Capitolo 2

Sistemi Logici Complessi: Memorie e Matrici Logiche Programmabili

2.1 Memorie

2.1.1 Introduzione

Un utilizzo sempre più diffuso nei sistemi digitali stanno acquistando le memorie. Una memoria è us sistema, organizzato in celle elementari, dove è possibile immagazzinare, per tempi più o meno lunghi, informazioni digitali quali bits, bytes, etc.. Le memorie possono essere di tipo "sequenziale" o di tipo "ad accesso casuale". Un esempio di memoria sequenziale è il nastro magnetico. In una memoria di questo tipo occorre leggere tutta l'informazione memorizzata, in modo appunto sequenziale, prima di poter accedere a quella voluta. In una memoria ad accesso casuale è invece possibile accedere all'informazione desiderata specificando la locazione o posizione dove essa è stata scritta (ciò presuppone l'esistenza di un "registro" dove, per ciascun gruppo di dati, cioè per ciascun "file", sia stato preventivamente registrato l'indirizzo).

Un esempio di memoria ad accesso casuale è costituito dai dischi floppy o anche dai dischi rigidi (Hard Disks) dei computer. In questi la memorizzazione dell'informazione è basata su processi di magnetizzazione.

L'elaboratore di un computer ha bisogno anche di memorie, sia permanenti come le ROM (Read Only Memories o memorie a sola lettura) che temporanee come le RAM (Random Access Memories o memorie ad accesso casuale) che consentano tempi di accesso e trasferimento dati molto più veloci di quelli caratteristici dei dischi magnetici.

Le ROM incontrano le applicazioni più disparate, che vanno dall'immagazzinamento del codice (istruzioni) che vengono eseguite all'accensione di un calcolatore, alla memorizzazione di tabelle che forniscono il valore di funzioni predefinite (quali sin(x), cos(x), log(x) etc), alla memorizzazione di sequenze predefinite di impulsi e di forme d'onda, alla memorizzazione dei codici corrispondenti ai vari caratteri (codice BCD, codice EBCDIC, ASCII..) e via dicendo.

Le RAM sono memorie dinamiche, cioè il loro contenuto può essere modificato da programmi. Esse sono di grande utilità nella memorizzazione temporanea di sequenze o istruzioni generate dal codice che viene eseguito da un calcolatore.

2.1.2 Struttura di una ROM

La struttura della più diffusa tra le memorie statiche è quella della ROM. Illustriamo il suo funzionamento con un esempio concreto. Immaginiamo di voler effettuare rapidamente il calcolo del fattoriale (m) di un intero n (m=n!) con n minore di un valore prefissato. Possiamo effettuare il calcolo una volta per tutte, in corrispondenza ad ogni valore di n, e poi memorizzare il risultato in una ROM. Quando si abbia bisogno del fattoriale di un certo intero sarà sufficiente fornire in forma binaria tale numero ad una ROM come quella di figura 2.1 (dove, per motivi pratici, ci si è limitati ad interi $\leq$ 5). La ROM fornirà sui suoi 7 piedini d'uscita il fattoriale m richiesto.

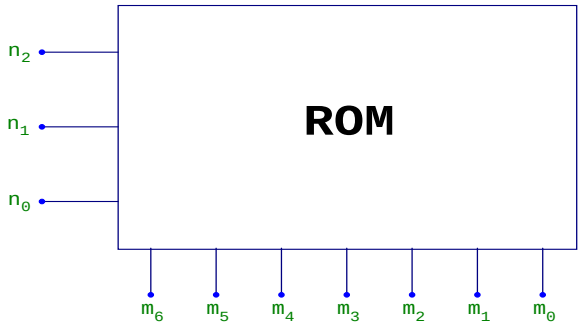

Figura 2.1:

É facile vedere che la ROM dovrà effettuare la conversione indicata nella tabella 2.1.

n	n_2	n_1	n_0	m	m_6	m_5	m_4	m_3	m_2	m_1	m_0
0	0	0	0	1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	0	0	0	0	1
2	0	1	0	2	0	0	0	0	0	1	0
3	0	1	1	6	0	0	0	0	1	1	0
4	1	0	0	24	0	0	1	1	0	0	0
5	1	0	1	120	1	1	1	1	0	0	0

Tabella 2.1:

Tale conversione può essere, con riferimento all figura 2.2, effettuata nel modo seguente. Facendo uso di un decodificatore da 3 a 7 linee, si attiva, per un dato valore di n impostato in ingresso, una delle sette linee di uscita. Ad esempio, in corrispondenza al codice 011 ingresso, si attiva la linea numero 3.

A questa linea un apposito codificatore associa (cioè genera in uscita) il codice 0000011, cioè il valore decimale 6. Analoga corrispondenza viene stabilita per gli altri valori applicati in ingresso, indicati nella tabella.

La struttura dei circuiti di decodifica e di successiva codifica può esser realizzata sulla falsariga di quelle indicate rispettivamente nelle figure 2.3 e 1.19. Da notare che il circuito di figura 2.3 è relativo ad una decodifica da 4 a 10 linee, mentre nel

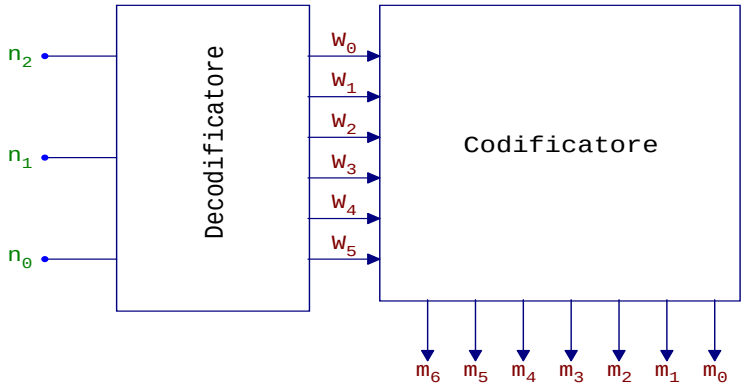

Figura 2.2:

nostro esempio attuale è sufficiente un decodificatore da 3 a 7 linee. Il circuito di decodifica potrebbe essere quello mostrato nella parte sinistra della figura 2.4.

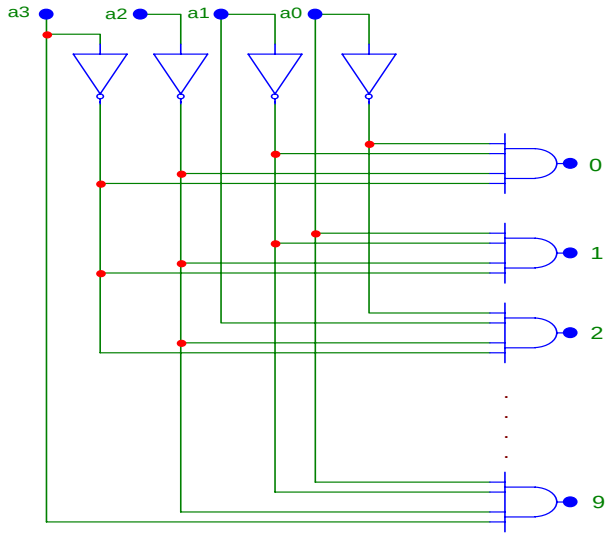

Figura 2.3:

Infatti, è immediato verificare che le porte AND mostrate in figura implementano la funzione di *decoding* indicata nelle prime quattro righe della tabella. Ad esempio, per $n \equiv n_2, n_1, n_0 = 0, 1, 1$, si vede che è $W_3 = \overline{n}_2 \cdot n_1 \cdot n_0 = 1$, mentre $W_0 = W_1 = W_2 = W_4 = W_5 = 0$.

La parte destra della medesima figura svolge la funzione di codifica. Ad esempio possiamo vedere che, nel caso appena esaminato, in cui $W_3 = 1$, i diodi collegati alle linee (colonne) m_1 ed m_2, conducono, con che $m_1 = m_2 = 1$, mentre le rimanenti colonne rimangono collegate a massa (livello 0). Avremo quindi:

$$m = 0, 0, 0, 0, 1, 1, 0 = 6_{10}$$

che è appunto 3!. É facile verificare il funzionamento in tutti gli altri casi.

