Appunti di reti logiche

Indice

Premessa le spiegazioni del Fornaciari (il professore), sono abbastanza incomplete e incomprensibili. Per una preparazione robusta ti conviene consultare il libro di testo ("Reti logiche" di Bolchini, Brandolese, Salice e Sciuto; Seconda edizione, Apogeo). Questi appunti sono solo una integrazione di tra quello che viene detto a lezione, a esercitazione e una breve revisione di questi ultimi.

Aritmetica di Boole

Lavorare con l'aritmetica binaria (booleana) ci permette di semplificare molto il nostro lavoro tramite una logica semplice basata su 2 livelli che modella a pennello la realizzazione su silicio. Inoltre la sua semplicità permette anche la realizzazione di strumenti per l'analisi automatica.

Iniziamo a dare le definizioni principali che utilizzeremo per costruire l'algebra booleana.

Definizione. ***Operazione*** Una operazione α sull'insieme $s = \{s_1, s_2, \dots\}$ è una funzione $S : S \times S \to S$.

Definizione. ***Sistema algebrico*** Combinazione di una o più operazioni.
Definizione. ***Algebra Booleana*** È un sistema algebrico identificato dalla quintupla $(B, +, *, O, I)$ dove:

1. B è l'insieme di specificazione (carrier)
2. $+, *$ sono operatori binari che corrispondo rispettivamente a somma e prodotto
3. O, I sono elementi speciali di B

Le proprietà delle due operazioni sono dedotte da assiomi:

- $+$ e $*$ sono commutative per ogni $x, y \in B$
- La somma è distributiva rispetto al prodotto ed il prodotto è distributivo rispetto alla somma per ogni $x, y, z \in B$
- O è l'elemento neutro rispetto a $+$ e I quello rispetto a $*$
- Ogni elemento $x \in B$ ammette un elemento x' complemento di x tale che $(x + x') = I$ e $(x + x') = O$. Questo elemento è unico, perciò possiamo definire $'$ l'operatore unario di complemento.

L'ordine delle operazioni viste è il consueto. Inoltre godono delle già viste proprietà (vedasi ACSO e LEA):

1. Associativa
2. Distributiva
3. Idempotenza
4. Elemento neutro
5. Assorbimento
6. Involuzione
7. De Morgan
8. Consenso

$$\text{Somma: } a * b + a' * c + b * c = a * b + a' * c$$
$$\text{Prodotto: } (a + b) * (a' + c) * (b + c) = (a + b) * (a' + c)$$

Definizione. ***Principio di dualità*** Ogni identità deducibile dai postulati dell'algebra di Boole è trasformata in un'altra identità se ogni somma è sostituita da un prodotto (e viceversa) e ogni elemento identità O è sostituito da un elemento identità I.
Definizione. ***Espressione Booleana*** Una espressione booleana E è definita in modo induttivo come parola composta da operatori booleani, parentesi costanti e letterali nel modo seguente:

1. Sia gli elementi di B, chiamati costanti, che i letterali x, y, z sono espressioni;
2. Se E_1 e E_2 sono espressioni booleane anche $(E_1 + E_2)$, $(E_1 * E_2)$, (E_1') lo sono;
3. Non esistono altre espressioni booleane oltre a quelle che possono essere generate da un numero finito di applicazione delle due regole precedenti.

Possiamo definire algebra di commutazione una particolare algebra booleana della seguente forma: $(\{0, 1\}, +, *, 0, 1)$.

Funzioni booleane ed espressioni booleane

Definizione ***Funzione di commutazione*** Una funzione di commutazione a n variabili è una funzione: $f : \{0, 1\}^n \to \{0, 1\}$

Le funzioni di commutazioni hanno una comoda rappresentazioni in forma tabellare (tabella di verità). Una funzione booleana di n variabili può essere espressa da una espressione booleana con n variabili.

Le proprietà dell'algebra possono essere utilizzate per manipolare un'espressione booleana ed ottenerne una equivalente. Due espressione sono equivalenti se sono riconducibili alla stessa funzione booleana.

