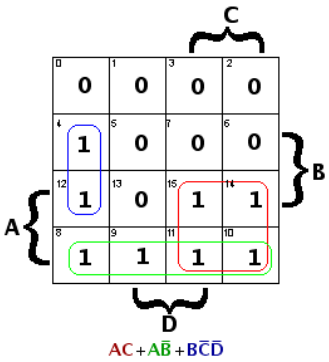
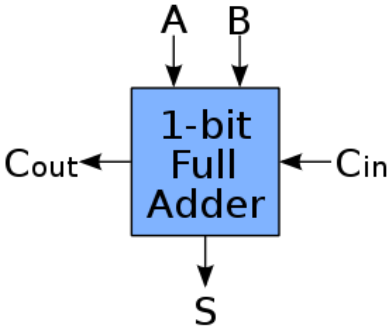
Reti Logiche Combinatorie





Materia: Architetture degli Elaboratori – 6 CFU

Facoltà di Scienze Matematiche Fisiche Naturali – Corso di Laurea in Informatica

Anno Accademico 2013/2014 - Università degli Studi di Palermo

Realizzato dalle lezioni della Professoressa

Simona Rombo

Scritto da

Alessio Lombardo

02 Luglio 2014 – Revisione 3

Indice

Nota Bene: Questa raccolta di teoremi e argomenti di Architetture degli Elaboratori è totalmente artigianale e pertanto non si assicura che tutte le informazioni riportate siano corrette. Preghiamo il lettore di segnalare eventuali errori ed inesattezze all'indirizzo mail **alessiox.94@hotmail.it** . Grazie.

Alessio Lombardo

Introduzione

Le reti logiche sono un insieme di dispositivi che, dato un input, eseguono una elaborazione per ottenere un certo output. Una rete logica ha in generale m ingressi ed n uscite. L'elaborazione viene eseguita in un certo tempo Δt.

Ogni ingresso ed ogni uscita (generalmente indicati con il nome di "bit", "linea", "morsetto") possono assumere alternativamente due stati. Lo stato 0 e lo stato 1.

Dal punto di vista matematico si dice che il "Dominio di Variazione" è: {0,1}. Di conseguenza, tutte le possibili combinazioni in ingresso saranno 2^m mentre quelle in uscita saranno 2^n.

Dal punto di vista elettrico, lo stato 0 è solitamente associato ad una tensione nulla mentre lo stato 1 è associato ad una certa tensione prestabilita (5 Volt per esempio). I dispositivi che rispettano questa convenzione sono detti "a logica positiva" mentre i bit in ingresso ed uscita sono detti "attivi alti". Viceversa, alcuni dispositivi funzionano "a logica negativa" con bit "attivi bassi" ovvero associano lo stato 0 al valore di tensione alto e lo stato 1 al valore di tensione nullo.

Le tensioni intermedie vengono generalmente ricondotte allo stato più vicino in modo tale che non siano possibili stati "intermedi".

Lo schema logico di una Rete Logica è il seguente:

Le reti logiche possono essere suddivise in 2 famiglie:

- Reti Logiche Combinatorie: Dato un input, trascorso un tempo Δt trascurabile si ottiene un output. Si tratta in sostanza di una "funzione" con dominio corrispondente agli input e codominio corrispondente agli output. Le reti di questo tipo non hanno nessuna capacità di memorizzazione e sono la base per la costruzione delle Reti Sequenziali.
- Reti Logiche Sequenziali: Data una sequenza di input, trascorso un tempo Δt si ottiene un output. Le reti di questo tipo sono anche dette "Con Memoria" proprio perché sono in grado di memorizzare un certo numero di bit per un determinato tempo.

In questo testo, si tratterà la sintesi di circuiti combinatori per mezzo delle Mappe di Karnaugh e si analizzeranno nel dettaglio alcuni dispositivi combinatori di fondamentale importanza per la realizzazione di un Sistema di Elaborazione dei Dati.

Funzioni di base e Porte Logiche

Come anticipato nel precedente paragrafo, una Rete Combinatoria è una funzione logica che per ogni input fornisce in uscita una ed una sola combinazione. Il tempo di "commutazione" ovvero il tempo che impiegano i segnali elettrici ad attraversare la rete è trascurabile al momento. Quindi, dato un input si suppone che istantaneamente si ottiene un output.

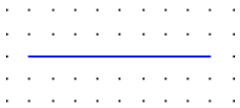
Ad ogni funzione logica può essere associata una "tabella di verità" che descrive il comportamento che deve assumere la rete logica.

Le funzioni logiche di base sono dette "porte logiche" o "logic gate" e sono l'elemento più piccolo che può comporre una rete logica. Le porte logiche sono rappresentate negli schemi elettrici con una simbologia standard che li rende immediatamente riconoscibili.

Si descrivono adesso le funzioni di base che verranno utilizzate in questo testo e la loro rappresentazione:

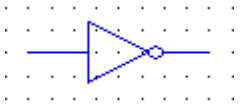
Funzione Identità: Nessun cambiamento. Non si tratta di una vera Porta Logica poiché non esegue nessuna funzionalità dal punto di vista elettrico e logico.

x	I(x)
0	**0**
1	**1**



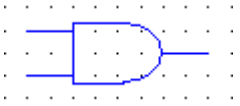
Porta NOT: Negazione. Si ottiene 1 se l'input è 0.

x	$\bar{x}$
0	**1**
1	**0**



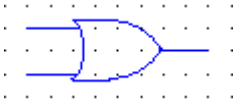
Porta AND: Congiunzione Logica. Si ottiene 1 se tutti gli input sono 1.

x	y	x·y
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**



Porta OR: Disgiunzione Logica. Si ottiene 1 se almeno un input è 1.

x	y	x+y
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**



Ovviamente sarà possibile utilizzare anche porte con più di 2 ingressi purché l'uscita rimanga una ed una soltanto.

Attraverso l'utilizzo delle porte OR, AND e NOT che rappresentano l'insieme degli operatori {+,·, ¯} è possibile sintetizzare qualsiasi funzione logica.

Inoltre, sfruttando i Teoremi di De Morgan, è possibile costruire la funzione AND utilizzando solo OR e NOT e la funzione OR utilizzando solo AND e NOT. In definitiva, si ottengono due insiemi di "Operatori Minimali" con i quali è possibile sintetizzare qualsiasi rete logica: {+, ¯} e {·, ¯}.

Si riportano per completezza le due relazioni di De Morgan:

$\overline{X \cdot Y} = \bar{X} + \bar{Y}$

$\overline{X + Y} = \bar{X} \cdot \bar{Y}$

È importante chiarire il fatto che esistono reti logiche fisicamente diverse ma che logicamente rappresentano la stessa funzione. In questi casi si parla di "reti logiche equivalenti" ovvero di reti che per ogni input forniscono in uscita la stessa configurazione e quindi hanno la stessa identica tabella di verità.