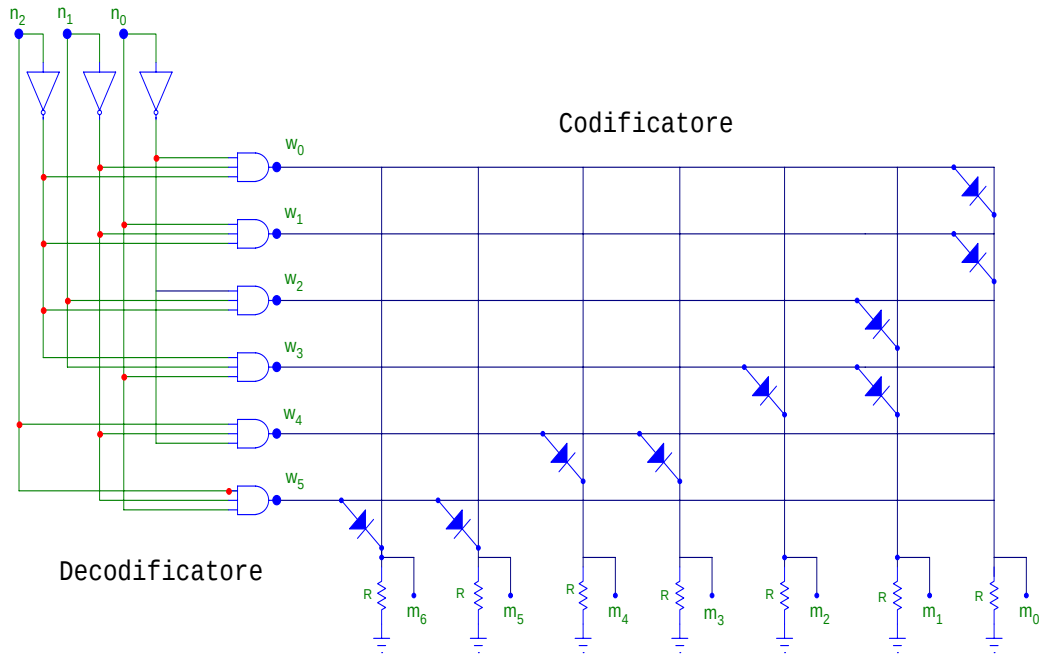

Figura 2.4:

Notiamo che l'architettura utilizzata per il decodificatore implica una porta AND per ciascuna linea (le linee vengono chiamate *word-lines*, le colonne *bit-lines*).

Tale architettura diviene costosa se il numero di word-lines è elevato. Per ovviare a tale problema, si può far uso di una decodifica del tipo di quella di figura 2.5, relativa al caso di una ROM da $512 \times 4 = 2048$ *bits* (codice in ingresso di 9 bit con codice d'uscita di 4 bit). Il decodificatore indicato nel blocco a sinistra in figura effettua la decodifica dei soli sei bit meno significativi, corrispondenti a 64 righe ($W_0 - W_{63}$). La matrice di codifica che segue ha 32 colonne (matrice di $64 \times 32 = 2048$ celle di memoria). Poichè in uscita sono richiesti quattro bit alla volta, i quattro blocchi selettori che seguono il codificatore, ciascuno con otto linee di ingresso, selezionano, per ciascun valore dei bit più significativi applicati in ingresso, una delle otto linee.

Tale tipo di indirizzamento è noto come indirizzamento X-Y o bidimensionale. Nell'esempio indicato sono necessarie solo 64 porte AND (o NAND) per la decodifica dei sei bit meno significativi. In più sono necessarie altre porte (36) per i selettori indicati. Il numero di porte richiesto è quindi pari a 100, da confrontare con le 512 necessarie nel caso di una struttura del tipo di quella indicata in figura 2.2.

Notiamo infine che, anche se per semplicità abbiamo discusso le matrici di codifica basandoci sull'esempio della matrice a diodi, questa è stata oramai sostituita da matrici in cui la funzione dei diodi è svolta da transistor o da MOSFET.

2.1.3 ROM programmabili

Le ROM che abbiamo descritto nella sezione precedente sono fornite, su specifica dell'utente, dai produttori. Le caratteristiche di tali ROM, cioè l'informazione in esse memorizzata, sono definite al momento della realizzazione e non possono essere alterate successivamente. É possibile far uso, in alternativa, di ROM programmabili

dall'utente, note come PROM (field-programmable ROM). Una PROM, al momento della produzione, contiene tutte le possibili connessioni (diodi o transistor) interne. L'utente può, facendo uso di un'apposito hardware, bruciare o eliminare tutte le connessioni non volute, in modo da ottenere la tabella di conversione (corrispondenze ingressi-uscite) desiderata. L'hardware necessario è noto come "programmatore di PROM" . In una generica PROM, una volta effettuata la programmazione non è più possibile modificarla.

Esistono tuttavia delle PROM in cui la cancellazione dell'informazione e quindi la modifica della programmazione, è possibile. In questa varietà di PROM (note come EPROM o "erasable PROM") la cancellazione dei dati avviene per l'esposizione alla luce ultravioletta. Ciò richiede che il chip venga posto, per tempi relativamente lunghi (circa un'ora), in un apposito dispositivo dove sia presente una sorgente UV. Dopo la riprogrammazione, la parte sensibile della EPROM deve esser protetta (mediante un nastro adesivo che non trasmetta l'UV) per evitare che l'eventuale esposizione a radiazione UV (quale quella emessa dal Sole) possa cancellare l'informazione.

Per facilitare il processo di riprogrammazione sono state realizzate delle EPROM in cui la cancellazione dei dati viene effettuata con l'applicazione di una tensione dell'ordine di 10 V agli elementi che la costituiscono. Queste sono note come "electrically erasable PROM's" o anche $EEPROM$ o E^2PROM.

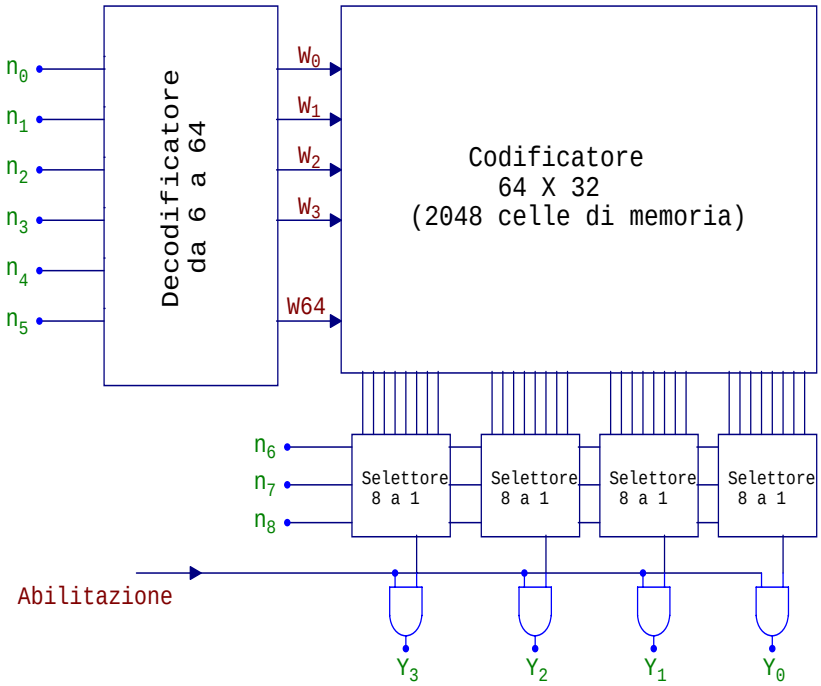

Figura 2.5:

2.2 Memorie ad accesso casuale

Le ROM, come detto, sono utilizzate in tutte quelle applicazioni in cui sia necessario disporre di una memoria che conservi permanentemente l'informazione. In moltissime situazioni è invece necessario modificare di continuo l'informazione memorizzata. Sono quindi necassarie delle memorie che possano esser *scritte* oltre che *lette*. Questo è ad esempio il caso di un computer, che elabora dei dati e memorizza i risultati dell'elaborazione per una successiva stampa o trasmissione. In tali casi si fa uso di *memorie ad accesso casuale*, note come RAM, o delle numerose varianti di queste.

La denominazione *accesso casuale* sta ad indicare il fatto che è possibile specificare un generico indirizzo in ingresso per avere in uscita la parola memorizzata a tale indirizzo. Ciò distingue questo tipo di memoria da un nastro magnetico, in cui occorre svolgere tutta la parte di nastro che precede quella dove la parola voluta è stata memorizzata. A rigore, anche la ROM è una memoria ad accesso casuale, nel senso appena detto per la RAM.

Esistono molti tipi di RAM diverse. Le SRAM (Static RAM) hanno come elemento di memoria il Flip-Flop. Esse conservano l'informazione memorizzata fino a che non si interrompa l'alimentazione. Le DRAM (Dinamic RAM) sfruttano come elemento di memoria la carica accumulata su di un dispositivo semiconduttore, integrato in un chip. Tale cella di memoria è assimilabile ad un minuscolo condensatore, di dimensioni micrometriche. Le DRAM sono disponibili anche con un numero di celle di memoria molto elevato, ma hanno lo svantaggio di richiedere un continuo *refresh*, cioè debbono, ad intervalli di qualche millisecondo, essere *ricaricate*.

Esiste infine un tipo di RAM statica che conserva l'informazione quando l'alimentazione venga interrotta. Questa è nota come NVRAM o NVSRAM, sigla che sta per Non Volatile RAM. Essa nasce dall'unione di una RAM statica con una EEPROM, integrate sul medesimo chip.