Data una funzione booleana spesso il problema è proprio definire un'espressione booleana soddisfacente. Inoltre, poiché esistono diverse rappresentazioni equivalenti bisogna anche definire una metrica che stabilisca la qualità della rappresentazione. Si osservi che la metrica è caratterizzata da un valore atteso e da una varianza, quindi due soluzioni a costo simile (e.g. $\pm 20\%$) sono da considerarsi equivalenti.

Data una funzione booleana la soluzione iniziale al problema di determinare una corrispondente espressione booleana è usare le forme canoniche. Le due forme canoniche sono `sum of products` e `products of sum` (vedasi ACSO e LEA). Entrambe le forme canoniche sono un'applicazione del teorema di espansione di Shannon:

Teorema. ***Teorema di espansione di Shannon*** Sia f una funzione booleana. Per ogni $(x_1, x_2, \dots, x_n) \in B^n$ si ha:

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= x_1' * f(0, x_2, \dots) + x_1 * f(1, x_2, \dots) = \\ &= x_2' * f(x_1, 0, \dots) + x_2 * f(x_1, 1, \dots) = \dots \end{aligned}$$

Dualmente si ha:

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= (x_1' + f(1, x_2, \dots)) * (x_1 + f(0, x_2, \dots)) = \\ &= (x_2' + f(x_1, 1, \dots)) * (x_2 + f(x_1, 0, \dots)) = \dots \end{aligned}$$

Non è possibile, però, identificare un algoritmo che trasforma espressioni algebriche in modo da minimizzare i criteri di costo. Quindi non si saprà mai se si è raggiunta la forma minima!

Sintesi di reti combinatorie a due livelli

Il nostro obiettivo sarà quello di ridurre la complessità di funzioni booleane espresse in forma PoS o SoP. Noi ci riferiremo solo alla SoP.

Le nostre ottimizzazioni punteranno a ridurre principalmente il numero di termini prodotto. In seconda istanza, ridurremo anche il numero di letterali.

Le metodologie di sintesi ottima sono tre:

1. Il metodo delle mappe di Karnaugh
2. Il metodo di Quine-McCluskey
3. Le euristiche per sintesi a due livelli

Diamo inizialmente delle definizioni che useremo durante questo capitolo.

Definizione. ***ONset, OFFset e DCset*** Sia f una generica funzione in n variabili. Si definiscono ONset, DCset e OFFset i seguenti insiemi:

$$ONset = \{X_i | f(X_i) = 1\} \quad DCset = \{X_i | f(X_i) = -\} \quad OFFset = \{X_i | f(X_i) = 0\}$$

Una notazione usata per indicare i mintermini di f è la seguente:

$$f(X) = ONset(\{configurazioni...\})$$

Per indicare i mintermini di una funzione $f(a, b, c)$ che vale 1 per $\{0, 0, 1\}$, $\{0, 1, 0\}$ e $\{1, 0, 0\}$ possiamo scrivere $f(a, b, c) = ONset(1, 2, 3)$: i numeri indicano la trasposizione in numeri binari delle configurazioni ($001 = 1$, $010 = 2$, $100 = 3$).

Metodo delle mappe di Karnaugh

Un primo metodo per l'identificazione di forme minime a due livelli è applicare la seguente regola di riduzione:

$$aZ + a'Z = Z$$

In cui Z è un termine prodotto di $n - 1$ variabili. Questo metodo può essere applicato ad un numero di termini pari a 2^n e mantiene inalterato il numero di livelli. In più le somme di prodotti rimangono tali, al più tali espressioni possono anche banalizzarsi diventando o semplici prodotti o costanti. Il problema consiste nell'identificare tutti i termini su cui applicare la riduzione e tutti i termini che partecipano a più riduzioni contemporaneamente e replicarli.

Il metodo delle mappe di Karnaugh consente di risolvere direttamente i problemi del metodo precedente. Esso è un metodo grafico la cui applicazione è semplice per funzioni con massimo fino a 4 variabili e diventa complesso per 5 o 6 variabili e addirittura inattuabile per più di 6. Una mappa di Karnaugh è uno schema deducibile dalla rappresentazione geometrica delle configurazioni binarie. Introduciamo la distanza di Hamming, necessaria all'applicazione del metodo:

Distanza di Hamming: In una rappresentazione binaria, la distanza di Hamming tra due stati è il numero di bit che variano tra i due stati.