L'obiettivo del progettista è quello di trovare un rete logica quanto più veloce ed economica possibile.

Il concetto di velocità è legato al fatto che un segnale elettrico che attraversa una porta logica impiega un certo tempo detto "di propagazione". Infatti le porte logiche, che qui vengono considerate "indivisibili", sono in realtà composte da diversi transistor che necessitano di un certo tempo per cambiare stato.

Il massimo numero di porte che un segnale in ingresso deve attraversare per raggiungere un'uscita è detto "Livello" della rete logica. Nel caso in cui ci siano più input e più output bisogna considerare la più lunga combinazione di instradamenti che può percorrere il segnale. Vista la relativa semplicità strutturale della porta NOT (solo due transistor), essa viene esclusa dal calcolo dei Livelli della rete poiché il suo tempo di propagazione è praticamente nullo.

Risulta di fondamentale importanza sintetizzare la rete logica in modo tale che abbia il più basso numero di Livelli.

Il punto di vista economico è invece legato al numero di porte utilizzate. Se il numero di porte è minore, è ovvio che il costo di realizzazione della rete logica sarà quasi certamente più basso.

Nei prossimi paragrafi si studierà nel dettaglio la sintesi di reti logiche a 2 livelli attraverso l'uso delle Mappe di Karnaugh.

Mintermini e Implicanti

Un Mintermine, indicato con P_i^k, è una congiunzione (AND) di k variabili booleane considerate dirette o negate tale che il numero binario ottenuto dalla combinazione delle variabili sia i. In totale si avranno 2^k possibili mintermini.

Esempi (il segno "·" viene omesso):

$P_3^4 = \bar{a}b\bar{c}d$ $P_5^5 = \bar{a}b\bar{c}\bar{d}e$ $P_1^2 = \bar{a}b$

Una qualsiasi funzione logica può essere espressa come disgiunzione (OR) dei mintermini associati agli 1 (chiamati implicanti).

Come esempio, si consideri la funzione "implicazione logica" data dalla seguente tabella di verità (nella quale sono inclusi i mintermini corrispondenti agli ingressi):

x	y	x ⇒ y	Mintermine
0	0	1	P_0^2
0	1	1	P_1^2
1	0	0	P_2^2
1	1	1	P_3^2

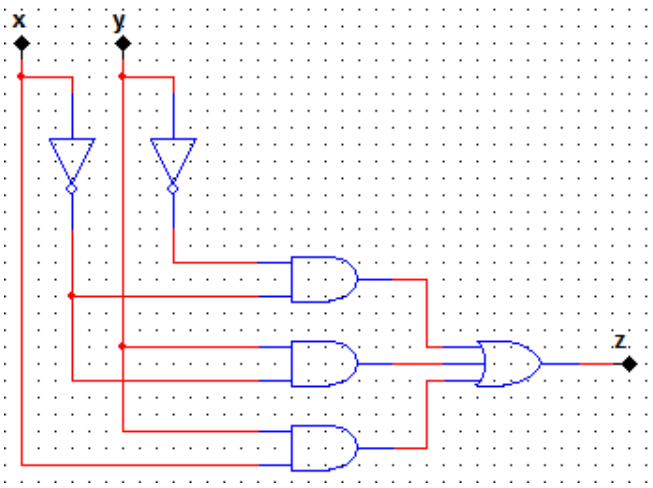
I mintermini da considerare (tutti definiti in 2 variabili) saranno solamente quelli a cui corrisponde un 1 in uscita, ovvero gli implicanti: P_0^2, P_1^2, P_3^2.

Il mintermine P_2^2 non va considerato poiché ad esso corrisponde un uscita 0 e quindi non è un implicante.

A questo punto gli implicanti andranno sommati logicamente per ottenere la funzione desiderata:

$x \Rightarrow y = P_0^2 + P_1^2 + P_3^2 = \bar{x}\bar{y} + \bar{x}y + xy$

Lo schema elettrico corrispondente sarà il seguente:



Gli implicanti risultano di fondamentale importanza per comprendere le Mappe di Karnaugh.

Mappe di Karnaugh

Le Mappe di Karnaugh sono uno dei metodi più utilizzati per la sintesi dei circuiti combinatori. Permettono di trasformare una tabella di verità nella corrispettiva rete logica senza particolari trasformazioni matematiche.

Una limitazione di questo metodo è data dal fatto che è particolarmente complesso lavorare su tabelle di verità che hanno 6 o più ingressi.

Nel caso in cui ci siano più uscite, è possibile applicare questo metodo per ogni uscita, semplicemente ignorando le altre.

Prima di procedere, è necessario definire il concetto di continuità logica:

Una coppia di numeri binari si definisce "logicamente contigua" se il primo numero differisce dal secondo solamente per una cifra binaria.

Esempi:

000 e 001 sono logicamente contigui.

000 e 101 non sono logicamente contigui.

Le mappe di Karnaugh sono tabelle formate da 2^k celle dove k indica il numero di variabili di input della tabella di verità considerata. Ogni cella è identificata da una certa configurazione di bit che rappresenta un mintermine.

Due celle sono fisicamente contigue se e solo se le configurazioni di bit delle due celle sono logicamente contigue.

Le mappe di Karnaugh rispettivamente per 2,3,4,5 e 6 variabili sono le seguenti:

x \ y	0	1
0		
1		

---

x \ yz	00	01	11	10
0				
1				

---

xy \ zt	00	01	11	10
00				
01				
11				
10				

xy \ zt	00	01	11	10
00				
01				
11				
10				

w=0

xy \ zt	00	01	11	10
00				
01				
11				
10				

w=1

xy \ zt	00	01	11	10
00				
01				
11				
10				

wu=00

xy \ zt	00	01	11	10
00				
01				
11				
10				

wu=01

xy \ zt	00	01	11	10
00				
01				
11				
10				

wu=10

xy \ zt	00	01	11	10
00				
01				
11				
10				

wu=11

Si può notare che, per rispettare la contiguità logica, la numerazione delle variabili non prosegue in modo crescente ma rispetta il "Codice Gray" ovvero una numerazione binaria nella quale ogni numero è logicamente contiguo al suo successivo.

Nel caso in cui si considerino 5 o 6 variabili, non è più sufficiente utilizzare una mappa ma se ne utilizzano rispettivamente 2 e 4. In questo caso le mappe si diranno "contigue per sovrapposizione" poiché una cella di una mappa resta fisicamente contigua alla rispettiva cella dell'altra mappa. In altri termini, si può immaginare che le 2 o 4 mappe siano sovrapposte fra loro e che formino un "parallelepipedo" 4X4X2 o un "cubo" 4X4X4 di celle/mintermini.

All'interno della mappa vanno contrassegnate con un 1 solo le celle che rappresentano un implicante della funzione, ovvero solo le celle a cui corrisponde un'uscita pari ad 1.