Un esempio di RAM statica disponibile commercialmente è la P4C148 della Performance Semiconductor Corporation. Questa è una RAM da 4096 bit ($1k \times 4$). La struttura del blocco funzionale è mostrata in figura 2.6, mentre la configurazione dei pin è mostrata in figura 2.7.

L'indirizzo (10 bit) viene impostato sulle linee $A_0 - A_9$, mentre la parola (a 4 bit) da scrivere/leggere, viene impostata sulle linee $I/O_4 - I/O_1$.

Se l'ingresso indicato con $\overline{CE}/\overline{CS}$ (Chip Enable/Chip Select) è *alto*, l'uscita è in uno stato di "alta impedenza", qualunque siano i segnali sulle linee d'ingresso/uscita e sulla linea $\overline{WE}$ (Write Enable). In questo stato il consumo da parte del chip è fortemente ridotto. Se si vuole scrivere dati in memoria occorre mettere al livello *basso* l'ingresso $\overline{CE}/\overline{CS}$ come pure quello di $\overline{WE}$ dopo avere impostato i dati sugli ingressi I/O e l'indirizzo sugli ingressi A. Per leggere i dati, sempre mettendo *basso* l'ingresso $\overline{CE}/\overline{CS}$, occorre mettere alto l'ingresso $\overline{WE}$ (cioè disabilitarlo). Questa RAM è molto rapida (tempi di accesso compresi tra 10 e 25 ns).

Tra le molte applicazioni di una RAM, val la pena di menzionare quella come *shift-register*. Occorrerà, come mostrato in figura 2.8 far uso di un contatore esterno per generare indirizzi successivi della RAM. Quando arriva un impulso di clock, il registro in ingresso (costituito da FF di tipo D) carica i 4 bit in Q. L'invertitore fa si che il contatore agisca solo quando il segnale di clock va *basso*. Il sistema effettua quindi successivi shift di 4 bit in parallelo.

FUNCTIONAL BLOCK DIAGRAM

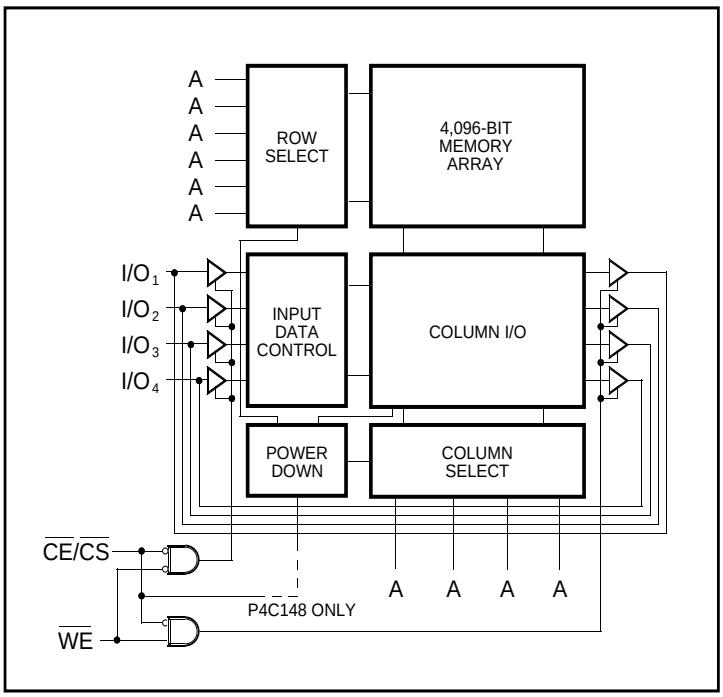

Figura 2.6:

PIN CONFIGURATIONS

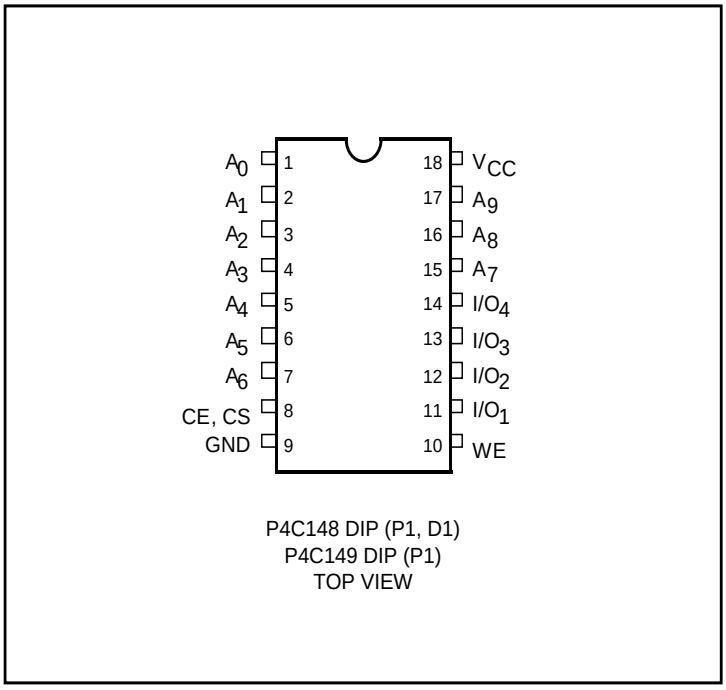

Figura 2.7:

2.3 Matrici logiche programmabili

In parallelo alle PROM ed alle RAM, hanno avuto un rapido sviluppo negli ultimi decenni sistemi costituiti da matrici di porte logiche i cui elementi possono essere,

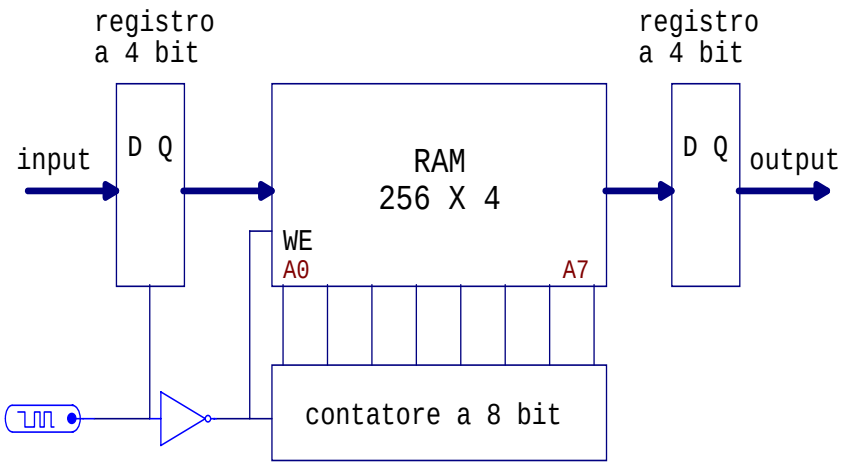

Figura 2.8:

facendo uso di appositi programmi, opportunamente collegati tra loro in modo da implementare funzioni logiche complesse. Appartengono a tale categoria le PAL (Programmable Array Logic) le PLA (Programmable Logic Array) le FPGA (Field Programmable Gate Arrays) etc. In questa sezione ci occuperemo delle PLA.

Queste hanno molte similarità con le PROM, per cui può esser utile iniziare la discussione partendo dalla struttura della PROM. Questa, come si è visto, può essere schematizzata come un decodificatore accoppiato ad un codificatore. Il decodificatore è costituito da una matrice di porte AND, il codificatore da una matrice di OR (che negli esempi esaminati era realizzata con diodi). In un PROM ancora non programmata, tutte le connessioni delle porte AND e tutte quelle delle porte OR sono presenti. Si consideri ad esempio il circuito mostrato in figura 2.9, che rappresenta una matrice 2×2.

Qui le crocette indicano i collegamenti esistenti. L'assenza di una crocetta starà a significare un collegamento interrotto. Il circuito di figura è una schematizzazione di quello reale. Ciascuna delle porte AND del circuito avrà in realtà quattro ingressi $(A, B, \overline{A}, \overline{B})$. L'eliminazione di uno o più dei collegamenti (operazione che nella EPROM può esser realizzata in laboratorio) implica l'eliminazione del relativo ingresso. Analoga osservazione va fatta per le porte OR, che sono, nel nostro esempio, a quattro ingressi, cioè ricevono le uscite delle porte AND 1,2,3,4 (almeno fino a che non siano eliminati uno o più dei relativi collegamenti).

Normalmente, in una PROM (o EPROM) il piano AND è fisso, cioè non tutti i collegamenti (le crocette nel grafico) sono presenti e quelli presenti non sono modificabili, mentre il piano OR contiene tutti i collegamenti. In una PLA sono presenti tutti i collegamenti del piano AND e tutti quelli del piano OR. In aggiunta, in una PLA sono presenti sul medesimo chip un certo numero di Flip-Flop di tipo D (registri) a cui sono collegate le uscite delle porte OR. Gli output di tali regsitri (nonchè i loro complementi) sono disponibili, insieme ai normali pin di ingresso, come inputs per la matrice logica. Ciò consente di realizzare, oltre a logiche *combinatorie* anche logiche *sequenziali*.