Possiamo vedere la regola di riduzione come l'identificazione di configurazioni binarie associate ai termini prodotto che sono distanti una unità secondo Hamming. A tali configurazioni corrispondono coppie di mintermini in cui una sola variabile è naturale in un mintermine e complementata nell'altro:

$abcd' + ab'cd'$ può essere vista come due configurazioni:

- $abcd' \equiv 1110$
- $ab'cd' \equiv 1010$

I mintermini 1110 e 1010 sono ad una distanza di Hamming pari ad 1.

Utilizzando questa impostazione, una funzione binaria a n variabili può essere rappresentata mediante una rappresentazione geometrica cartesiana in uno spazio a n dimensioni in cui gli assi sono le variabili della funzione. In questa rappresentazione possiamo definire gli n-cubi:

Definizione. ***N-Cubi*** Data la rappresentazione cartesiana di una funzione binaria a n variabili, si dice n-cubo la figura ottenuta collegando i vertici le cui configurazioni sono a distanza di Hamming unitaria.

Si può facilmente trasformare una tabella di verità e n variabili in un n-cubo: basta segnare opportunamente le configurazioni per cui la funzione assume valore `1` o `0`.

La rappresentazione in uno spazio n-dimensionale non è maneggevole. Conviene perciò passare allo sviluppo nel piano degli n-cubi. Al cubo sviluppato nel piano con 2^n vertici si sovrappone una mappa con 2^n caselle organizzate secondo righe e colonne.

Esempio per un n-cubo bidimensionale:

b a	0	1
0	0	1
1	1	0

Esempio per un n-cubo tridimensionale

c ab	00	01	11	10
0	1	0	1	0
1	0	1	0	1

Si noti la disposizione delle variabili: si deve mantenere la distanza di Hamming pari a 1 tra colonne adiacenti.

La mappa così realizzata è detta mappa di Karnaugh. Le configurazioni assunte dalle variabili di ingresso danno origine agli indici di riga e colonna della mappa. In ogni casella di trascrive il valore assunto dalla funzione quando la configurazione delle variabili corrisponde a quella della coordinate che contrassegnano le caselle. Due caselle che condividono un lato di un n-cubo corrispondo a due configurazioni di variabili adiacenti.

Definizione. ***Implicante*** Un implicante è una funzione p associata ad un termine prodotto di m letterali con $1 \leq m \leq n$ tale per cui $f \geq p$. Ciò significa che per ogni `1` in p ne corrisponde uno in f. Un mintermine è un implicante per cui $m = n$.

Sostanzialmente gli implicanti sono dei raggruppamenti di `1` di variabili adiacenti. Affinché esso sia una riduzione, esso deve avere dimensione pari (2, 4 o 8). Esistono 2 tipi di implicanti:

Definizione. ***Implicanti primi*** È un implicante associato ad un termine prodotto a cui corrisponde un raggruppamento di

dimensione massima.

Definizione. ***Implicanti primi essenziali*** Un implicante primo che copre uno o più `1` non coperti da nessun altro implicante.

Definizione. ***Copertura*** Viene detta copertura la scelta del minor numero di implicanti primi ed essenziali.

Il metodo consiste nel trovare una copertura con il minor numero di implicati primi. Per fare ciò, partiamo dall'individuare tutti gli implicanti primi essenziali. Il seguente teorema ci può aiutare:

Teorema. Se una forma minima è composta da solo implicanti primi essenziali essa è unica.

In seguito eliminiamo tutti gli implicanti primi coperti da quelli essenziali e selezioniamo il numero minore di implicanti primi tra gli uni rimasti.

Ad ogni implicante selezionato è associato un termine prodotto. Esso è ottenuto identificando le variabili che non cambiano mai di valore e riportando ogni variabile in modo normale se il valore che essa assume è `1` e complementata se è `0`. La riduzione sarà la somma dei termini prodotto associati ai vari implicanti.