Come esempio, si considera la tabella di verità e la mappa di Karnaugh di una funzione che riporta in uscita 1 solo se i tre bit in ingresso formano un numero binario multiplo di 3:

x	y	z	t
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

x \ yz	00	01	11	10
0	1		1	
1				1

Nel caso in cui ci siano implicanti contigui è possibile attuare delle semplificazioni considerando degli implicanti di dimensione maggiore che "coprono" più celle.

Si definisce per prima cosa il concetto di Implicante Primo.

Si tratta di un Implicante che gode delle seguenti proprietà:

- È Massimale, ovvero copre più celle possibile.
- È composto da celle che contengono esclusivamente 1 (altrimenti non sarebbe un Implicante).
- È composto da un numero di celle che è potenza del 2.

La semplificazione può essere attuata non considerando nell'Implicante Primo le variabili che cambiano stato.

Esempio:

x \ y	0	1
0	1	
1	1	

L'implicante cerchiato è un Implicante Primo nel quale la variabile x cambia stato. Quindi si può scrivere l'implicante considerando solo l'altra variabile. In definitiva, supponendo che z sia l'uscita, si ottiene:

z=ȳ

Gli Implicanti Primi possono eventualmente sovrapporsi. Tuttavia, se un Implicante Primo è totalmente sovrapposto dagli altri, allora si dice "non essenziale" o "ridondante" e non verrà considerato come implicante.

Esempio:

ab \ cd	00	01	11	10
00			1	
01	1	1	1	
11		1	1	1
10		1		

Si nota che l'Implicante Primo cerchiato in rosso (che corrisponde a "bd") è ridondante e quindi non verrà considerato.

z = ābc̄ + ācd + abc + ac̄d

La forma algebrica ottenuta è detta "ottimale" proprio perché non presenta ridondanze.

Si noti il fatto che senza l'utilizzo della Mappa di Karnaugh, la sintesi di quest'ultima rete logica sarebbe risultata abbastanza complessa, soprattutto dal punto di vista elettrico (bisognava considerare la disgiunzione fra gli 8 implicanti non primi, ciascuno dei quali formato da 4 variabili).

Funzioni non completamente specificate

Fin ora si sono considerate funzioni logiche e tabelle di verità nelle quali ad ogni input corrisponde uno ed un solo output.

Nella realtà capita spesso di progettare reti logiche nelle quali alcune configurazioni in ingresso non siano possibili o quantomeno non siano previste. In questi casi, non è importante sapere quale valore assumono le uscite poiché queste stesse uscite sono frutto di combinazioni in ingresso impossibili.

Dal punto di vista matematico, queste configurazioni si possono immaginare come "punti di discontinuità" della funzione.

Le uscite in corrispondenza di input non specificati sono indicate con "X" o in certi casi con "-".

Una tabella non completamente specificata può essere la seguente (funzione che rileva un numero binario dispari da 1 a 9):

x_3	x_2	x_1	x_0	z_1
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Le configurazioni non specificate sono molto vantaggiose se utilizzate insieme alle mappe di Karnaugh. Nell'esempio appena visto, la mappa corrispondente è:

x_0x_1 \ x_2x_3	00	01	11	10
00		1	1	
01		1	1	
11	X	X	X	X
10		1	X	X

Si noti che all'interno della mappa sono state inserite anche le uscite delle configurazioni non specificate, come se si trattasse di uscite 1.

La differenza sostanziale è che gli implicanti X, a differenza degli implicanti 1, possono essere scelti in modo del tutto facoltativo.

Se sono utili per creare un implicante primo più grande vengono considerati, altrimenti vengono ignorati.

Per esempio, non ha senso considerare un implicante primo formato solo da implicanti X.

Nel caso in oggetto, la forma algebrica ottenuta sarà la seguente:

x_0x_1 \ x_2x_3	00	01	11	10
00		1	1	
01		1	1	
11	X	X	X	X
10		1	X	X

$z = x_3$

Senza gli implicanti X, la sintesi del circuito sarebbe stata decisamente più complessa.

Un altro esempio nel quale si utilizzano le configurazioni non specificate è il Decoder per Display a 7 segmenti (chiamato anche “BCD to 7 Segment”). Nella rete logica del decoder, le configurazioni che vanno da 10 a 15 sono non specificate proprio perché un unico Display deve mostrare solitamente solo le cifre che vanno da 0 a 9.

Un esempio di Decoder “BCD to 7 Segment” che utilizza le configurazioni non specificate è l’integrato 7447. Se si provasse ad inserire in input un numero superiore a 9, nel display ad esso collegato si otterrebbe una configurazione che non rappresenta nessun numero.

L’integrato DM9368 utilizza invece tutte le configurazioni e visualizza quindi anche le cifre superiori a 9 indicandole con le lettere A,B…F.

Encoder e Decoder

Gli Encoder o Coder (in italiano Codificatori) e i Decoder (in italiano Decodificatori) sono dispositivi combinatori molto utilizzati nella costruzioni di sistemi informatici.

L'Encoder è un dispositivo che possiede 2^n linee in ingresso ed n linee in uscita.

Le configurazioni valide in ingresso sono solamente quelle che prevedono una ed una sola linea ad 1. Le altre 2^n-1 linee rimanenti devono essere 0.

Di conseguenza, non si avranno 2^{2*n} possibili ingressi ma solamente 2^n.

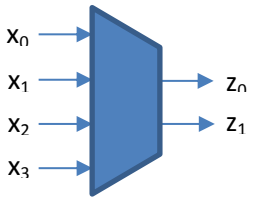
L'obiettivo della rete logica dell'Encoder è quello di riportare in uscita il numero binario corrispondente all'ingresso attivo in quell'istante. In simboli:

$z_0z_1...z_{n-1}=i$

Dove $z_0z_1...z_{n-1}$ sono le cifre binarie del numero in uscita ed "i" è l'indice della linea attiva in ingresso.

La tabella di verità dell'Encoder dipende ovviamente dal numero di ingressi/uscite. Si considera come esempio un Encoder "4 to 2" (con 4 ingressi e 2 uscite) e il suo schema elettrico:

x_3	x_2	x_1	x_0	z_0	z_1
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
Altre configurazioni				Errore	



Notare che l'indice dell'ingresso attivo è proprio il numero binario ottenuto alle uscite.

Esiste in commercio l'Encoder "8 to 3" 74148 con ingressi attivi bassi che, utilizzato in parallelo ad altri Encoder dello stesso tipo, può essere utilizzato per creare Encoder di taglia maggiore ("16 to 4", "32 to 5", ecc.).

Il Decoder è un dispositivo che si comporta in modo opposto rispetto all'Encoder.

Possiede n ingressi e 2^n uscite e, a differenza dell'Encoder, ogni configurazione in ingresso è valida.