Ammettiamo di voler realizzare, a partire dalla PLA di figura, un circuito (com-

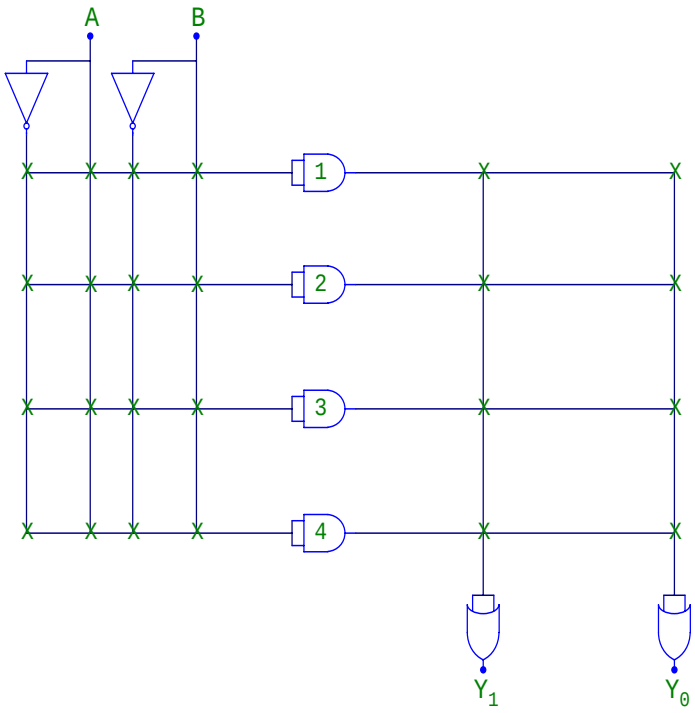

Figura 2.9:

binatorio) che fornisca:

$$Y_1 = A \oplus B$$

$$Y_0 = A + B$$

cioè l'OR esclusivo e l'OR. Per ottenere ciò è sufficiente, come à facile verificare, eliminare alcuni dei collegamenti, come mostrato nella figura 2.10.

Vediamo che con poche modifiche è possibile implementare funzioni logiche di tipi diversi, a partire da un circuito dato. Un ulteriore esempio, relativo ad una matrice 3×4 è mostrato in figura 2.11

In aggiunta alle PLA sono disponibili in commercio le cosiddette PAL (Programmable Array Logic) ([1]). Queste differiscono dalle PLA per il fatto di avere la matrice delle porte AND programmabili mentre è fissata la matrice delle porte OR. Sia le PLA che le PAL sono disponibili sia in forma combinatoria (cioè senza registri) che in forma sequenziale (con registri).

La differenza tra la struttura combinatoria e quella sequenziale è di notevole importanza. Un circuito combinatorio fornisce infatti un'uscita che è funzione esclusivamente dei segnali presenti in quell'istante sui pin d'ingresso. Un circuito sequenziale invece fornisce in uscita, all'arrivo di un segnale di clock, dei livelli che sono funzioni *sia dei segnali presenti sui terminali d'ingresso che dei valori presenti alle uscite subito prima dell'arrivo dell'impulso di clock.*

Esaminiamo, come esempio di circuito sequenziale, il contatore in base 3 mostrato in figura 2.12.

[1] Storicamente, le PLA hanno fatto la loro prima comparsa nel 1972, mentre le PAL sono divenute standard nel 1980. Le prime FPGA, che esamineremo dopo, sono state introdotte dalla Xilinx nel 1985

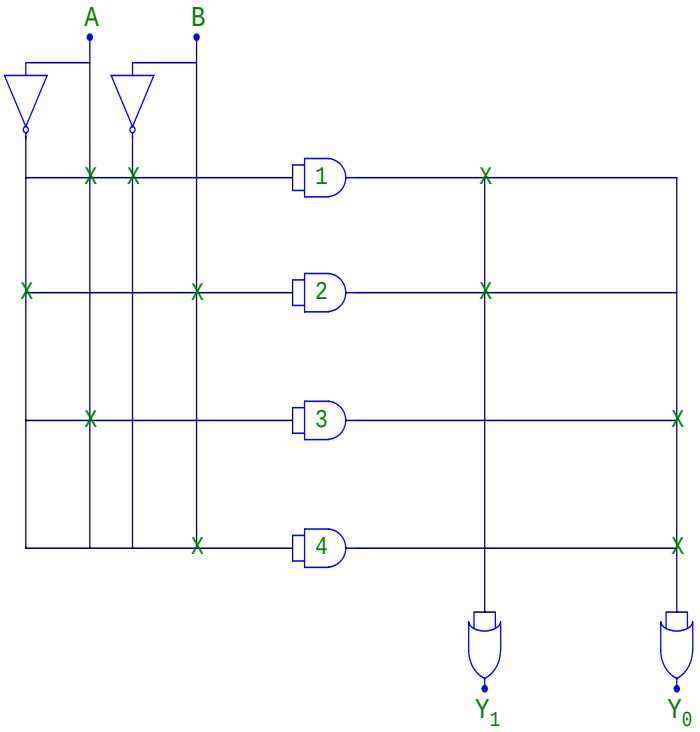

Figura 2.10:

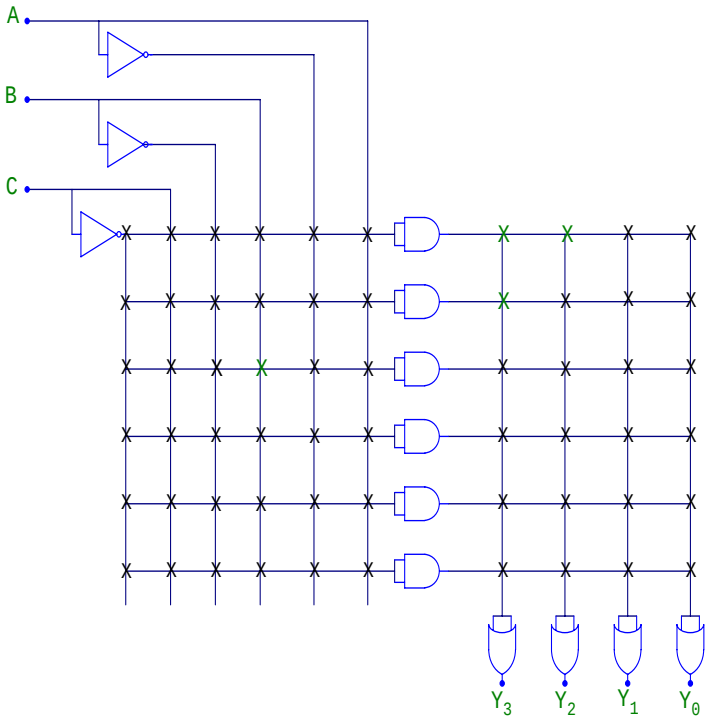

Figura 2.11:

Questo è un caso molto particolare di circuito sequenziale, poichè l'unico ingresso esterno è quello di clock. La porta NOR di figura riceve in ingresso l'uscita del

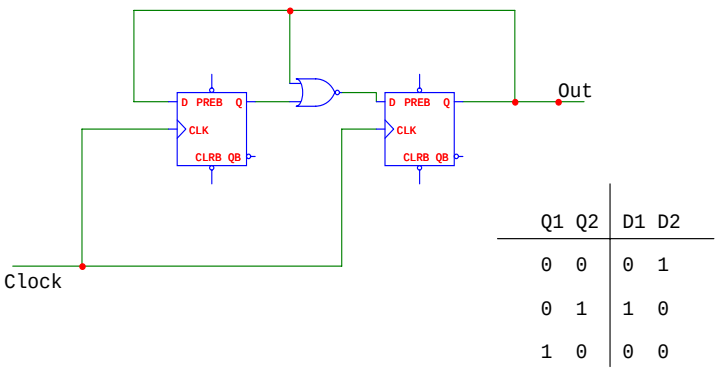

Figura 2.12:

secondo FF. Ammettiamo che lo stato inziale del sistema sia quello con Q_1=0, Q_2=0. Con ciò avremo D_2=1 prima dell'arrivo del primo impulso di clock, come mostrato in tabella 2.2.

clock	D_1	D_2	Q_1	Q_2
	0	1	0	0
↑	1	0	0	1
↑	0	0	1	0
↑	0	1	0	0

Tabella 2.2:

All'arrivo di questo, D_1 sarà trasferito in Q_1 (cioè Q_1=0) e D_2 in Q_2 (cioè Q_2=1). Avremo così la situazione mostrata nella seconda riga della tabella. Il successivo impulso di clock porta D_1 (che è 1) in Q_1, e D_2 (che è 0) in Q_2. La situazione sarà ora quella mostrata nella terza riga della tabella. Infine il terzo impulso di clock ripristina lo stato iniziale: Q_1=0, Q_2=0. Il conteggio ora riprende.