Condizioni di indifferenza

La specifica di un progetto spesso contiene delle condizioni di indifferenza (*don't care*). Le condizioni di indifferenza corrispondono a configurazioni di ingresso per le quali il valore dell'uscita non è noto e non è neppure di interesse sapere quanto può valere. Sulla tabella delle verità, o su una mappa di Karnaugh, il valore specificato della funzione si indica coi simboli `-` o `x`.

Le condizioni di indifferenza sono dei gradi di libertà nel processo di sintesi. A esse si può assegnare un valore a seconda di quanto conviene per minimizzare la funzione. Inoltre essa non deve essere per forza coperta da un implicante. È importante sottolineare che gli implicanti primi realizzati con sole condizioni di indifferenza non hanno nessuno scopo. Inoltre un implicante primo non diventa essenziale quando è l'unico a coprire una data condizione di indifferenza.

Metodo di Quine-McCluskey

I nostri obiettivi sono analoghi a quelli della sintesi tramite mappe di Karnaugh. Il metodo di Quine-McCluskey è un metodo di minimizzazione tabellare facile da tradurre in un algoritmo. Il numero di variabili per cui è applicabile è teoreticamente illimitato. Il procedimento è simile a quello delle mappe di Karnaugh: prima si cercano gli implicanti primi e dopo si cerca la copertura ottima.

Partiamo dai mintermini di funzione. Confrontiamo tutti i termini e semplifichiamo con la regola di riduzione tutte le coppie che hanno una parte comune ed una sola variabile differente. I termini semplificati sono marcati e non sono primi poiché hanno partecipato alla creazione di un implicante con meno letterali. Successivamente si crea con le riduzioni un nuovo insieme di termini prodotto da confrontare e si riparte dal confronto. Il processo termina se non ci sono più possibili riduzioni. I termini prodotto non marcati sono implicanti primi.

Il numero di confronti può essere ottimizzato: non vale la pena confrontare i termini che hanno sicuramente diversi per più di un letterale. Costruiamo allora dei gruppi di mintermini/implicanti contenenti lo stesso numero di `1`. Si comparano tra loro le configurazioni che appartengono a gruppi che differiscono per un solo `1`.

Per cercare la copertura si usa la cosiddetta tabella degli implicanti o tabella di copertura. Essa è una matrice binaria in cui le righe sono gli implicanti primi identificati e gli indici colonna sono i mintermini della funzione. Gli elementi della matrice sono pari a `1` quando l'implicante i-esimo copre il mintermine j-esimo, altrimenti `0`. Ad ogni riga associamo un costo, che di solito considereremo pari al numero di letterali usati nell'implicante. Definiamo l'insieme di copertura $\mathscr{C}(f) = \varnothing$. L'analisi della tabella può essere ricondotta al seguente algoritmo:

1. **Ricerca degli implicanti primi essenziali:** se una colonna contiene un solo `1`, la riga corrisponde è quella di un implicante primo essenziale. La riga dell'implicante e le colonne da essa coperte vanno eliminate dalla tabella. L'implicante viene aggiunto all'insieme di copertura.
2. **Dominanza di riga e colonna:** se vengono esauriti gli implicanti primi essenziali, possiamo applicare i metodi di dominanza di riga e colonna per tirarne fuori altri. Questi due metodi semplificano la tabella senza modificare l'insieme di copertura:

1. **Criterio di dominanza di riga:** la riga i domina la riga j se l'implicante P_i copre tutti i mintermini che copre l'implicante P_j più almeno uno ed il costo di P_i è minore o uguale al costo di P_j. La riga dominata può essere rimossa dalla tabella.
2. **Criterio di dominanza di colonna:** la colonna i domina la colonna j se il mintermine m_j è coperto dagli stessi implicanti da cui è coperto m_i più almeno uno. La colonna dominata può essere rimossa dalla tabella.

Se un implicante viene scelto come essenziale dopo la prima iterazione, ossia dopo che è stata applicata almeno una dominanza, esso viene detto implicante primo essenziale secondario, altrimenti viene detto primario. Alla fine dell'algoritmo si dovrebbe ottenere una tabella contenente un solo implicante (non è sempre vero come vedremo dopo) che verrà aggiunto all'insieme di copertura.