L'obiettivo del Decoder è quello di attivare la linea in uscita corrispondente al numero binario fornito in ingresso. In simboli:

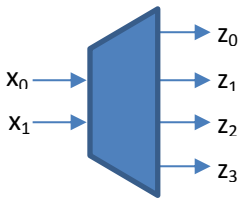
$z_i=1 \Leftrightarrow x_0x_1x_2x_{n-1}=i$

$z_j=0 \ \forall \ j\neq i$

Dove $x_0x_1...x_{n-1}$ rappresentano le cifre binarie del numero fornito in ingresso ed "i" è l'indice della linea attiva in uscita. Le altre linee, indicate con "j", saranno disattivate.

Anche in questo caso, la tabella di verità dipende dal numero di ingressi-uscite del decoder. La tabella sottostante, con il relativo schema elettrico, è valida per un Decoder "2 to 4" (2 ingressi e 4 uscite):

x_0	x_1	z_3	z_2	z_1	z_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



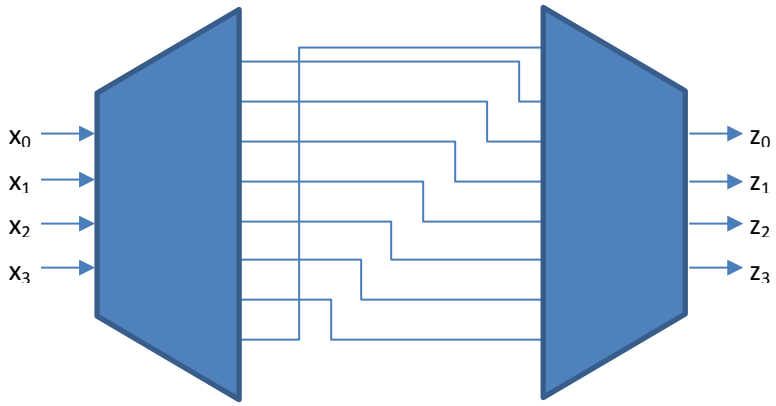
Notare il fatto che le tabelle del Decoder e dell'Encoder sono invertite: Gli ingressi di una corrispondono alle uscite dell'altra e viceversa. Proprio per questo, se si collegano insieme un Decoder e un Encoder (e viceversa) si ottiene la funzione identità. Ovvero, l'output corrispondente ad un input è l'input stesso.

In commercio si trovano gli integrati 74139, 74138 e 74154, rispettivamente Decoder "2 to 4", "3 to 8" e "4 to 16" con uscite attive basse.

Inserendo fra un Decoder ed un Encoder un'opportuna rete logica chiamata "Switch" si ottiene un dispositivo chiamato "Transcoder". Questo dispositivo accetta in input un numero binario ed emette in output un altro numero binario. Lo switch si occupa di "scambiare" le linee che collegano il Decoder con l'Encoder e di fatto attua una conversione del numero presente in input.

Per esempio, i Trancoder possono essere utilizzati per convertire un numero binario espresso in NBN (Notazione Binaria Naturale, anche detta BCD: Binary Coded Decimal) in altre "codifiche" come il codice "Eccesso 3", il codice "Gray", codice "Aiken", ecc.

Come esempio di Transcoder, si riporta un dispositivo che accetta in input un numero binario a 3 bit e lo incrementa di 1. Se esso non è rappresentabile in 3 bit, allora riporta in uscita lo 0:



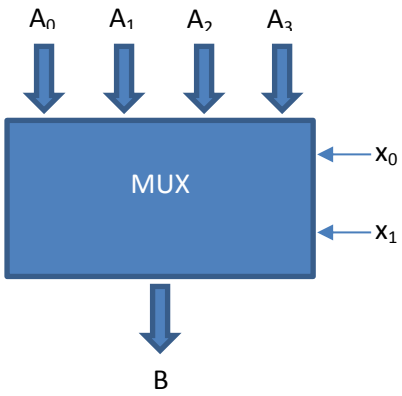
Multiplexer e Demultiplexer

I Multiplexer (Selettori d'Ingresso) e i Demultiplexer (Selettori d'Uscita) sono dispositivi che permettono di selezionare uno o più ingressi e di convogliarli ad una o più uscite. Sia i Multiplexer che i Demultiplexer sono in grado di agire non soltanto su singole linee ma anche sui "Bus".

I "Bus" sono un insieme di linee disposte in parallelo fra loro che solitamente sono utilizzati per collegare fra loro vari dispositivi all'interno di un'architettura. Per esempio, 4 linee che rappresentano le 4 cifre di un numero binario possono essere considerate un Bus.

I Multiplexer (indicati spesso con "MUX") possiedono 2^n ingressi, una sola uscita ed n "selettori". I selettori servono a convogliare uno solo degli ingressi verso l'uscita, ignorando tutti gli altri ingressi. Inserendo il numero "i" come selettore, l'ingresso "i-esimo" sarà convogliato verso l'uscita.

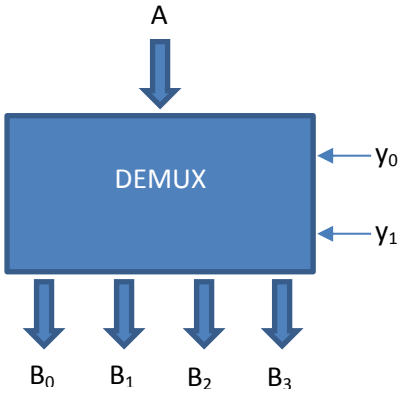
Sotto, lo schema logico di un Multiplexer a 4 ingressi (gli ingressi e l'uscita sono Bus di generica dimensione):



Dispositivi commerciali di questo tipo sono gli integrati 74150 (16 ingressi), 74151 (8 ingressi), 74153 (4 ingressi – doppio) che agiscono su ingressi e uscita a bit singoli (non sui Bus).

I Demultiplexer (indicati con "DEMUX") sono i dispositivi duali dei multiplexer. Possiedono un solo ingresso, 2^n uscite ed n "selettori". Questa volta, i selettori convogliano l'entrata verso una delle uscite. Al valore "i" inserito come selettore corrisponderà l'uscita "i-esima" che quindi sarà messa in collegamento diretto con l'ingresso. Le altre uscite non saranno considerate.

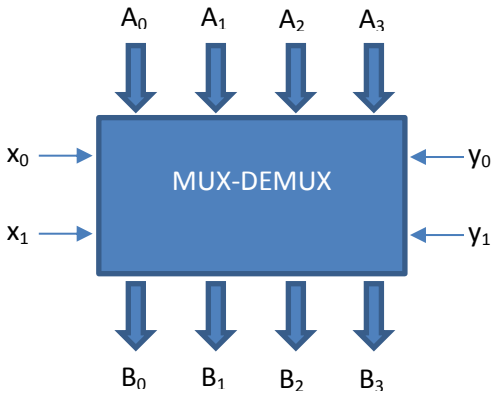
Si riporta lo schema logico di un Demultiplexer a 4 uscite (anche in questo caso si considerano dei Bus):



I Demultiplexer con ingresso ed uscite a singolo bit sono elettricamente ed anche logicamente molto simili ai decoder. Per questo, gli integrati che svolgono la funzione di Decoder possono anche essere utilizzati come Demultiplexer (sempre per singoli bit e non per i Bus).