Lo schema di un sistema sequenziale, nel caso generale, è quello mostrato in figura 2.13.

Le linee solide indicate nella figura costituiscono i "bus" che forniscono da un lato in input, alla parte combinatoria del circuito, le uscite dei registri (Flip-Flop) e dall'altro caricano nei registri le uscite delle porte logiche (gli OR). Sono poi indicati separatamente i registri di input e di output, nonchè il clock.

In un circuito sequenziale il sistema passa attraverso una sequenza di stati o livelli dei suoi componenti. Se esso ha raggiunto un certo stato e se riceve da eventuali linee esterne opportuni impulsi, esso potrà passare ad uno di k possibili stati diversi, a seconda dello stato in cui esso è e dei valori presenti sulle linee di ingresso esterrne. Un sistema di questo tipo è descrivibile come una *macchina a stati finiti*. Esempi sono l'elettronica che controlla la distribuzione delle bibite in un distributore automatico, il sistema di controllo del ciclo in una lavabiancheria, di un semaforo etc.

Può accadere che un sistema di questo tipo venga a raggiungere uno *stato stabile*,

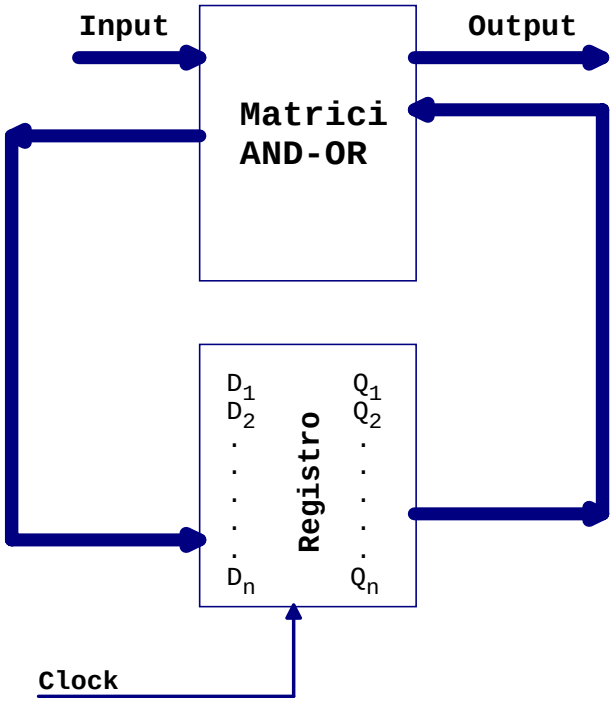

Figura 2.13:

cioè uno stato da cui esso non potrà più uscire. Tale situazione, se si vuole che la macchina funzioni come programmato, vanno opportunamente evitate. Nell'esempio fatto prima, del divisore per 3, ciò non accade. In effetti, anche se il sistema si trovasse inizialmente nello stato $Q_1 = Q_2 = 1$, non previsto, dopo il successivo impulso di clock esso passerebbe nello stato $Q_1 = 0$, $Q_2 = 0$, dopo di che ricomincerebbe con la sequenza già vista.

Esaminiamo ancora, come esempio dell'uso di FF in circuiti sequenziali, un *temporizzatore*.

Ammettiamo che un certo strumento generi, in occasione di un evento esterno (l'accensione di un allarme, il passaggio di una particella, un brusco salto di tensione su di una linea) un impulso, e che questo impulso debba esser *letto* da un sistema "sincrono", cioè da un sistema digitale in cui tutte le transizioni tra stati hanno luogo in sincronia con un segnale di clock fissato. L'evento esterno avviene invece ad un istante casuale e non è quindi sincrono con tale clock. Se vogliamo che l'impulso generato possa agire sul sistema, dobbiamo in qualche modo "sincronizzarlo". Cioè dobbiamo,a partire da esso, generare un altro impulso che sia sincrono con l'impulso di clock che arriva subito dopo l'evento. Ciò può esser ottenuto facendo uso del circuito di figura 2.14, che fa uso di un FF.

L'evento esterno aziona, per un breve intervallo di tempo, un deviatore, portandolo nella posizione A e riportandolo poi, dopo un tempo t_D (maggiore della durata dell'impulso di clock) nella posizione B. Il sistema costituito dalle due porte NAND accoppiate genererà un livello alto sull'ingresso D del FF, di durata t_D. Tale segnale non arriverà però in Q, e quindi all'uscita della porta AND, se non all'arrivo del successivo impulso di clock. Vediamo così che l'impulso in uscita dalla porta AND è sincrono con il clock. Il funzionamento del circuito è visibile in figura 2.15.

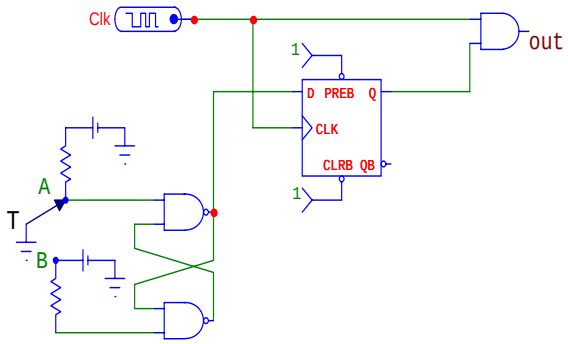

Figura 2.14:

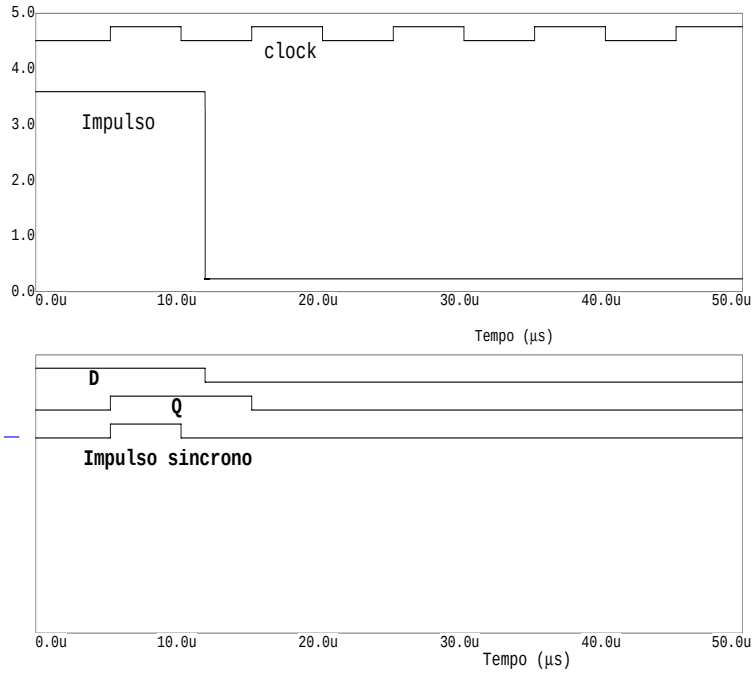

Figura 2.15:

É possibile realizzare un circuito sequenziale anche facendo uso di soli FF, cioè senza l'uso di porte logiche (AND, NAND etc.). Vediamo ad esempio un contatore sincrono in base 3, che fa uso di solo due FF JK, come mostrato in figura 2.16.

É facile verificare, come mostrato nella tabella 2.3 ed in figura 2.17, il funzionamento del circuito.

Vediamo infatti dalla tabella che dopo il terzo impulso (fronte di discesa) di clock, il sistema torna nello stato iniziale. Esaminiamo più in dettaglio il comportamento. Se lo stato iniziale è quello con $Q_1 = Q_2 = 0$, avremo $\overline{Q}_1 = 1$. Vediamo allora che è $J_2 = K_2 = 1$, mentre $J_1 = Q_2 = 0$. IL primo impulso di clock trova il secondo FF pronto ad eseguire un cambiamento di stato (toggle), mentre l'uscita del primo sarà ancora 0. Avremo così: $J_1 = Q_2 = 1$, $\overline{Q}_2 = 0$, $Q_1 = 0$, $\overline{Q}_1 = J_2 = K_2 = 1$. Il secondo impulso di clock troverà che ora il primo FF è in condizione di "toggle", come continua ad esserlo il secondo. La transizione causata dal clock porterà quindi

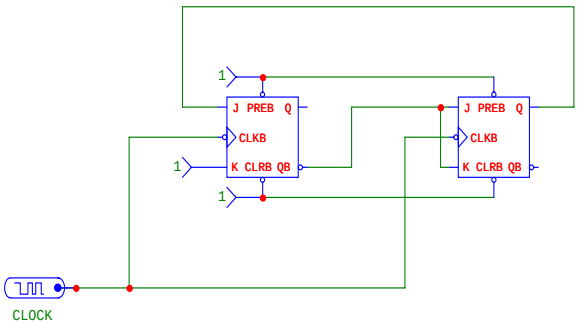

Figura 2.16:

ck	J_1	K_1	J_2	K_2	Q_1	$\overline{Q}_1$	Q_2	$\overline{Q}_2$
-	0	1	1	1	0	1	0	1
$\downarrow$	1	1	1	1	0	1	1	0
$\downarrow$	0	1	0	0	1	0	0	1
$\downarrow$	0	1	1	1	0	1	0	1

Tabella 2.3:

a: $Q_1 = 1$, $Q_2 = 0$ e quindi: $J_1 = Q_2 = 0$, $Q_1 = 1$, $\overline{Q}_1 = 0$. Ora il secondo FF non è più in condizione di 'toggle', nè lo è il primo. La transizione causata dal clock porterà allora a $Q_1 = 0$, $\overline{Q}_1 = 1$, mentre Q_2 e $\overline{Q}_2$ rimangono inalterati (essendo $J_2 = K_2 = 0$). Abbiamo così l'ultima riga della tabella, identica alla prima. Il ciclo ricomincia quindi dopo tre cicli di clock.

