queste considerazioni si introduce il concetto di “copertura”.

Si dice allora che un automa A_2 copre un automa A_1 se per ogni stato S_i di A_1 esiste uno stato S_j di A_2 tale che per qualsiasi sequenza d'ingresso applicata ad A_1 in S_i ed ad A_2 in S_j si generano sequenze d'uscita uguali ovunque la sequenza generata da A_1 è specificata.

Tornando all'esempio si può osservare che, le sequenze d'uscita della macchina coprente sono più densamente specificate di quelle della macchina coperta e, per effetto della non separazione delle classi di compatibilità, la corrispondenza tra gli stati dei due automi delle figure precedenti può essere:

A , B , C

Oppure:

A , B , C

Il problema della minimizzazione di un automa non completamente specificato è legato, dunque, alle scelte fatte nella suddivisione degli stati tra le classi di compatibilità non disgiunte.

Il procedimento è simile a quello degli automi completamente specificati e parte da una **definizione operativa di compatibilità** tra gli stati che riecheggia quella dell'equivalenza.

Due stati si dicono "*compatibili*" se, per ogni possibile combinazione d'ingresso X_h , le combinazioni di valori logici delle variabili d'uscita, Z , date per i due stati $Y_i = S_i$ e $Y_j = S_j$ dalla funzione G sono uguali, ovunque le uscite sono entrambe specificate ed i relativi prossimi stati, dati dalla funzione F sono "*compatibili*", ovunque entrambi sono specificati.

L'individuazione delle classi di compatibilità, tralasciando considerazioni generali sulle complicazioni che possono nascere dalla possibilità di attribuire un nodo ad una classe o ad un'altra, si svolge usando una tabella a scala simile a quella usata per gli automi completamente specificati.

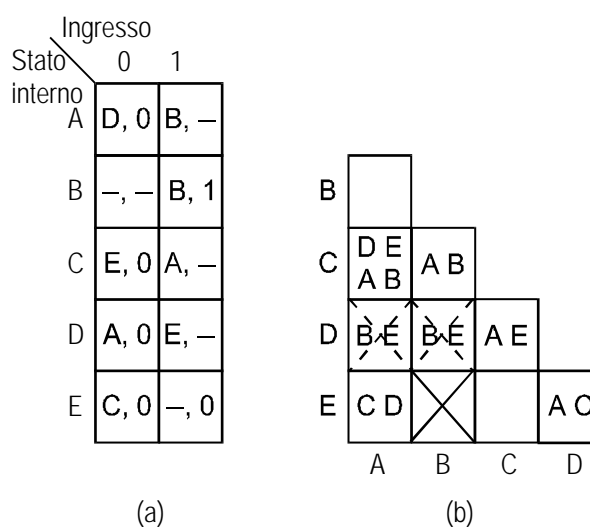


Fig. 31

In figura 31 sono presentati la tabella di flusso di un automa non completamente specificato e la tabella a scala che se ne estrae per rilevare le compatibilità degli stati. Le due caselle contrassegnate da una croce tratteggiata corrispondono a coppie di stati la cui condizione di compatibilità inizialmente ipotizzata è risultata poi non confermata.

Le compatibilità che emergono sono:

$$A \approx B, A \approx C, A \approx E, B \approx C, C \approx D, C \approx E, D \approx E$$

Le classi di compatibilità, basate sulla mutua compatibilità degli stati sono:

$$(A, B, C); (A, C, E); (C, D, E)$$

A questo punto è possibile costruire un automa compatibile A_c che può essere usato in luogo di A, perché "copre" A, ma non è necessariamente l'automata minimo che gode di questa proprietà.

La tabella di flusso dell'automata che ne risulta è riportata nella figura seguente:

Stato interno	Ingresso	
	0	1
(A, B, C)	, 0	, 1
(A, C, E)	, 0	, 0
(C, D, E)	, 0	, 0

Fig. 32

In realtà A_c gode anche di un'altra proprietà, quella della "*chiusura*". Questa caratteristica fa riferimento al fatto che per ciascuna combinazione d'ingresso i "*prossimi stati*" di tutti gli stati di una classe di compatibilità, se specificati, **appartengono tutti ad una stessa classe di A_c** .

Infatti, con riferimento alla tabella di flusso originaria, si può constatare, ad esempio, che da A, B, e C, per ingresso "0", si va a, D, -, E e gli stati D ed E sono entrambi della classe di compatibilità indicata con ed analogamente per le altre due classi.

L'unica caratteristica che manca ad A_c perché costituisca quello che cerchiamo è la "*minimalità*" che però non può essere raggiunta attraverso un procedimento codificato e va cercato volta per volta per tentativi.

Nel caso dell'esempio, si può facilmente verificare che l'automata che gode delle tre caratteristiche che deve possedere *l'automata compatibile minimo* e cioè: "*copertura*", "*chiusura*" e "*minimalità*" è quello che ha due sole classi di compatibilità:

$$(A, B, C); (D, E)$$

La relativa tabella di flusso è riportata in figura 33.

Stato interno	Ingresso	
	0	1
(A, B, C)	, 0	, 1
(D, E)	, 0	, 0

Fig. 33

E' chiaro che il progettista deve opportunamente valutare la convenienza di cercare per tentativi un automa minimo che magari abbassa solo di una unità od al massimo due il numero di stati, senza magari produrre nessun vantaggio in termini di numero di variabili di stato.

E' già stato detto, infatti, che **il numero di stati deve essere dimezzato per ridurre di una unità il numero di variabili.**

Scheda G

Interconnessione tra registri.

Esistono molte soluzioni per trasferire informazioni tra i registri interni di un sistema, o tra registri di sistemi diversi ma tutte le strutture si possono considerare intermedie fra le due scelte estreme, da una parte la connessione indipendente di ciascun registro a qualsiasi altro registro (collegamenti punto a punto) con possibilità quindi di attivare contemporaneamente tutti i trasferimenti che si desiderino, purché non interessino in maniera incompatibile uno stesso registro (parallelismo assoluto), dall'altro una unica interfaccia universale (bus) che consenta la connessione di qualsiasi registro a qualsiasi altro registro (o ad un gruppo di registri) ma con un solo trasferimento per volta (soluzione puramente seriale). La prima corrisponde alla massima complessità hardware, la seconda è improntata ad una grande economia hardware e duttilità ma condiziona fortemente il funzionamento della rete

Nel primo caso la struttura logica che si utilizza intensivamente è il multiplexer . La rete si dice di tipo "mesh" e si possono effettuare allo stesso tempo tutti i collegamenti. Si può anche leggere e scrivere nello stesso registro se è del tipo master/slave, così come, ovviamente, si può trasferire lo stesso contenuto da un registro in più registri.

Per rendersi conto del funzionamento di un trasferimento dati da un registro ad un altro basta considerare che, a parte i bit di selezione di eventuali multiplexer di interconnessione, i livelli logici dei registri si propagano lungo le linee di collegamento solo quando è attiva la linea di abilitazione in uscita (spesso del tipo "tristate"), mentre un nuovo contenuto può essere registrato solo quando è attiva l'abilitazione in ingresso. Alcuni registri possono avere contemporaneamente attive sia l'abilitazione in ingresso che quella in uscita e possono, quindi, diventare "trasparenti".