Funzioni non completamente specificate

L'estensione alle funzioni non completamente specificate richiede l'aggiunta di qualche regola:

1. **Ricerca degli implicanti primi:** Nel passo relativo alla generazione degli implicanti primi, le condizioni di indifferenza sono trattate come `1`;
2. **Ricerca della copertura ottima:** Nella tabella di copertura compaiono, come indici di colonna, solo i mintermini appartenenti all'ONset poiché sono essi a vincolare la funzionalità. Il DCset è l'insieme dei termini che rappresenta i gradi di libertà, perciò non è necessario sceglierli ma può essere conveniente.

Funzioni a più uscite

Nel caso di funzioni a più uscite, una prima soluzione consiste nel minimizzare le funzioni singolarmente. Il risultato ottenuto dall'unione, però, non potrebbe essere ottimo poiché le funzioni potrebbero condividere alcuni implicanti. Questi implicanti condivisi possono non essere unici per le singole funzioni. In generale è necessario considerare sia gli implicanti primi delle singole funzioni che quelli ottenuti combinando in tutti i modi possibili le funzioni da minimizzare.

L'estensione a funzioni a più uscite del metodo richiede delle modifiche ai vari passi:

1. **Costruzione della tabella:** Si procede come nel caso scalare con la differenza che si associa ad ogni mintermine un identificatore costituito da tanti bit quante sono le funzioni considerate. Questo bit assumerà il valore `1` se e solo se la funzione che ad esso corrisponde contiene tale mintermine e `0` altrimenti.
2. **Generazione di implicanti primi:** Si procede secondo le stesse modalità del caso scalare. L'identificatore del nuovo implicante è ottenuto come AND dei due identificatori. Se l'identificatore ottenuto è `00...0`, allora l'espansione non è valida. Marcheremo l'indicatore (anche entrambi) che coincide con il risultato dell'AND.
3. **Tabella di copertura:** È ottenuta dalla giustapposizione delle tabelle relative ad ogni funzione in cui si riportano solo i mintermini.
4. **Identificazione della copertura ottima:** avremo un insieme di copertura per ogni funzione Si applicano i criteri di scelta come nel caso scalare:
    1. **Essenzialità:** Se un implicante è essenziale per tutte le funzioni la riga è eliminata insieme a tutte le colonne coperte. Se invece l'implicante non è essenziale per tutte le funzioni la riga è mantenuta ed è scelto solo per le funzioni in cui è essenziale con costo pari a 1 (solo nel caso di costo come numero di letterali, nel caso di costo come numero di implicanti viene posto a 0). In queste ultime vengono eliminate le sole colonne coperte. L'implicante viene ovviamente aggiunto all'insieme di copertura della funzione corrispondente.
    2. **Dominanza di riga:** Come nel caso scalare.
    3. **Dominanza di colonna:** Come nel caso scalare; ha validità solo all'interno della funzione.

Alee

A causa dei ritardi differenti nell'attraversamento di porte logiche da parte dei segnali, si possono verificare malfunzionamenti transitori, costituiti da variazioni temporanee non corrette dal punto di vista logico, dei valori delle uscite al variare degli ingressi. La complessità del problema non è affrontabile se si vuole considerare ogni possibile variazione degli ingressi; diventa invece gestibile se ci si limita a considerare reti logiche funzionanti in modo fondamentale, ossia nelle quali un solo ingresso alla volta può variare il proprio valore e la rete deve avere tempo di assestarsi prima che avvenga una successiva variazione. Distinguiamo due tipi di malfunzionamento:

1. Alee statiche: dati due ingressi i_1 e i_2 passati in sequenza con uscite uguali $f = f(i_1) = f(i_2)$, l'uscita assume per un breve istante di tempo nel passaggio tra i due ingressi il valore f';

2. Alee dinamiche.

Eliminazione di alee statiche

L'eliminazione di alee statiche non si risolve inserendo ritardi nei circuiti, ma modificando il circuito. Il motivo dell'esistenza delle alee statiche è la presenza di più percorsi che collegano lo stesso ingresso attraverso porte `AND` all'uscita nei quali l'ingresso assumi valori diversi.

Le alee vengono risolte aggiungendo degli implicanti ridondanti. Infatti osservando la mappa di Karnaugh di una funzione che soffre di alee statiche, si può osservare la presenza di `1` adiacenti non coperti dallo stesso implicante. Ovviamente la risoluzione di alee può comportare la non minimalità della sintesi.

Circuiti combinatori speciali

Nella realizzazione di sistemi digitali complessi si ricorre spesso all'uso di un numero limitato di elementi di base con funzionalità be note. Vediamo alcuni di questi componenti.

Reti combinatorie di base

Multiplexer

Il multiplexer è una rete combinatoria in cui un segnale di selezione s permette di portare sull'uscita y uno dei segnali d'ingresso x_i. Nel caso più semplice i segnali di ingresso sono due, x_0 e x_1 e il segnale s è un singolo bit. La tabella di verità nel caso a due variabili è la seguente:

s	x_0	x_1	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

E porta alla seguente sintesi:

$$y = s' x_0 + s x_1$$

Si può facilmente generalizzare ad un numero maggiore di ingressi: utilizzando n segnali di selezione è possibile selezionare fino a 2^n ingressi. Una applicazione del multiplexer è legata alla sintesi di funzioni di molte variabili mediante la loro scomposizione secondo il teorema di Shannon. Consideriamo la funzione:

$$f(x_0, \dots, x_n) = x_0' x_1' f(0,0,\dots) + x_0' x_1 f(0,1,\dots) x_0 x_1' f(1,0,\dots) + x_0 x_1 f(1,1,\dots)$$

Si può vedere che questo tipo di scrittura ricalca quella del multiplexer usando x_0 e x_1 come ingressi di selezione.

Demultiplexer

Il demultiplexer è l'elemento che svolge la funzione duale rispetto al multiplexer. Nella sua versione più semplice, ha un ingresso dati x, un ingresso di selezione s e due uscite y_0 e y_1. Il demultiplexer associa il valore x alle uscite in base al valore del segnale di selezione. Come il suo duale, si può generalizzare ad un numero arbitrario di uscite.

Decoder

Un decoder è un elemento combinatorio dotato di n ingressi $x_0, \dots, x_n$ e di 2^n uscite $y_0, \dots, y_{2^n-1}$. Tali uscite sono tutte uguali a zero ad eccezione di quella avente come indice il valore decimale corrispondente al vettore di bit in ingresso.

Un decoder a 2 ingressi ha la seguente tabella di verità:

x_0	x_1	y_0	y_1	y_2	y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

La sintesi fornisce la seguente funzione multivariabile:

$$y_0 = x_0'x_1' \quad y_1 = x_0'x_1 \quad y_2 = x_0x_1' \quad y_3 = x_0x_1$$

Si può quindi notare che le quattro funzioni trovate sono tutti i mintermini possibili di una funzione di due variabili $f(x_0, x_1)$. La proprietà di sintesi dei mintermini rende il decoder particolarmente adatto per circuiti che sintetizzano funzioni in prima forma canonica.

Priority encoder

Un priority encoder è un elemento combinatorio dotato di 2^n ingressi $x_{2^n-1}, \dots, x_0$ e di $n+1$ uscite $y_0, \dots, y_{n-1}$ e z. L'uscita z assume il valore `1` se e solo se tutti gli altri ingressi valgono `0`. Quando almeno uno degli ingressi ha valore `1`, allora le uscite y_i sono la codifica binaria della posizione dell'uno più significativo nel vettore di ingresso e z vale 0.

x_3	x_2	x_1	x_0	y_0	y_1	z
0	0	0	0	-	-	1
0	0	0	1	0	0	0
0	0	1	-	0	1	0
0	1	-	-	1	0	0
1	-	-	-	1	1	0

La sintesi del componente restituisce le seguenti funzioni:

$$z = x_0'x_1'x_2'x_3' \quad y_0 = x_2 + x_3 \quad y_1 = x_1x_2' + x_3$$

Dispositivi programmabili

Sono dispositivi hardware che mettono a disposizione componenti logici, come porte logiche, flip-flop e buffer, e linee di connessione. Si dicono programmabili in quanto i componenti disponibili possono essere connessi a seconda delle esigenze del progetto. Esistono molte tipologie di dispositivi programmabili tra cui:

1. PAL, PLA, ROM, GAL
2. CPDL
3. FPGA

Possiamo classificarli in base alla programmabilità e al tipo di connessioni:

1. One-Time Programmable (OTP)
2. Riprogrammabili
3. Riconfigurabili
4. A connessioni globali

5. A connessioni locali e distribuite

OTP

Fuse e antifuse

Le linee del dispositivo sono prodotte in modo da essere sempre connesse. La programmazione consiste nel "bruciare" alcune connessioni in modo da mantenere solo quelle necessarie. La programmazione avviene mediante una tensione più elevata di quella normale di funzionamento.

L'antifuse è il complementare: le linee del dispositivo sono prodotte in modo da essere sempre disconnesse.

Riprogrammabili

E2PROM

Le linee del dispositivo sono prodotte in modo da essere sempre disconnesse. La programmazione consiste nel depositare carica sul floating gate del transistor in modo da mantenerlo in conduzione.

Riconfigurabili

SRAM

Le linee del dispositivo sono prodotte in modo da essere sempre disconnesse. La programmazione consiste nel memorizzare un valore logico (`0` o `1`) in una cella di RAM statica.

Connessioni globali

Alcuni tipi di dispositivi dispongono di linee di connessioni globali. La lunghezza delle linee è maggiore e di conseguenza lo sono i ritardi. Inoltre ogni linea è condivisa da molti elementi logici e può essere usata come uscita da solo uno di questi. La flessibilità è quindi limitata.

I dispositivi di questo tipo sono il PAL, il PLA, il GAL e le relative estensioni.

Logiche programmabili a 2 livelli

Sono usate per realizzare funzioni a due livelli. Dispongono di un numero di ingressi e uscite fissi e di due "piani": il piano AND è usato per la costruzione dei mintermini mentre il piano OR è usato per la connessione dei mintermini.

Ci sono 3 tipi principali:

1. PLA: entrambi i piani programmabili;
2. PAL: solo il piano AND è programmabile;
3. ROM: solo il piano AND è programmabile tramite un decoder.

PLA Consideriamo le seguenti funzioni:

$$f_1 = ab + ac' + a'b'c$$
$$f_2 = ab + ac + a'b'c$$

Possiamo riscriverle al seguente modo in SOP:

$$P_1 = ab$$
$$P_2 = ac$$
$$P_3 = ac'$$
$$P_4 = a'b'c$$

$$f_1 = P_1 + P_3 + P_4$$
$$f_2 = P_1 + P_2 + P_4$$

Una rappresentazione compatta sarà

Mintermine	a	b
11-	1	0
1-0	1	0
001	1	0
11-	0	1
1-1	0	1
001	0	1

I mintermini vengono programmati nel piano AND, mentre `a` e `b` vengono utilizzate per specificare quali connessioni vengono usate nel piano OR.

PAL Poiché il piano OR è a connessioni fisse, possiamo rappresentare solo funzioni con un numero limitato di mintermini. Inoltre dobbiamo tener conto del fatto che non è possibile condividere mintermini tra le funzioni, quindi potremo dover introdurre duplicazioni.

ROM Una memoria a sola lettura, o ROM, associa ad ogni indirizzo (ingresso) una parola (uscita). Le memorie ROM, grazie al decoder d'indirizzo, forniscono tutti i 2^n mintermini ottenibili dalle n variabili di ingresso x_i.

Gli ingressi del decoder saranno le variabili x_i mentre le uscite sono tutti i mintermini costruiti a partire dalle variabili di ingresso.

Da un punto vista più matematico, il decoder associa ad ogni tupla $(x_1, \dots, x_i)$ degli ingressi un'altra tupla $(f_1, \dots, f_i)$ contenente i valori che le varie funzioni assumono per $(x_1, \dots, x_i)$.

Circuiti sequenziali e macchine a stati finiti

Fai prima a studiarteli dal libro bro (capitoli 6, 7, 8).