2.4 Gli integrati FPGA

Gli FPGA (Field Programmable Gate Arrays) sono stati introdotti nel 1985 dalla Xilinx Inc.. Essi hanno incontrato una larghissima diffusione grazie alla loro grande versatilità e relativa facilità d'uso. Essi, al pari delle PAL, PLA, EPROM, EEPROM, possono essere programmate in modo da rispondere a tutto un campo di applicazioni particolari. Esistono in commercio FPGA con strutture e caratteristiche diverse tra loro. Come il nome ci dice, si tratta di circuiti programmabili "sul campo", vale a dire che non occorrono *programmatori* appositi, come nel caso delle EPROM.

Questi integrati contengono milioni di transistor, che realizzano un elevato numero di circuiti logici i quali possono essere collegati tra loro facendo uso di commutatori programmabili. Gli FPGA sono una varietà di PLD (Programmable Logic Devices). Tuttavia, a differenza dei PLD, gli FPGA non contengono porte AND ed OR. Al loro posto essi contengono dei blocchi logici costituiti da "Lookup Tables", che vengono adoperati per implementare funzioni logiche. Tali blocchi logici sono accessibili tramite blocchi di input/output (I/O) che corrono lungo la periferia del chip e sono connessi tra loro per mezzo di blocchi di interconnessione, come è possibile vedere in figura 2.18. Una tipica FPGA contiene da circa 10,000 a milioni di elementi equivalenti a porte logiche, con interconnessioni programmabili dall'utente,

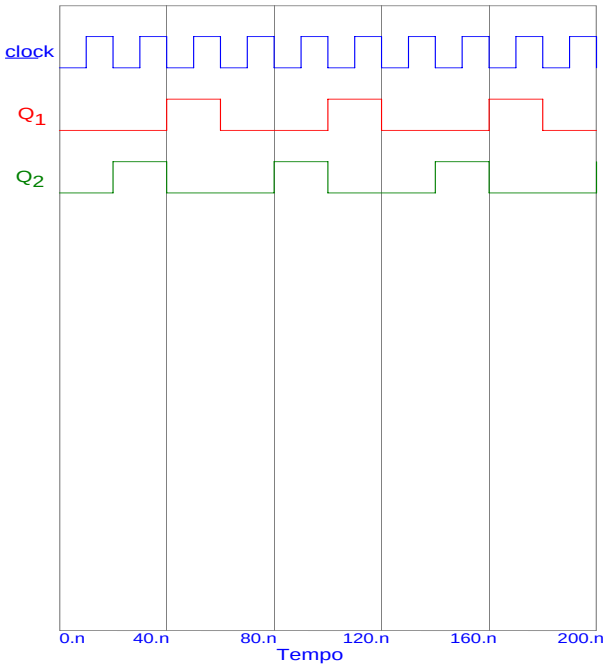

Figura 2.17:

oltre a centinaia di migliaia di celle di memoria. Una qualsiasi funzione logica, sia di tipo combinatorio che sequenziale, puo' essere implementata facendo uso delle intrinseche strutture logiche quali le Celle Logiche (LC) o Blocchi Logici (LB). Le LC (LB) sono come delle piccole ROM, in cui sono immagazzinate delle Lookup Tables (LUT) che contengono i valori delle funzioni logiche elementari, per ogni combinazione di valori impostati in ingresso.

La figura 2.19 mostra un po più in dettaglio la struttura di una tipica FPGA. Si possono vedere le celle logiche, le interconnessioni, le celle di I/O e la memoria. Si vede anche il dettaglio di una tipica cella logica, costituita in questo esempio da tre "generatori di funzioni logiche" (basati su lookup-tables) ed un FF di tipo D.

La varietà delle "architetture" disponibili in commercio è enorme. Esse differiscono sia nelle dimensioni che nella struttura delle LC e delle interconnessioni. Alcune LC si riducono all'equivalente di porte NAND a due ingressi, altre hanno una struttura più complessa e possono esser dotate di Multiplexers e di Look-Up Tables (LUT). In alcune FPGA un blocco logico corrisponde ad una intera struttura del tipo di una PAL. In ogni caso, una caratteristica comune alla più gran parte delle architetture è la presenza di FF di tipo D in ciascuna LC.

2.4.1 Gli FPGA della Xilinx

Descriveremo ora, a titolo di esempio, le caratteristiche di una FPGA prodotta dalla Xilinx.

Tale FPGA ha tre elementi configurabili principali: i Blocchi Logici Configurabili (CLB), i Blocchi di Input-Output (IOB) e le Risorse Programmabili di Interconnessione IPR. I CLB costituiscono gli elementi funzionali per costruire la logica voluta dall'utente. Gli IOB forniscono l'interfaccia tra i pins dell'integrato e le linee interne di segnale. Le IPR forniscono i percorsi di collegamento per connettere gli ingressi

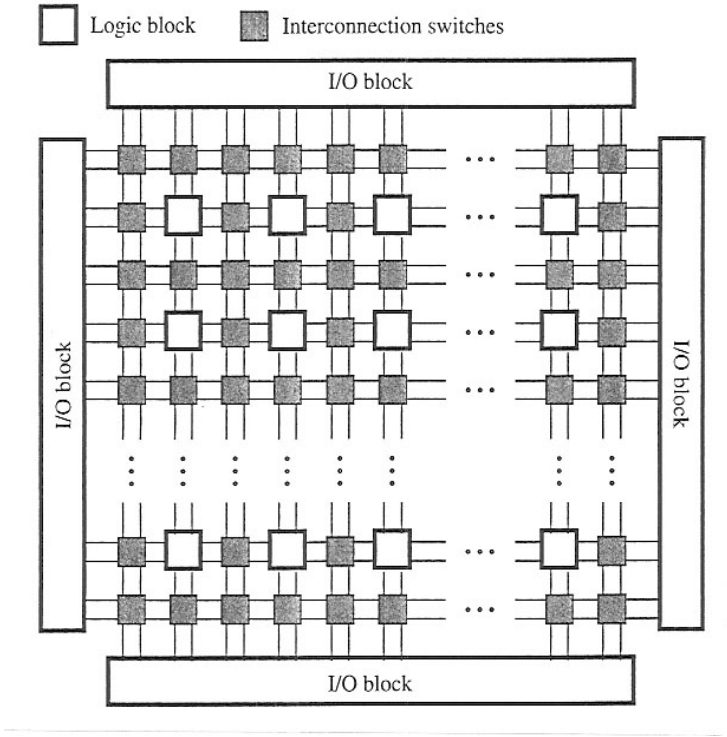

Figura 2.18:

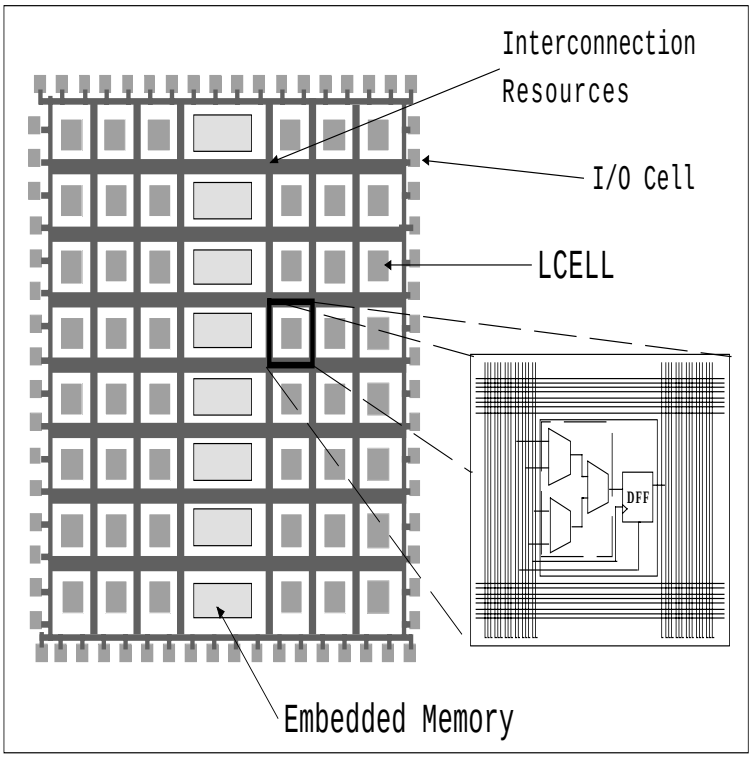

Figura 2.19:

e le uscite dei CLB e degli IOB alle reti appropriate. É possibile programmare le celle di memoria statica interna per determinare le funzioni logiche da svolgere come

pure i collegamenti esterni dell'FPGA.