Combinando un Multiplexer con un Demultiplexer è possibile ottenere un "MUX-DEMUX". Dispositivi di questo genere hanno quindi 2^m ingressi, 2^n uscite, m selettori di ingresso ed n selettori di uscita. In questo modo è possibile selezionare una delle entrate e riportarla in una delle uscite.

Un esempio di schema logico di un MUX-DEMUX a 4 ingressi e 4 uscite è il seguente:



Sommatore Parallelo

Il Sommatore Parallelo (o semplicemente Sommatore) è un dispositivo combinatorio che effettua la somma aritmetica fra 2 numeri binari di n bit.

Un Sommatore ad n bit è a sua volta formato da n Sommatori che sommano i singoli bit singolarmente.

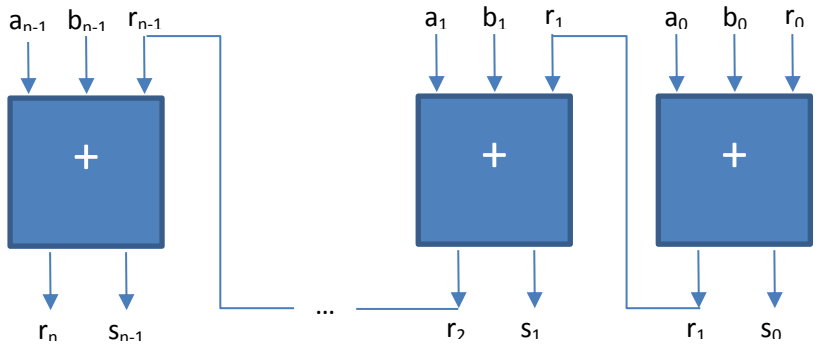
Nel caso in cui si sommino numeri a più bit, bisogna anche considerare i riporti che si generano durante la somma. Per questo, il Sommatore non solo somma i due bit "i-esimi", ma somma anche il riporto iniziale e genera il riporto da fornire al sommatore successivo.

La tabella di verità del sommatore che somma la coppia i-esima di bit sarà quindi la seguente:

a_i	b_i	r_i	s_i	r_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Dove ovviamente a_i e b_i sono i bit da sommare, r_i è il riporto iniziale, si la somma e r_{i+1} il riporto finale.

Lo schema logico di un Sommatore ad n bit è il seguente:



Si nota che i Sommatori sono tutti collegati fra loro attraverso i bit di riporto. Una configurazione di questo tipo è chiamata in gergo "Ripple Carry" (propagazione del riporto).

Il primo bit di riporto r_0 solitamente è posto a 0 se si deve effettuare la semplice somma fra i due numeri. Se viene impostato ad 1 la somma viene incrementata di 1. Il bit di riporto finale r_n è solitamente ignorato nelle architetture più semplici. Nei microprocessori viene utilizzato per altre funzionalità qui omesse.

Adesso si studierà come creare lo schema elettrico di un Sommatore a partire dalla tabella di verità già illustrata.

Esistono due approcci diversi che portano allo stesso risultato dal punto di vista funzionale:

- Approccio Half-Adder/Full-Adder: più economico (usa meno porte logiche) ma più lento e didatticamente più complesso da illustrare (si utilizzano porte EXOR). Nella realtà, i Sommatori sono realizzati utilizzando proprio questo approccio e la maggioranza dei testi illustrano questo metodo.
- Approccio classico con le mappe di Karnaugh: molto più costoso ma leggermente più veloce e didatticamente più semplice. Nella realtà, Sommatori di questo tipo sono praticamente inesistenti.

Si illustrerà nel dettaglio solamente il secondo approccio.

Si considerano dunque le mappe di Karnaugh associate alla tabella di verità e si creano le corrispettive funzioni risultanti:

Somma (s_i):

r_i \ a_ib_i	00	01	11	10
0		1		1
1	1		1	

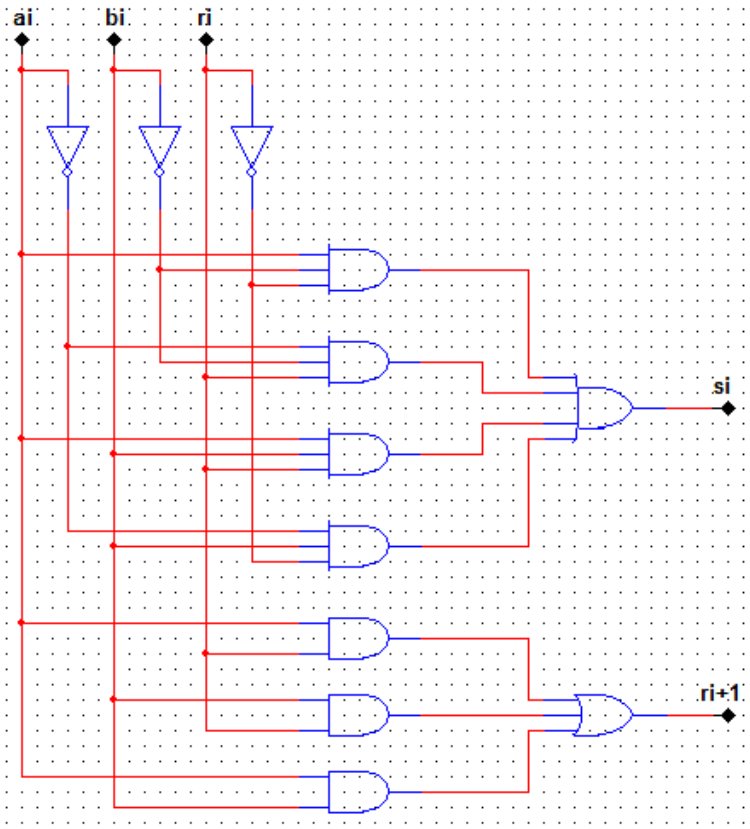
Riporto (r_{i+1}):

r_i \ a_ib_i	00	01	11	10
0			1	
1		1	1	1

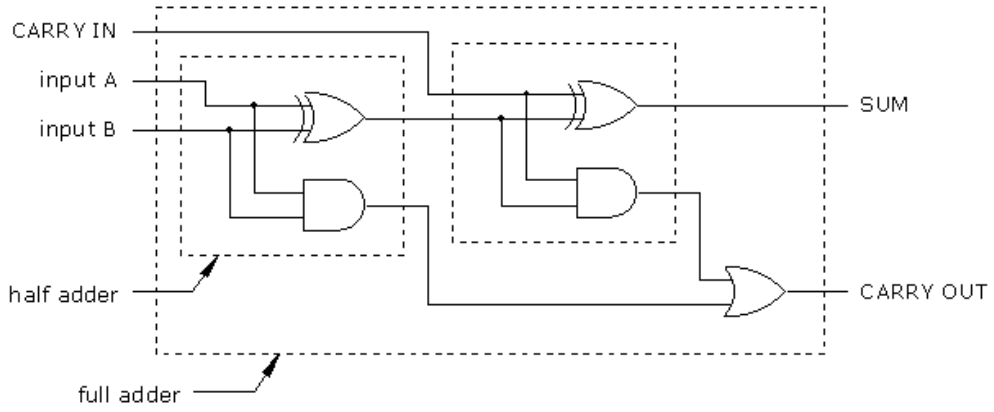
$$s_i = a_i\bar{b}_i\bar{r}_i + \bar{a}_i\bar{b}_i r_i + a_i b_i r_i + \bar{a}_i b_i \bar{r}_i$$

$$r_{i+1} = a_i r_i + b_i r_i + a_i b_i$$

A questo punto sarà possibile creare lo schema elettrico del Sommatore:



Per completezza, si riporta anche lo schema elettrico di un Sommatore creato con l'approccio Half-Adder/Full-Adder:



Si nota immediatamente il motivo per il quale è preferibile quest'ultimo schema rispetto al precedente: le porte logiche utilizzate sono meno della metà.

In commercio si trovano diversi Sommatori Paralleli tra i quali si ricorda l'integrato 74283 (per numeri a 4 bit).

Rilevatore di Overflow

L'Overflow (traboccamento) aritmetico è una condizione di errore che può avvenire durante l'esecuzione di una operazione aritmetica, in particolare, durante una somma fra due numeri binari.

Qualsiasi sommatore è infatti in grado di sommare un certo numero di bit. Se i due numeri da n bit da sommare generano un risultato di n+1 bit, allora si è generato un Overflow poiché l'ultimo bit andrà perduto e la somma sarà quindi scorretta.

Nel caso di somme fra numeri naturali (quindi positivi), basterebbe valutare lo stato del riporto finale per rilevare l'Overflow. Se il riporto finale è 0, la somma è corretta, viceversa, la somma è in Overflow. Tuttavia, nel caso in cui si effettuano somme fra numeri relativi (dotati di segno) non è più possibile considerare il riporto finale poiché i numeri negativi saranno in realtà sottratti e non sommati e di conseguenza il riporto finale perde la sua funzione di rilevatore di Overflow.

In questi casi per stabilire se l'operazione ha generato un Overflow è necessario focalizzarsi sui Most Significant Bit (MSb, ovvero i bit più significativi) degli addendi e del risultato.

Si ricorda che se il MSb è 0 allora il numero è positivo, altrimenti è negativo.

Se i MSb degli addendi sono discordi allora non si verifica mai un Overflow. Infatti, questo significherebbe che uno dei numeri è negativo mentre l'altro è positivo. Si tratta cioè di una sottrazione e quindi il risultato è sicuramente più piccolo dei numeri di partenza.

Se i MSb degli addendi e del risultato sono concordi allora non si verifica mai un Overflow. Questo significa che se gli addendi sono positivi, generano un numero positivo, se gli addendi sono negativi generano un numero positivo. Il risultato è quindi corretto.

Se i MSb degli addendi sono concordi ma il segno del risultato è discorde allora si verifica un Overflow. Infatti, la somma di due numeri positivi non può essere negativa e viceversa la somma di due numeri negativi non può essere positiva. Un risultato di questo tipo è sicuramente errato e l'unico tipo di errore che può generare un sommatore è appunto l'Overflow!

Si considera quindi un circuito combinatorio che, dati in input i tre bit MSb degli addendi e del risultato, genera un bit detto "di Supero" o "di Overflow" pari a 1 solo se si è verificato un Overflow.

La tabella di verità di un rilevatore di Overflow sarà quindi la seguente:

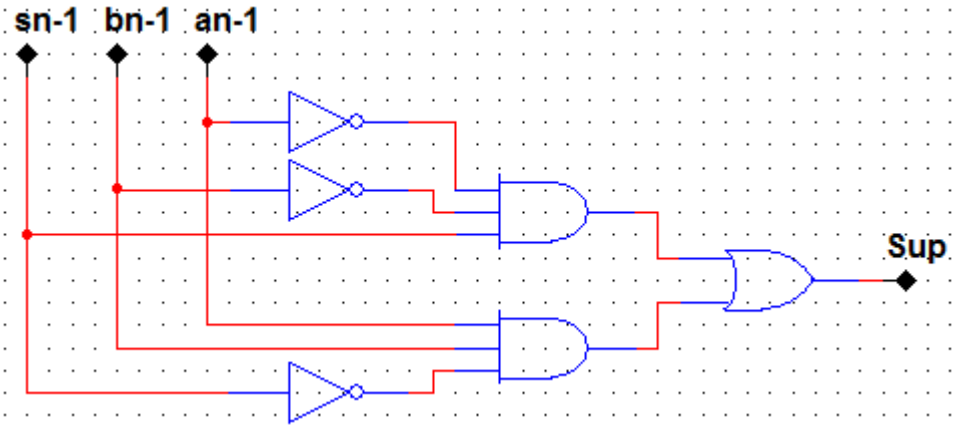
a_{n-1}	b_{n-1}	s_{n-1}	Sup	Descrizione
0	0	0	0	Operazione Corretta (addendi e risultato concordi)
0	0	1	1	Overflow (addendi concordi ma risultato discorde)
0	1	0	0	Operazione Corretta (addendi discordi)
0	1	1	0	Operazione Corretta (addendi discordi)
1	0	0	0	Operazione Corretta (addendi discordi)
1	0	1	0	Operazione Corretta (addendi discordi)
1	1	0	1	Overflow (addendi concordi ma risultato discorde)
1	1	1	0	Operazione Corretta (addendi e risultato concordi)

Si procede quindi alla sintesi del circuito combinatorio attraverso una mappa di Karnaugh:

s_{n-1} \ $a_{n-1}b_{n-1}$	00	01	11	10
0			1	
1	1			

Sup = $\bar{a}_n\bar{b}_n s_n + a_n b_n \bar{s}_n$

Lo schema elettrico è il seguente:



Arithmetic Logic Unit

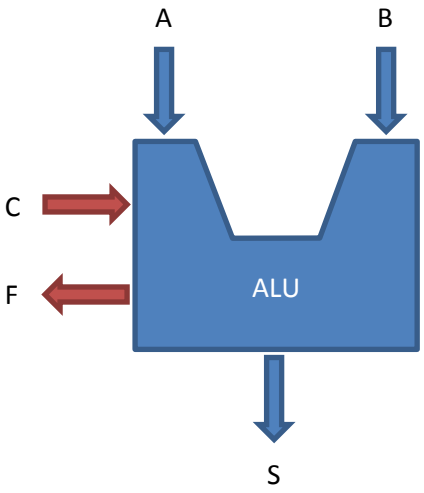
L'Arithmetic Logic Unit (ALU) è un dispositivo combinatorio utilizzato per eseguire operazioni logiche e aritmetiche su una coppia di valori binari di "n" bit.

Esempi di operazioni logiche sono: AND, OR, NAND, NOR, EXOR, EXNOR, NOT (o complemento ad 1), Selezione di uno dei due numeri, CLEAR (zero in uscita), SET (2^n-1 in uscita), SHL (Shift a sinistra – Moltiplicazione per 2), SHR (Shift a destra – Divisione per 2), Rotazione, ecc.

Esempi di operazioni aritmetiche sono: Somma, Sottrazione, Complemento a 2, Incremento, Decremento, Moltiplicazione, Divisione, ecc.

È importante sottolineare che l'ALU può eseguire esclusivamente calcoli su numeri binari interi (positivi e negativi). Per eseguire calcoli fra numeri binari frazionari è necessario un altro dispositivo chiamato FPU (Float Point Unit).

Il simbolo dell'ALU è il seguente:



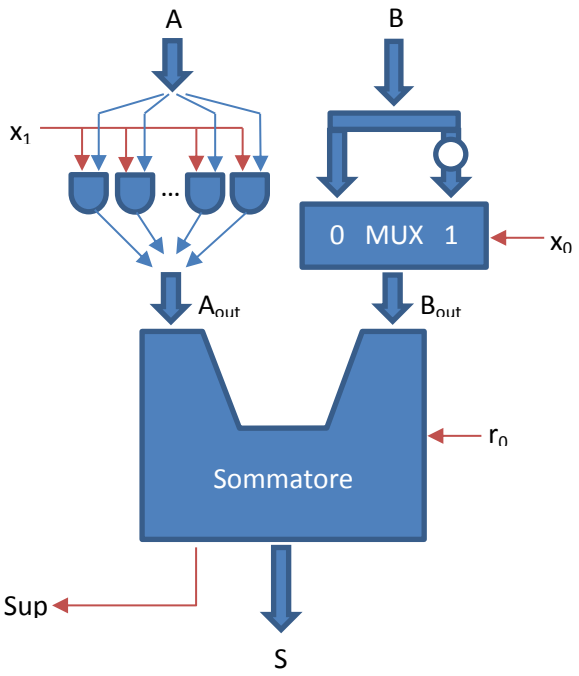
Gli ingressi sono:

- A: Il primo numero su cui eseguire il calcolo.
- B: Il secondo numero su cui eseguire il calcolo.
- C: Control (Selettori), ovvero un certo numero di bit che definiscono il tipo di operazione da effettuare (solitamente tra questi bit è presente anche il Riporto iniziale (o Carry IN) utilizzato proprio per definire nuove operazioni altrimenti impossibili).

Le uscite sono:

- S: Il numero risultante.
- F: Flags, un certo numero di bit utilizzati per segnalare eventuali particolarità dell'operazione appena eseguita (Overflow, riporto finale (Carry OUT), risultato negativo, divisione per zero, ecc.).

Si analizzerà adesso nel dettaglio lo schema logico di una semplice ALU in grado di effettuare alcune operazioni logiche e aritmetiche. Si ricaverà quindi la rispettiva tabella di verità:



Il "nucleo" dell'ALU è un semplice sommatore binario ad "n" bit con generatore del "bit di supero" (Overflow). Al suo interno sono presenti "n" sommatori collegati fra loro, ognuno dei quali si occupa di sommare ad uno ad uno i bit e di fornire l'eventuale riporto al sommatore più significativo. Il bit "r_0" è il riporto iniziale e quindi viene fornito al sommatore del bit meno significativo. Se è 0, allora viene eseguita la normale somma fra i due addendi. Se è 1, la somma viene incrementata di 1 poiché al sommatore meno significativo arriva un bit 1 in più da sommare. Il generatore del bit di supero è collegato al sommatore più significativo mentre il riporto in uscita di quest'ultimo viene ignorato.

r_0	Uscita (S)
0	$A_{out}+B_{out}$ (somma)
1	$A_{out}+B_{out}+1$ (somma e incremento)

Il bus A viene condotto ad un dispositivo in grado di bloccare/sbloccare il flusso dei dati agendo sul bit x_1. In particolare, ogni bit del bus viene condotto ad una entrata di una porta AND. L'altra entrata di tutte le porte AND è invece collegata al bit x_1. Quando questo bit è 0, tutte le uscite delle porte saranno 0, a prescindere dal numero binario che era presente sul bus A. Quando il bit è 1, il dispositivo si comporta come se non esistesse (in gergo elettronico si parla di "Modalità Transparente"). Ovvero, il numero viene semplicemente riportato alle uscite delle porte AND. Dispositivi di questo genere (in realtà leggermente più complessi) vengono chiamati "Buffer three-state".

x_1	Primo Addendo (A_{out})
0	0 (zero)
1	A (modalità trasparente)

Il bus B viene sdoppiato in due bus identici. Il secondo bus viene interamente negato attraverso n porte NOT, una per ogni bit del bus. I due bus, che conterranno rispettivamente B e $\bar{B}$, vengono quindi condotti ad un Multiplexer. Poiché questo MUX possiede 2 soli ingressi, per selezionarli è necessario un unico bit di

selezione indicato con x_0. Quando questo bit è 0, viene selezionato il bus contenente B e viene condotto all'uscita del MUX. Viceversa, viene selezionato il bus contenente $\bar{B}$.

x_0	Secondo Addendo (B_{out})
0	B (bus diretto)
1	$\bar{B}$ (bus negato)

Combinando le tre tabelle di verità, si ottiene la seguente:

x_0	x_1	r_0	Operazione	Descrizione
0	0	0	B	Selezione di B
0	0	1	B+1	Incremento di B
0	1	0	A+B	Somma
0	1	1	A+B+1	Somma e incremento
1	0	0	$\bar{B}$ = -B-1	Negazione/Complemento ad 1 di B
1	0	1	$\bar{B}$+1 = -B	Complemento a 2 di B
1	1	0	$\bar{B}$+A = A-B-1	Sottrazione e decremento
1	1	1	$\bar{B}$+A-1 = A-B	Sottrazione

Attenzione: "+" indica la normale somma binaria. In alcuni Datasheet di ALU indica invece l'OR logico fra i due numeri mentre la somma è indicata con "plus".

Si può notare che l'utilizzo dell'ingresso negato nel Multiplexer e del riporto iniziale permettono di ottenere l'opposto del numero B ovvero il suo "complemento a 2". Di conseguenza, sarà possibile eseguire anche la sottrazione fra A e B senza utilizzare dispositivi aggiuntivi.

Un'altra curiosità è il fatto che è possibile utilizzare questa ALU in modalità "quasi" trasparente. Si nota infatti che con la combinazione "010" si ottiene in uscita esattamente il bus B iniziale (ovvero si "seleziona" B, in modo simile a quanto avviene in un MUX o DEMUX). Tuttavia, lo stesso non si può dire per A. In alcune architetture di microprocessori, la modalità trasparente dell'ALU è fondamentale per movimentare i dati da una parte all'altra della CPU.

Due esempi di ALU commerciali (in verità ormai introvabili) sono gli integrati 74181 e 74381 che lavorano su numeri a 4 bit con 64 operazioni diverse.

Alea Statica e Sincronizzazione

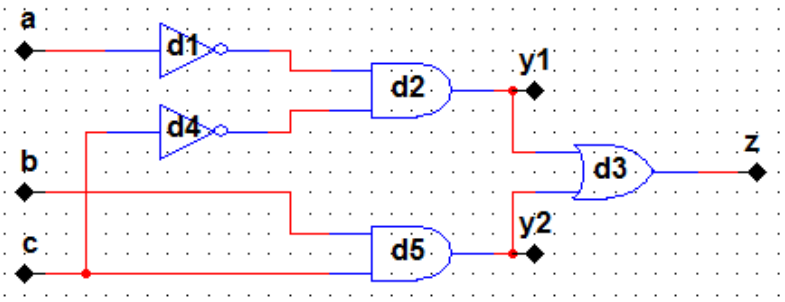
L'Alea Statica è un fenomeno temporaneo che si può verificare nei circuiti combinatori e che provoca una commutazione errata e non prevista nell'uscita o nelle uscite del circuito. Questo errore si verifica per due diverse cause:

1) I ritardi generati dalle singole porte logiche non sono sempre uguali fra loro. Quando si varia un ingresso, questi ritardi non uguali possono generare internamente delle commutazioni non previste che ovviamente si ripercuotono sull'uscita finale.
2) La variazione di due o più ingressi non è detto che avvenga nello stesso istante. Anche se in modo impercettibile, è possibile che un ingresso commuti leggermente prima di un altro e anche in questo caso si possono generare errori.

Si prenda come esempio il circuito riportato sotto:

a \ bc	00	01	11	10
0	1		1	1
1			1	

z = āc̄ + bc



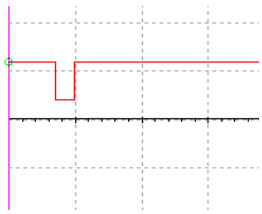
I ritardi delle singole porte sono indicati con "d1",…, "d5". Le uscite "y1" e "y2" sono uscite intermedie mentre l'uscita della rete combinatoria è "z".

Supponendo "a=0" e "b=1", si consideri adesso la variazione dell'ingresso "c" da 1 a 0 e il corrispettivo comportamento di "y1" e "y2".

Supponendo che i ritardi delle porte siano tutti uguali, all'aumentare del numero di porte aumenterà il tempo impiegato da un segnale per raggiungere una determinata porta. Poiché d5<d4+d2, la prima uscita intermedia a variare sarà "y2" e solo successivamente varierà "y1".

La tabella e il grafico (relativo a z in funzione del tempo) mostrano la variazione avvenuta:

a	b	c	y1	y2	z
0	1	1	0	1	1
			0	0	0
0	1	0	1	0	1



Si può notare che per un breve istante l'uscita "z" ha cambiato stato. Questo non era assolutamente previsto come si può notare dalla Mappa di Karnaugh della rete. Questa variazione è appunto un'Alea Statica.

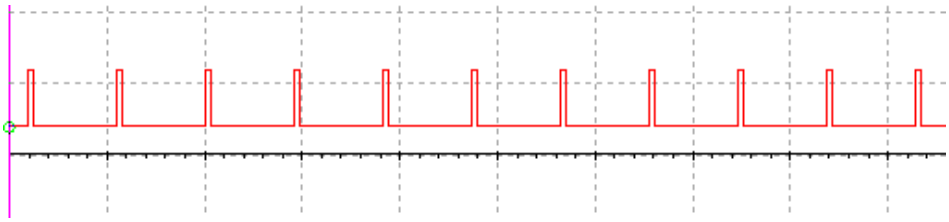
Per risolvere questo inconveniente è necessario “ridondare” la rete considerando tutti gli implicanti primi (anche quelli ridondanti). La mappa di Karnaugh del circuito senza alee statiche sarà quindi:

a \ bc	00	01	11	10
0	1		1	1
1			1	

z = āc̄ + bc + āb

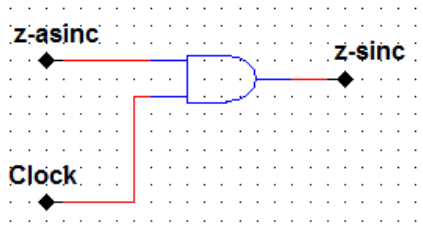
Questo meccanismo di ridondanza non risolve il caso in cui a variare siano più ingressi. Come spiegato sopra, gli ingressi possono non variare contemporaneamente e questo può generare errori dovuti al fatto che si considerano combinazioni di ingressi non previsti.

Una buona soluzione a questo problema è quella di rendere il circuito “sincrono” ovvero sincronizzato con un opportuno segnale chiamato “clock”. Il clock è un semplice segnale periodico ad onda quadra caratterizzato da “picchi” alti molto stretti:



Il clock viene inserito o nelle entrate o nelle uscite della rete combinatoria. Solitamente per fare ciò vengono utilizzate delle porte AND.

Si considererà il caso in cui il clock viene posizionato in uscita. In questo caso, in un’entrata della porta AND verrà inserita l’uscita della rete da sincronizzare, nell’altra entrata verrà inserito il segnale di clock e in uscita si otterrà l’uscita della rete sincronizzata. Il circuito sarà quindi il seguente (dove “z-asinc” è l’uscita del circuito combinatorio da sincronizzare):



Quando il clock è basso, ciò che avviene all’interno della rete è ininfluente. Sia le variazioni lecite che quelle non previste (alee statiche) saranno completamente ignorate poiché all’uscita della porta AND si avrà sempre 0. Quando il clock è alto l’uscita potrà essere letta. Questo meccanismo non evita la creazione di alee statiche ma fa in modo che esse avvengano nella fase “bassa” del clock, ovvero nel momento in cui l’uscita non viene letta.

Ovviamente, per far si che tutto funzioni correttamente, gli ingressi devono variare in modo tale da rispettare la frequenza del clock. In caso contrario si verifica uno “sfasamento” fra la variazione degli ingressi e le letture dell’uscita e questo ovviamente provocherà errori. Per questo, è bene utilizzare lo stesso generatore di clock per tutti i dispositivi combinatori presenti in un progetto.

Appunti

Questo foglio è lasciato libero per l’inserimento degli appunti del lettore.