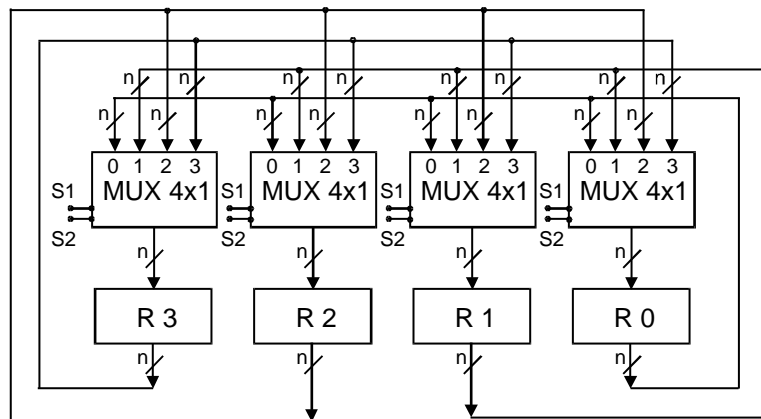


Fig. 34

Se supponiamo di avere quattro registri da n bit e gli ingressi di ciascuno di essi sono collegati a tutte le uscite da multiplexer "quattro in uno" ad n bit (figura 34). Questa rete, detta di tipo "mesh", consente di trasferire il contenuto di ciascun registro in qualunque altro, senza interferire con un altro contemporaneo trasferimento che non collida col primo alla destinazione.

Il difetto sostanziale della struttura, che si manifesta sempre di più quando il numero dei registri aumenta, è il numero di funzioni logiche necessarie per realizzarla.

Una soluzione più economica ma che limita il numero di trasferimenti effettuabili allo stesso tempo è mostrato in figura 35.

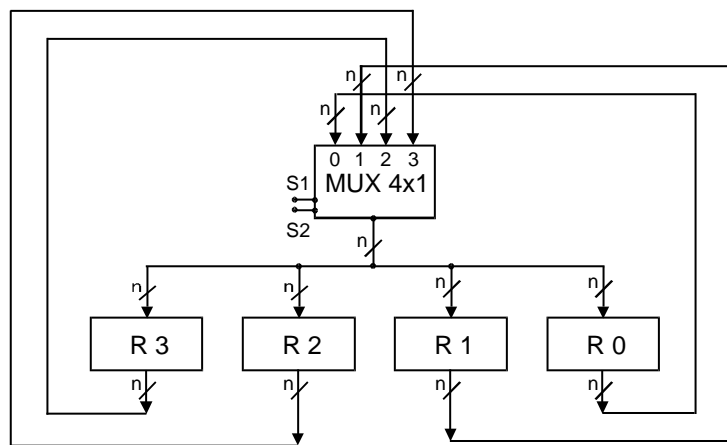


Fig. 35

Una volta che il trasferimento in corso (per esempio R1 → R0) ha impegnato il multiplexer non è possibile effettuare altre operazioni che coinvolgano la rete a valle del multiplexer ma rimane possibile utilizzare il contenuto dei registri, anche di quello che è la destinazione del trasferimento in corso, se i registri sono del tipo Master/Slave, per eventuali trasferimenti verso registri che fanno parte di altri gruppi (per esempio R7 → R2).

Questa possibilità è completamente esclusa quando si adotta la modalità di interconnessione più economica che è quella detta "bus" mostrata in figura 36.

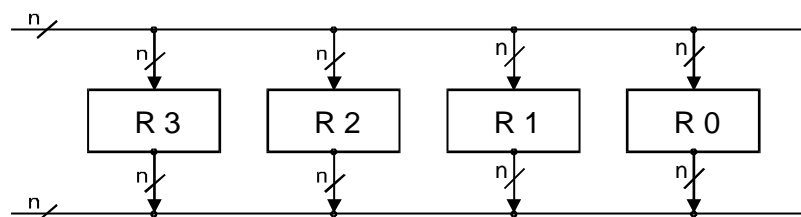


Fig. 36

Un "bus dati, formato da tante linee quanti sono i bit dei registri, collega tra loro ordinatamente sia gli ingressi che le uscite dei registri. Le linee sono bidirezionali e le uscite debbono necessariamente essere del tipo "tristate" o "open drain".

Nel caso si usino buffer tristate è necessario che la logica di gestione garantisca che non verranno attivate contemporaneamente le uscite di due registri.