La figura 2.20 mostra una parte dei blocchi logici e delle relative interconnessioni di un FPGA. Tutte le connessioni interne sono composte di segmenti metallici con punti di contatto programmabili, per implementare i collegamenti desiderati. Le IPR sono sovrabbondanti, in modo da consentire un'implementazione efficiente dei collegamenti. Vi sono quattro tipi diversi di interconnessioni, tre delle quali si distinguono solo per la lunghezza dei segmenti: linee di lunghezza singola, di lunghezza doppia e linee lunghe. In aggiunta esistono otto linee globali, adoperate per clocks o segnali di controllo globali.

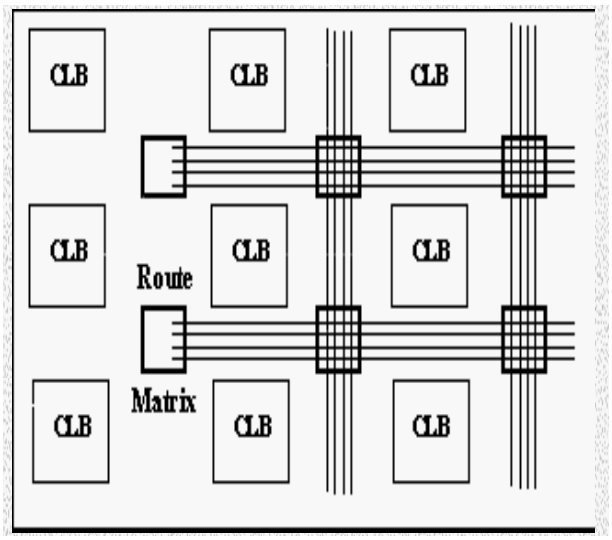

Figura 2.20:

Un esempio di CLB è quello mostrato in fig 2.21. Ciascun CLB contiene una coppia di FF e due generatori di funzioni logiche indipendenti (indicati con F e G nella figura), ciascuno dei quali ha quattro ingressi. Questi generatori di funzioni logiche hanno un elevato grado di flessibilità, grazie appunto alla presenza di quattro ingressi (la più gran parte delle porte logiche combinatorie hanno meno di quattro ingressi). Tuttavia, un terzo generatore di funzioni logiche (H) è aggiunto, come mostrato in figura. Il generatore H ha due ingressi. Uno o entrambi questi ingressi possono essere le uscite di F o G; l'altro (o entrambi) possono essere esterni al CLB. Ne segue che il CLB può implementare funzioni di un numero di variabili fino ad un massimo di 9.

I CLB implementano la parte principale della logica di questo FPGA. La flessibilità e la simmetria dell'architettura dei CLB facilitano il posizionamento ed i collegamenti in una qualsiasi applicazione.

La figura 2.22 mostra più in dettaglio la struttura interna del CLB. I due FF possono essere adoperati per immagazzinare i segnali in uscita dai generatori di funzioni logiche. Tuttavia i FF ed i generatori di funzioni logiche possono anche essere adoperati indipendentemente. L'ingresso indicato con DIN può essere adoperato come un input diretto per ciacuno dei due FF. Il segnale H1 può pilotare l'altro FF attraverso il generatore H. Le uscite dei generatori di funzioni logiche sono anche disponibili all'esterno del CLB, facendo uso di due outputs indipendenti da quelli dei FF. Tale versatilità aumenta la densità di componenti logici e semplifica i collegamenti.

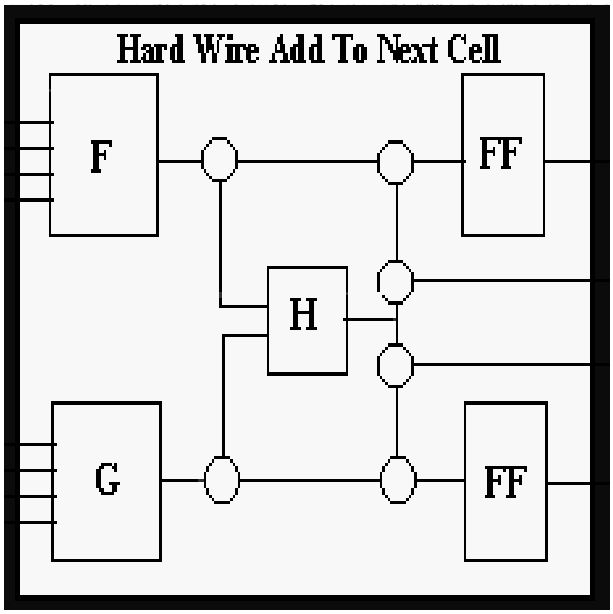

Figura 2.21:

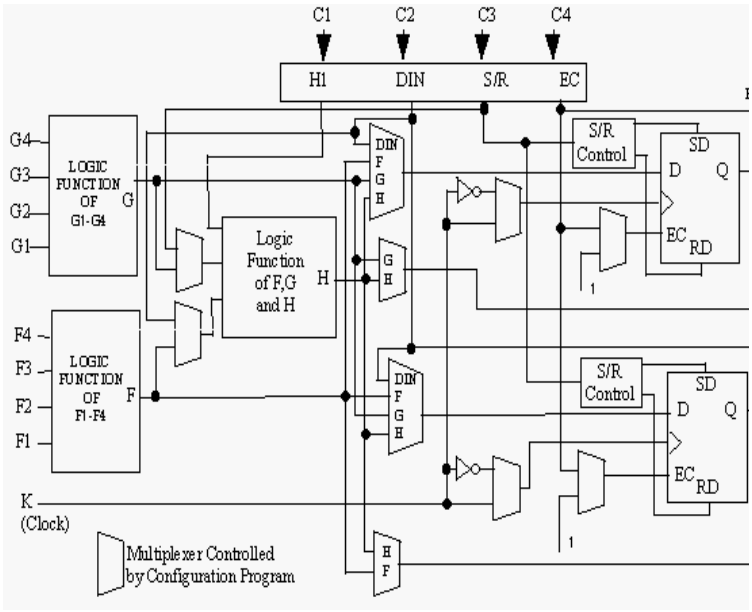

Figura 2.22:

2.4.2 L'FPGA della ALTERA

La ALTERA produce FPGA con un numero di celle logiche (chiamate Logic Elements dalla ALTERA) che va da circa 500 ad oltre 12000. Concentreremo la nostra attenzione sul modello denominato FLEX 10K. La sigla FLEX sta per Flexible Logic Element Matrix, ed indica una struttura in cui i LE sono a loro volta organizzati in blocchi di matrici logiche (Logic Array Blocks) o LAB (vedasi la figura 2.23). Ciascun LAB contiene otto LE. Gli elementi di un dato LAB sono collegati tra loro da una connessione locale (quindi veloce) detta Local Interconnect. Ciascun LE consiste di una LUT a quattro ingressi, un FF programmabile, oltre ad un certo numero

di linee dedicate. Gli otto LE di un LAB possono essere adoperati per creare blocchi logici di medie dimensioni (contatori a 8 bit, decodificatori d'indirizzo o macchine di stato). Più LAB possono esser uniti insieme per creare blocchi di complessità ancora maggiore. Ciascun LAB rappresenta l'equivalente di 96 porte logiche effettive.

Come si vede in figura 2.23, sono presenti sul chip anche un certo numero di Embedded Array Blocks (EAB) che sono costituiti essenzialmente da FF, Multiplexers e RAM. Essi possono essere adoperati per implementare ROM, RAM, Registri FIFO, funzioni logiche etc.. Nel caso della serie FLEX 10K, ciascun EAB può contribuire con l'equivalente di 100-600 porte logiche, per realizzare funzioni logiche complesse (moltiplicatori, microcontrollori, macchine di stato etc.). Ciascun EAB può esser adoperato indipendentemente, o diversi EAB possono esser combinati per implementare funzioni di maggior complessità.

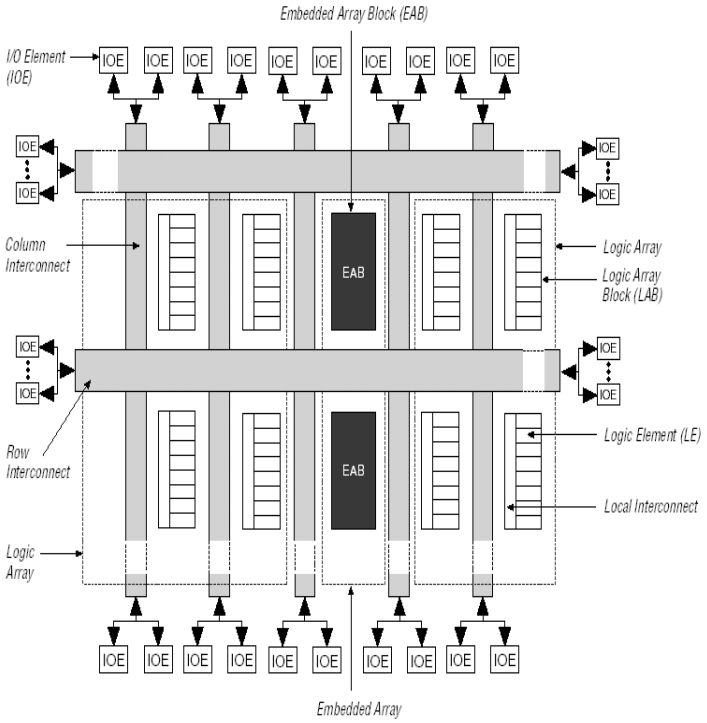

Figura 2.23:

La figura 2.24 mostra in dettaglio la struttura di un EAB.

Come accennato, questo è un blocco flessibile di RAM, dotato di registri sia alle porte d'ingresso che a quelle d'uscita. Esso può essere adoperato per implementare quelle funzioni logiche di maggiore complessità note come "megafunzioni". Ad esempio un EAB può essere adoperato per programmare moltiplicatori, filtri digitali, circuiti di correzione di errori etc.. Le funzioni logiche effettuate dal EAB vengono implementate costruendo, al momento della programmazione, costruendo delle LUT. Ciò rende molto più veloci i tempi di calcolo. Un EAB può anche essere adoperato come RAM, con una struttura di 256 parole di 8 bit, 512 da 4, 1024 da 2 o 2048 da 1. Inoltre più EAB possono esser combinati insieme per realizzare dei blocchi di RAM di dimensioni maggiori.

Un software dedicato aiuta l'utente a realizzare tutte le operazioni volute. Nei

pacchetti software forniti sono già implementate moltissime delle funzioni logiche, comprese quelle di maggiore complessità.

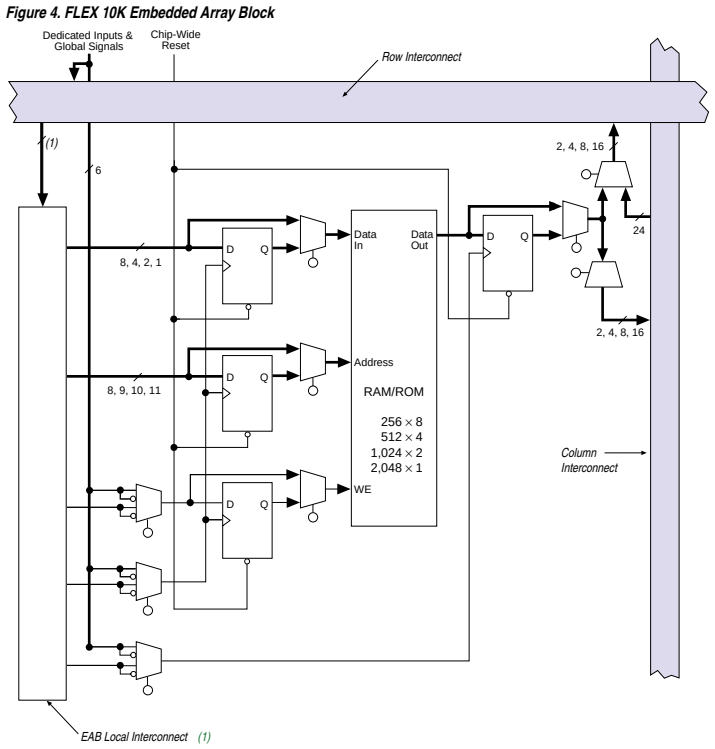

Figura 2.24:

La figura 2.25 mostra la struttura di un insieme di LE, cioè di un LAB.

Ciascun LAB consiste di otto LE, nonchè dei relativi ingressi e delle interconnessioni locali. Inoltre esso fornisce quattro segnali di controllo (con inversioni programmabili) che possono essere adoperati da tutti e otto gli LE del blocco. Due di questi ingressi possono essere adoperati come clocks; gli altri due come ingressi di controllo di clear/preset.

La figura 2.26 mostra il dettaglio di un LE. Questo costituisce la più piccola delle unità logiche nell'architettura di questo FPGA. Ciascun LE contiene una LUT a quattro ingressi, che costituisce un generatore di funzioni logiche attraverso il quale è possibile calcolare una generica funzione logica di quattro variabili. In aggiunta, esso contiene un FF programmabile, dotato di un Enable sincrono, e connessioni da/per altri elementi analoghi. Il FF può esser programmato come un JK, SR, D etc.. Gli ingressi di clock, clear e preset sul FF possono esser pilotati da segnali esterni globali, da segnali di I/O, o da altra logica interna.

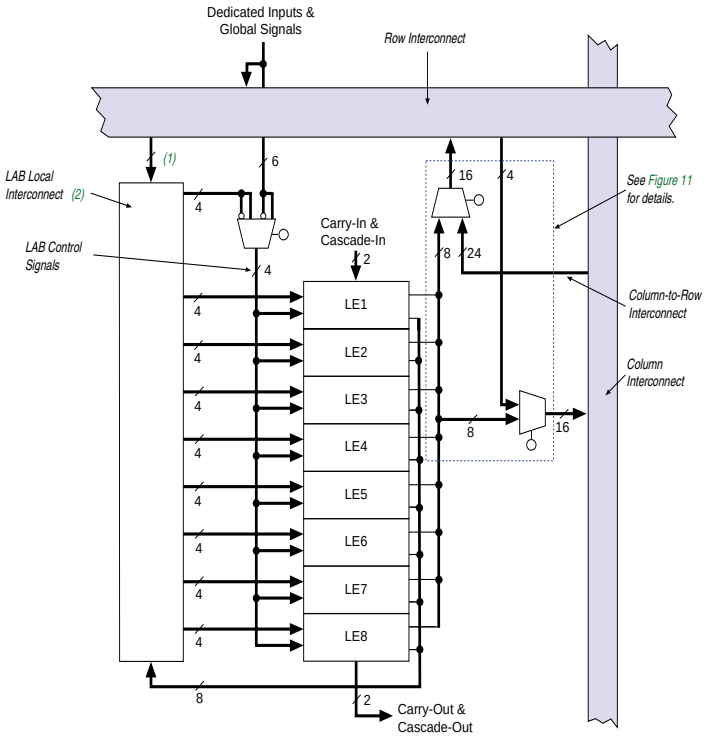

Figura 2.25:

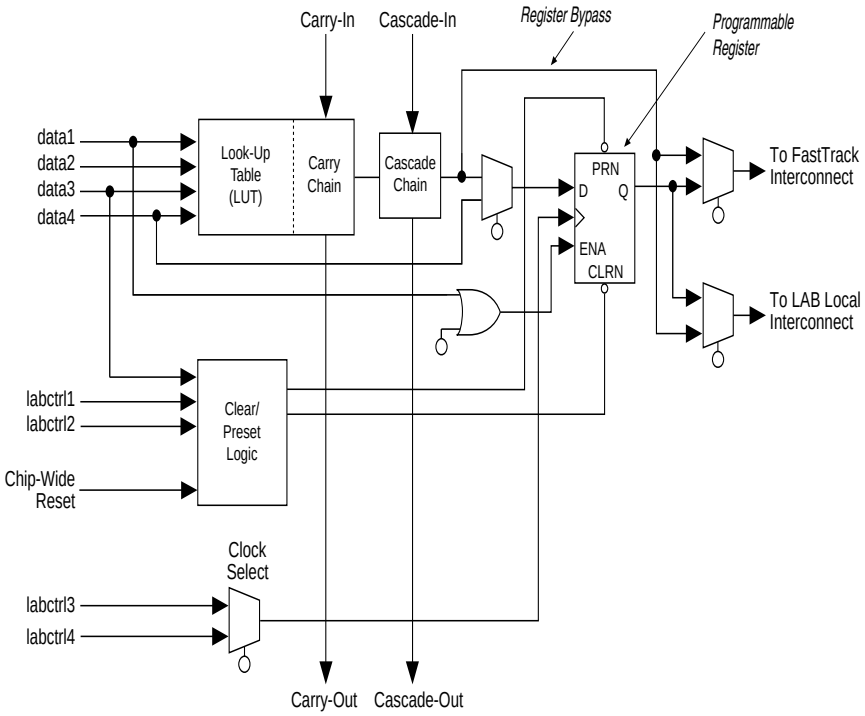

Figura 2.26: