CAPITOLO 5

PORTE LOGICHE

5.1 Introduzione ai circuiti logici combinatori

Lo scopo del presente e del successivo capitolo è di utilizzare le conoscenze acquisite nei capitoli precedenti per formulare criteri di progetto ed analisi di circuiti combinatori e di interfaccia.

I circuiti elettronici logici per sistemi digitali possono essere o combinatori o sequenziali. Un circuito combinatorio consiste di porte logiche le cui uscite sono, ad ogni istante, determinate direttamente dalla combinazione dei suoi ingressi in quell'istante, senza tenere conto della precedente evoluzione. *Un circuito combinatorio esegue un preciso processamento della informazione specificato completamente da un insieme di funzioni Booleane*.

I circuiti sequenziali usano, oltre alle porte logiche, elementi di memoria (Flip - Flop). Le loro uscite sono, ad ogni istante, funzione degli ingressi e dello stato dei suoi elementi di memoria *nello stesso istante,* ma lo stato degli elementi di memoria è funzione degli ingressi *precedenti quell'istante*. Come conseguenza di ciò le uscite di un circuito sequenziale dipendono non solo dagli ingressi attuali, ma anche dagli ingressi precedenti ed *il comportamento del circuito deve essere specificato da una sequenza temporale sia degli ingressi che dello stato delle sue memorie*.

Un circuito combinatorio consiste:

1) di variabili di ingresso;
2) di porte logiche;
3) di variabili di uscita.

Le porte logiche accettano segnali ai suoi ingressi e generano segnali in uscita, che sono, istante per istante, funzioni assegnate dei segnali di ingresso. La Fig. 5.0 mostra un diagramma a blocchi di un circuito combinatorio.

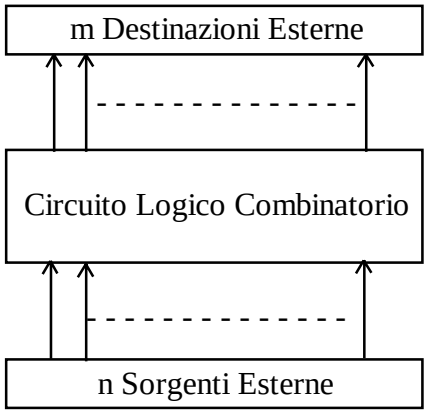


Fig. 5.0

Le *n* variabili binarie di ingresso provengono da sorgenti esterne; le *m* variabili binarie di uscita vanno ad una destinazione esterna. In molte applicazioni sia la sorgente che la destinazione delle variabile binarie sono dei registri di memoria posti sia in unità esterne vicine o lontane. Per definizione, i registri esterni non influenzano il comportamento del circuito combinatorio perché se lo facessero il sistema in totale sarebbe sequenziale.

Per *n* variabili di ingresso, ci sono 2^n combinazioni di valori binari. Per ogni una combinazione c'è una ed una sola possibile uscita. *Un circuito combinatorio è descritto da* m

$\leq 2^n$ *funzioni Booleane, una per ogni variabile di uscita espressa in termine delle* n *variabili di ingresso.*

Ciascuna delle variabili di ingresso può essere fornita per mezzo di uno o due fili. Se è fornita con un solo filo la variabile può essere in forma normale (non accentata) o complementata (accentata). Siccome, in una funzione Booleana, una variabile può apparire in forma normale o complementata è necessario dotare gli ingressi di un inverter, uno per ogni variabile non disponibile nella forma richiesta. Se una variabile è fornita con due fili, in uno comparirà la variabile in forma normale e nell'altro in forma complementata. In questo caso non si ha bisogno di includere inverter agli ingressi. Il tipo di celle binarie, utilizzate normalmente nei sistemi digitali, sono dei flip - flop che forniscono la variabile binaria in uscita sia in forma normale che complementata. Nel prosieguo noi supporremo di essere sempre in questo caso.

5.2 Variabili esterne e controlli

Prima di descrivere il procedimento di progetto di circuiti logici combinatori complessi, cosa che faremo dal prossimo capitolo, è bene analizzare i vari aspetti di uso e le caratteristiche di alcuni semplici operatori binari. Queste funzioni sono realizzati con una classe di circuiti logici integrati chiamata SSI (Small Scale Integration). Le funzioni logiche più complesse del prossimo capitolo fanno invece parte della classe di circuiti detti MSI (Medium Scale Integration)

Spesso, in un circuito logico combinatorio, è opportuno fare distinzione tra le variabili binarie esterne al sistema, e come già detto, trasferite ad esso attraverso registri di memoria, e *variabili di controllo* o *programmazione*. Queste ultime non sono diverse dalle prime ma hanno la particolarità che il loro valore controlla una o più modalità di funzionamento dell'intero sistema ed il cui valore è determinato dalla volontà dall'utente.

Es. 1: *Analizzare il circuito AND mostrato il Fig. 5.1 in cui C rappresenta la variabile di controllo.*

La funzione Booleana è ovviamente F = C f e la relativa tabella di verità è presentata nella Tab. 5.1. C è la variabile di controllo cioè una variabile il cui valore è determinato dall'utente, mentre la variabile f ha un valore che potrebbe a sua volta essere funzione di un certo numero di altre variabili su cui l'utente non ha controllo o esterne. Per es. è una funzione Booleana degli stati logici di un certo sistema fisico che si evolve autonomamente.

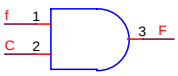


Fig. 5.1

C	f	F
0	0	0
0	1	0
1	0	0
1	1	1

Tab. 5.1

Dalla tabella di verità è evidente che se C = 0, F = 0 indipendente dal valore assunto da f. Possiamo chiamare questo: *stato controllato* dell'uscita F. Quando C = 1 il valore di F = f, chiameremo questo: *stato trasmesso* di F. La

variabile binaria C opera come un usciere che, se chiude la porta non fa passare nessuno, mentre se apre la porta passano tutte le persone che arrivano senza che queste subiscano alcun condizionamento. Questa visione allegorica spiega perché si parla di *porte logiche* quando ci si riferisce, in generale, ad operatori binari. Una porta AND, sotto questo punto di vista, può quindi essere considerata come un elemento che controlla, attraverso uno dei suoi ingressi, la trasmissione o meno di una variabile binaria. Questa può essere una variabile di ingresso o di uscita di un circuito logico. Possiamo riassumere il nostro ragionamento dicendo che:

a) Lo stato controllato dell'uscita di una AND a 2 ingressi è F = 0 e viene controllato dal valore 0 della variabile di controllo.

b) Lo stato trasmesso dell'uscita di una AND a 2 ingressi è F = f e viene trasmesso dal valore 1 della variabile di controllo.

Es. 2: *Analizzare il circuito OR mostrato il Fig. 5.2 in cui C rappresenta la variabile di controllo*.

La funzione Booleana è ovviamente F = C + f e la relativa tabella di verità è presentata nella Tab. 5.2. Da questa tabella è evidente che se C = 1, F = 1, mentre se C = 0, F = f. Cioè:

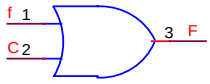


Fig. 5.2

a) Lo stato controllato dell'uscita di una OR a 2 ingressi è F = 1 e viene controllato dal valore 1 della variabile di controllo.

b) Lo stato trasmesso dell'uscita di una OR a 2 ingressi è F = f e viene trasmesso dal valore 0 della variabile di controllo.

C	f	F
0	0	0
0	1	1
1	0	1
1	1	1

Tab. 5.2

In questo caso lo stato controllato di F ed il valore della variabile di controllo sono il complemento di quelli relativi al circuito AND, tuttavia la variabile trasmessa non viene modificata.

Es. 3: *Analizzare il circuito NAND mostrato il Fig. 5.3 in cui C rappresenta la variabile di controllo*.

La funzione Booleana è ovviamente F = (C f)' e la relativa tabella di verità è presentata nella Tab. 5.3. Da questa tabella è evidente che se C = 0, F = 1, mentre se C = 1, F = f'. Cioè:

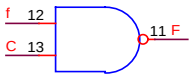


Fig. 5.3

a) Lo stato controllato dell'uscita di una NAND a 2 ingressi è F = 1 e viene controllato dal valore 0 della variabile di controllo.

b) Lo stato trasmesso dell'uscita di una NAND a 2 ingressi è F = f' e viene trasmesso dal valore 1 della variabile di controllo.

C	f	F
0	0	1
0	1	1
1	0	1
1	1	0

Tab. 5.3

Vediamo che in questo caso all'effetto di porta si è aggiunto una operazione logica sul valore della variabile trasmessa: viene complementata.

Es. 4: *Analizzare il circuito NOR mostrato il Fig. 5.4 in cui C rappresenta la variabile di controllo.*

La funzione Booleana è ovviamente F = (C + f)' e la relativa tabella di verità è presentata nella Tab. 5.4. Da questa tabella è evidente che se C = 1, F = 0, mentre se C = 0, F = f'. Cioè:

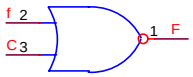


Fig. 5.4

a) Lo stato controllato dell'uscita di una NOR a 2 ingressi è F = 0 e viene controllato dal valore 1 della variabile di controllo.

b) Lo stato trasmesso dell'uscita di una NOR a 2 ingressi è F = f' e viene trasmesso dal valore 0 della variabile di controllo.

C	f	F
0	0	1
0	1	0
1	0	0
1	1	0

Tab. 5.4

In questo caso lo stato controllato di F è lo stesso che per l'AND, ma il valore della variabile di controllo e quella della trasmessa sono i valori complementari di quelli.

Possiamo riassumere quanto trovato con gli Es. 1 - 4 con la seguente tabella

Operatore Binario	stato controllato C	stato controllato F	stato trasmesso C	stato trasmesso F
AND	0	0	1	f
OR	1	1	0	f
NAND	0	1	1	f'
NOR	1	0	0	f'

Tab. 5.5

Questa tabella ci permette di dire che: quando abbiamo da controllare la trasmissione di una variabile binaria possiamo scegliere il valore dello stato controllato 0, o 1 indipendentemente dal fatto che lo stato trasmesso debba essere complementato o meno. In particolare *se il valore dello stato controllato deve essere*:

a) *0* useremo una NOR se vogliamo complementare il valore della variabile trasmessa, altrimenti useremo una AND;

b) *1* useremo una NAND se vogliamo complementare il valore della variabile trasmessa, altrimenti useremo una OR.

Es. 5: *Analizzare il circuito Exclusive OR mostrato il Fig. 5.5 in cui C rappresenta la variabile di controllo.*

La tabella di verità è presentata nella Tab. 5.6. Da questa tabella è evidente che se C = 0, F = f, mentre se C = 1, F = f'.

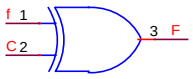


Fig. 5.5

In questo caso non è possibile definire uno stato controllato in quanto l'uscita dipende sempre dall'altra variabile di ingresso. Tuttavia la proprietà ora messa in luce è di grande importanza per la compatibilità di una data strumentazione verso altre che usano sia logica positiva che logica negativa. Se i due strumenti usano logica diversa diamo alla variabile di controllo C il valore 1, F sarà complementata. Se i due strumenti usano lo stesso tipo di

C	f	F
0	0	0
0	1	1
1	0	1
1	1	0

Tab. 5.6

logica diamo alla variabile di controllo C il valore 0; F *non* sarà complementata.

Es. 6: *Analizzare il circuito Equivalence mostrato il Fig. 5.6 in cui C rappresenta la variabile di controllo.*

C	f	F
0	0	1
0	1	0
1	0	0
1	1	1

Tab. 5.7

La tabella di verità è presentata nella Tab. 5.7. Da questa è evidente che se C = 0, F = f', mentre se C = 1, F = f. L'uso di questo circuito è analogo al quello del' XOR ma con i controlli invertiti.

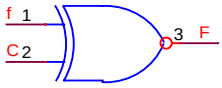


Fig. 5.6

5.3 Buffer Tri - State

Si consideri il circuito di Fig. 5.7a. dove un transistor CMOS, n-channel, è posto in serie ad un segnale logico, *f*, proveniente per es. dall'uscita di una funzione realizzata con la famiglia logica TTL. Come sappiamo (e si vede nell'equivalente meccanico), quando il segnale di gate è C= 0 il transistor si comporta come un interruttore aperto, mentre quando è C= 1, come un interruttore chiuso.

Nel primo caso il valore della variabile binaria, *f*, non si può propagare in uscita. La variabile *F* non ha allora un valore ben determinato di tensione (0 Volt o Vcc=+5V). Infatti non esiste

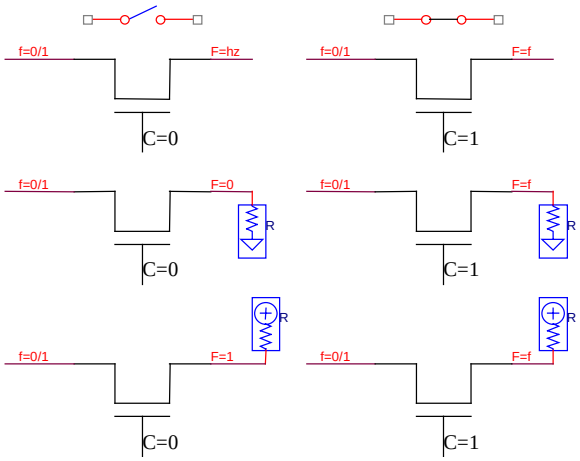


Fig. 5.7a Interruttori con Transistor *n-channel*

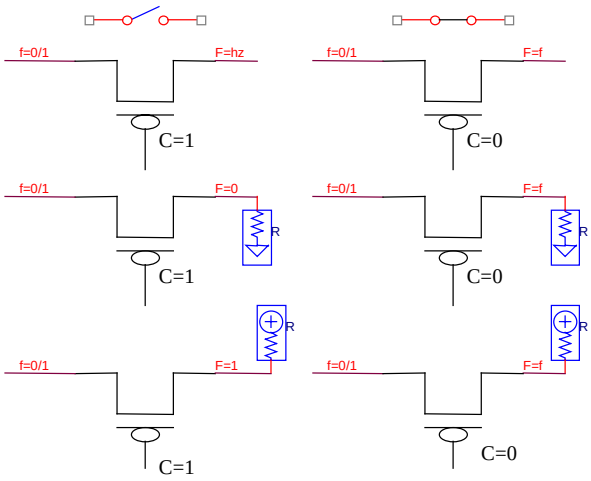


Fig. 5.7b Interruttori con Transistor *p-channel*

un percorso elettrico che connetta il terminale di uscita verso massa o verso la tensione di alimentazione. Quindi possiamo dire che *F* non assume un valore binario e, per il motivo ora detto, si dice che *F* è in uno stato di alta impedenza, **hz,** (*high impedence* all'inglese) o *Tri State* (il simbolo che, normalmente, si usa per indicare un'impedenza è la lettera *z* ed in

inglese alto si dice high). Per poter far si che *F* assuma un valore di tensione corrispondente ad uno dei livelli logici, dobbiamo modificare l'uscita in uno dei seguenti modi:

1) Collegare una resistenza, di valore opportuno, tra il terminale di uscita e massa come mostrato nel secondo rigo di Fig. 5.7a. Una resistenza così connessa si dice di *Pull Down*. Questa connessione fornisce il valore logico zero.
2) Collegare una resistenza tra il terminale di uscita e Vcc, come indicato nel terzo rigo della Fig. 5.7a. Una resistenza così connessa si dice di *Pull Up*. Questa connessione fornisce il valore logico uno.

Nel secondo caso F assume il valore logico determinato da quello della variabile di ingresso.
La tabella di verità, che descrive i tre casi di Fig. 5.7a, è data in Tab. 5.8

Da questa tabella notiamo che:

1) Il secondo rigo ha la stessa tabella di un circuito AND cioè: $F = C \cdot f$.
2) Il terzo rigo fornisce: F'= C f' => F = C'+f che è una implicazione: se C= 1 allora F = f se C= 0 => F = 1 (stato controllato)
3) Poiché la funzione del primo rigo non può essere descritta in termini di algebra Booleana, nella tabella, lo stato della variabile di uscita F quando C= 0, è indicato con *hz*. Questo per richiamare la caratteristiche di alta impedenza

C	F	I° F	II° F	III° F
0	0	hz	0	1
0	1	hz	0	1
1	0	0	0	0
1	1	1	1	1

Tab. 5.8

Poiché un circuito con queste caratteristiche è entrato a far parte, a tutti gli effetti, di tutte le famiglie logiche, gli è stato assegnato il simbolo grafico rappresentato in Fig. 5.8 ed in nome di *buffer tri-state*. Questo simbolo richiama quello di un buffer con l'aggiunta del terminale di controllo chiamato, normalmente, *OE*, che è un acronimo di Output Enable (abilitazione di uscita) giusta la sua funzione. La distinzione rispetto all'altra variabile binaria è necessaria in quanto non possono essere scambiate tra di loro (come per gli altri operatori binari). In seguito saranno illustrati alcuni esempi del suo uso.

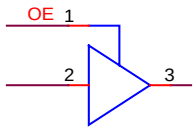


Fig. 5.8

La Fig. 5.7b mostra lo stesso circuito di Fig. 5.7a con la modifica del transistor, che ora è un canale p invece di n. I problemi discussi precedentemente, circa la indeterminazione dello stato logico di uscita si ripresentano ora quando C= 1 e si risolvono, quando necessario, ancora per mezzo di una resistenza di pull-up o di pull down come già mostrato ed indicato nella stessa figura. In questo caso, come si ricava facilmente, la tabella di verità è quella di Tab. 5.9.

C	f	I° F	II° F	III° F
0	0	0	0	0
0	1	1	1	1
1	0	hz	0	1
1	1	hz	0	1

Tab. 5.9

Allo scopo di utilizzare la sola tabella di verità, Tab. 5.8, si preferisce pensare che la configurazione resta quella precedente e che tra il suo ingresso di controllo e l'uscita venga interposta una NOT. Con questa modifica la rappresentazione grafica di Fig. 5.8 si modifica in quella di

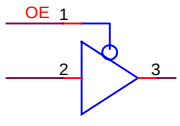


Fig. 5.9

Fig. 5.9. In questa il cerchietto all'ingresso di OE sta a significare che l'uscita del buffer è abilitata quando la variabile di controllo OE= 0 (cioè il complemento del caso precedente). Questa è una simbologia adottata, in generale, per tutte le variabili di controllo: *se nella rappresentazione simbolica c'è un cerchietto significa che il controllo è attivo quando la variabile assume il valore logico 0* .

Una ulteriore modifica ai circuiti presentati nelle Figg. 5.8 e 5.9 è realizzata introducendo, subito dopo la variabile di ingresso, una NOT (vedi Fig. 5.10 in cui il transistor CMOS è rappresentato con un interruttore). In tal modo l'uscita del buffer tri-state, quando è abilitata, fornisce il complemento della variabile di ingresso. Ci si riferisce a tale circuito come *buffer tri-state invertente*. la Fig. 5.11 ne fornisce i simboli grafici per i due diversi controlli. Notare che il valore dello stato controllato è sempre determinato dalle resistenze di pull-up e pull-down nel modo in cui è stato sopra spiegato e che nei buffer tri-state le resistenze non sono mai incluse. Sarà cura dell'utilizzatore trovarne sia il valore opportuno (che dipende sia dalla famiglia logica che dall'uso che se ne fa) che effettuare il collegamento.

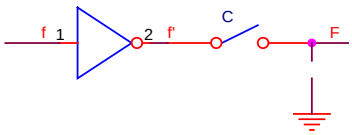
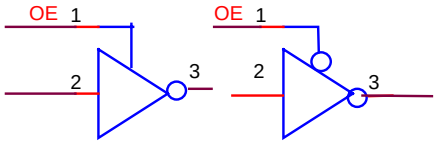


Fig. 5.10



(a) Fig. 5.11 Buffer 3state inv. (b)

5.4 Applicazione dei Buffer tri-state: Multiplexer

Si consideri il circuito di Fig. 5.12. In esso sono utilizzati due buffer-tri state non invertenti, uno con il controllo normale e l'altro con il controllo invertito. Notare che le due uscite sono collegate direttamente o, come si dice in gergo, sono in corto circuito. *In tutti gli altri casi* collegare in corto circuito le uscite è un errore che viene punito con la rottura dei componenti usati. Infatti se cortocircuitiamo le uscite di due porte logiche qualsiasi, succede che lo stato dell'uscita resta determinato solo se entrambe hanno lo stesso valore logico, ma se uno ha il valore 0 e l'altro il valore 1 quale sarà l'uscita? Quello che normalmente accade è che la porta che ha lo stato a tensione più elevata tenta di mantenerlo e quindi fornisce corrente all'uscita dell'altra porta. Poiché questo è un modo di funzionare anomalo e non previsto succede che, quasi subito, i circuiti di uscita di entrambe le porte si guastano.

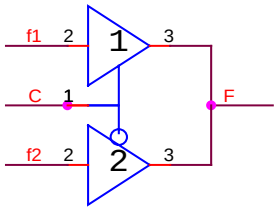


Fig. 5.12 MPX a 2 Inp.

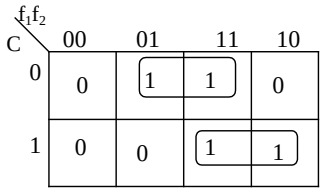
Nel nostro caso ciò non succede perché quando la variabile di controllo:

- C = 0, viene abilitata l'uscita del buffer (2) e viene posto in tri-state quella di (1). Il fatto che l'impedenza tra il terminale di uscita di quest'ultimo e massa è molto grande (al

limite infinita) significa che l'uscita del buffer (1) non richiede corrente a quella del buffer (2).

- C = 1, viene abilitata l'uscita del buffer (1) e viene posto in tri-state quella del buffer (2). Per lo stesso motivo non succede alcun guasto.

La funzione realizzata con questo circuito è che: l'uscita F = f_1 quando C = 1 mentre F = f_2 quando C = 0. In altre parole il ricettore della variabile Booleana, F si trova collegato o a f_1 o a f_2, in funzione del valore di C. Questa operazione per cui su una stessa linea possono essere trasmesse più uscite diverse (una alla volta), controllate da una o più variabili (tante quanto ne servono) si chiama operazione di *multiplexing*: cioè una operazione per cui *un grande numero di informazioni* (nel nostro caso 2) *vengono trasmesse su un numero più piccolo di canali o fili* (nel nostro caso 1).



F= $Cf_1+C'f_2$

Fig. 5.13

La mappa di questa funzione è riportata in Fig. 5.13 e la funzione minimizzata è: F=Cf_1 + C'f_2. Una realizzazione con porte AND e OR è data in Fig. 5.14. Dal confronto del circuito qui disegnato con quello di Fig. 5.12 si nota, innanzi tutto, a parte la maggiore complessità del primo, che, in ogni caso, l'operazione logica è una OR tra f_1 e f_2, con la scelta delegata allo stato della variabile C. La funzione OR, realizzata con i buffer tri-state prende il nome di *Wired-Or*. Con ciò si indica che la funzione OR viene realizzata collegando i fili (wire in inglese) in cortocircuito, invece che con l'uso di una porta logica.

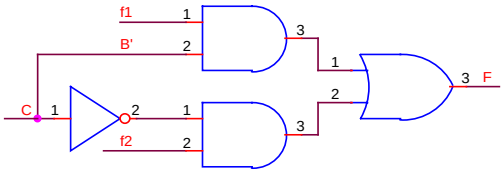
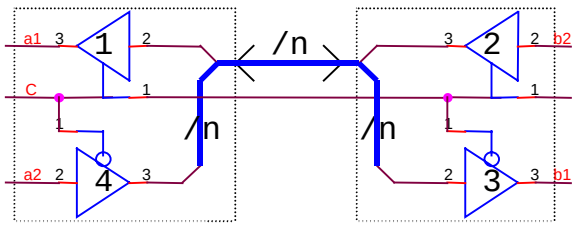


Fig. 5.14 Multiplexer a 2 ingressi

5.5 Bus Unidirezionali e Bidirezionali

Per BUS si intende un insieme di fili che trasferisce un insieme di segnali logici che, nel loro complesso, rappresentano un numero finito di informazioni. Normalmente il segnale va in una sola direzione. Questo perché le porte logiche utilizzate sono esse stesse unidirezionali, cioè ci sono dei terminali di ingresso su cui il segnale, che codifica l'informazione logica, può solo arrivare e ci sono dei terminali di uscita da cui il segnale può solo partire. Per ciò si parla di ***bus unidirezionale***. Quando il numero di fili per trasferire tutte le informazioni logiche è molto grande, per cui per es. il numero di piedini in un package non è sufficiente, si possono utilizzare i buffer Tri-State per far si che lo



a Fig. 5.15 b

stesso filo possa servire, *in tempi diversi*, sia per l'ingresso e sia per l'uscita dei segnali; si realizza così un ***bus bidirezionale*** che dimezza la necessità del numero di piedini di ingresso/uscita dell' integrato. Il circuito mostrato nella Fig. 5.15 mostra come ciò è realizzato. Racchiusi, in un rettangolo tratteggiato, nella parte (a) ci sono disegnati i circuiti di interfaccia dell'apparato (a) che riceve i segnali di ingresso attraverso il filo indicato con *a1* e trasmette i segnali di uscita attraverso il filo indicato con *a2*. Analogamente l'apparato (b) riceve i segnali attraverso il filo *b1* e invia i suoi segnali attraverso il filo *b2*. Nota che i fili *a1*, *a2*, *b1* e *b2* possono far parte di un bus unidirezionale contenente molti fili, su *ogni uno* dei quali si opera come qui di seguito descritto. I segnali di tutte e quattro i bus, prima di essere ricevuti o inviati, debbono passare attraverso dei buffer tri-state. All'esterno degli apparati (a) e (b), a ciascuno dei segnali del bus, è assegnato soltanto un filo sia per effettuare il trasferimento di ingresso sia per quello di uscita.
Quando l'informazione va da (b) ad (a) debbono essere abilitati i buffer contrassegnati con 1 e 2 (e suoi analoghi per tutti gli altri fili del relativo bus), quando l'informazione va da (a) ad (b) debbono essere abilitati i buffer contrassegnati con 3 e 4. Tale abilitazione è demandata al segnale di controllo (ne serve solo uno per tutti i buffer tri-state presenti); se C= 1 sono abilitati i buffer 1 e 2 (ed analoghi) se C= 0 sono abilitati i buffer 3 e 4 (ed analoghi)

Per semplicità, i diagrammi di circuiti contenenti bus sia unidirezionali sia bidirezionali, sono rappresentati invece che con tutti i fili, con uno soltanto, normalmente più spesso, con una freccia si segna il senso di trasmissione della informazione. Il numero di fili contenuto nel bus è indicato con una /, che attraversa la linea che rappresenta il bus, accompagnata dall'indicazione del numero di fili.

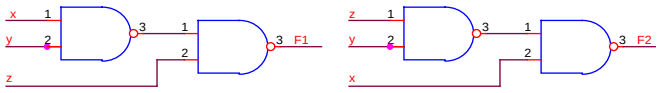
5.6 Operatori con più di due variabili di ingresso

Finora abbiamo utilizzato operatori binari ma è interessante chiederci se è possibile utilizzare operatori che hanno un numero di ingressi maggiore di 2. Diciamo subito che perché un tale operatore possa essere definito ed utilizzato, come operatore di un'algebra Booleana a due valori, bisogna che soddisfi a tutti gli assiomi dell'algebra, ed in particolare la legge commutativia, e quella associativa. Solo così noi potremo disinteressarci dell'ordine delle variabili di ingresso.
Sappiamo già che, per gli operatori binari AND e OR, vale la legge associativa. Di questi ci siamo serviti per definire l'algebra Booleana a due valori. Pertanto è definibile una AND e una OR a più di 2 ingressi ed in realtà le abbiamo già usate.
Gli operatori binari NAND e NOR ***non*** *godono della proprietà associativa*.

a) Operatore NAND: $[(x \cdot y)' \cdot z]' \neq [x \cdot (y \cdot z)']'$, infatti:

F1 = [(x•y)'•z]' = (x•y) + z' = xy + z' F2 = [x•(y•z)']' = [x'+ (y•z)] = x' + yz

Lo schema logico delle due funzione è rappresentato in Fig. 5.16. Il risultato ci dice che se commutiamo gli ingressi a tale operatore i risultati sono diversi. Notare che, se prima eseguiamo l'operazione AND a più ingressi e poi eseguiamo (sul risultato) l'operazione NOT, la commutatività è assicurata dalle proprietà dello stesso operatore AND. Concludiamo quindi che è possibile realizzare porte logiche NAND con più di 2 ingressi se la stessa rispecchia la sequenza di operazioni ora descritta cioè (x•y•z•...)'



(a) Fig. 5.16 (b)

b) Operatore NOR: [(x + y)' + z]' ≠ [x +(y + z)']'. Questa affermazione è vera per dualità. Pertanto l'operazione NOR a più ingressi deve essere realizzata da un operatore OR a più ingressi seguito da una NOT cioè:(x + y + z+...)'

5.6.1 Exculsive OR ed Equivalence

Gli operatori binari Exclusive-Or (XOR) e Equivalence (EQ) *godono della proprietà associativa*:

$x \oplus (y \oplus z) = (x \oplus y) \oplus z = x \oplus y \oplus z$ $x \odot (y \odot z) = (x \odot y) \odot z = x \odot y \odot z$

Infatti: ricordando che

$$XOR(x, y) = x \oplus y = x'y + xy',$$
$$EQ(x, y) = x \odot y = x'y' + xy$$

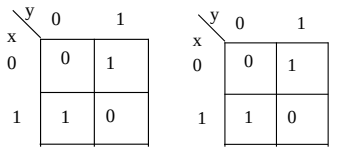
e che XOR e EQ sono uno l'operatore complementare dell'altro, abbiamo per l'XOR:

$$x \oplus (y \oplus z) = x'(y \oplus z) + x(y \oplus z)' = x'(y \oplus z) + x\,(y \odot z) =$$
$$x'(yz' + y'z) + x\,(yz + y'z') = x'yz' + x'y'z + xyz + xy'z' =$$
$$z'\,(x'y + xy') + z(xy + x'y') = z \oplus (x \oplus y) = (x \oplus y) \oplus z$$

La dimostrazione della associatività per EQ, si trova in modo simile, ed è lasciata per esercizio.

Possiamo concludere che è possibile avere operatori XOR ed EQ a più ingressi e quindi le relative porte logiche. In realtà ciò non è realizzato per complicazioni costruttive. Infatti sia l'XOR che l'EQ a due ingressi sono realizzati a partire da gate AND e OR o NOR per cui quelli a più ingressi sarebbero troppo complicate e quindi tanto costose da rinunciare a costruirle per un uso intensivo come avviene per le altre porte logiche.

Es. 5 *Realizzare una funzione Booleana F_d (x,y) di 2 variabili tale che è F_d = 1 quando il numero di 1 presenti nei minterm delle 2 variabili è dispari ed una funzione*



$F_d = x \oplus y$ $F_p = x \odot y$

Fig. 5.17 (a)

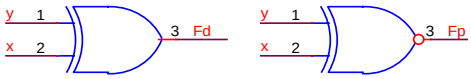
Booleana F_p (x,y) di 2 variabili tale che è F_p = 1 quando il numero di 1 presenti nei minterm delle 2 variabili è pari (nessun 1 equivale a numero pari di 1).

La Fig. 5.17a mostra la mappa per F_d (x,y) e quella per F_p (x,y). Nessuna semplificazione è possibile in entrambi i casi per cui si ha:

$$F_d = xy' + x'y = x \oplus y$$

$$F_p = xy + x'y' = x \odot y$$

La Fig. 5.17 (c e d) ne mostra il circuito logico.

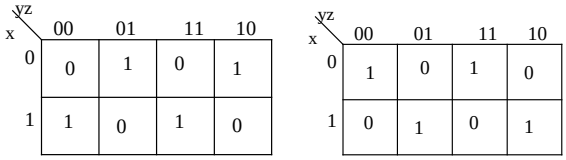


(b) Fig. 5.17 (c)

Es. 6 *Realizzare una funzione Booleana(x,y,z) di 3 variabili tale che è F_d = 1 quando il numero di 1 presenti nei minterm delle 3 variabili è dispari ed una funzione Booleana F_p (x,y,z) di 3 variabili tale che è F_p = 1 quando il numero di 1 presenti nei minterm delle 3 variabili è pari (nessun 1 equivale a numero pari di 1).*

La Fig. 5.18 mostra la mappa per F_d(x,y,z) e quella per F_p (x,y,z). Nessuna semplificazione è possibile in entrambi i casi ma, per scrivere la funzione Boolena ci si può aiutare con i risultati dell'Es. 5. Per fissare le idee consideriamo la mappa per F_d(x,y,z). Notiamo che questa può essere considerata come costituita di due parti:

1) in cui la variabile x = 0 e coinvolge la prima riga. A parte il valore di x, questa è una mappa di due variabili y e z ed ha il valore 1 nei quadratini con un numero di 1 *dispari* nei minterm con 2 variabili per cui l'espressione x'(y ⊕ z) li copre entrambi.



$F_d = x \odot y \odot z = x \oplus y \oplus z$ $F_p = x \odot y \oplus z = x \oplus y \odot z$

Fig. 5.18

2) in cui la variabile x = 1 che coinvolge le seconda. A parte il valore di x, questa è una mappa di due variabili y e z ed ha il valore 1 nei quadratini con un numero di 1 *pari* per cui l'espressione x(y ⊙ z) li copre entrambi.

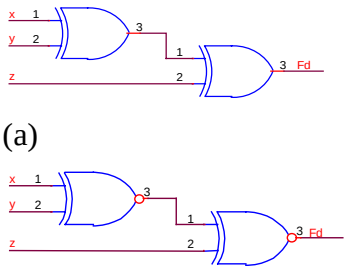
Per coprire i 4 termini facciamo la somma logica delle due espressioni ora trovate. Per cui:

$$F_d = x'(y \oplus z) + x(y \odot z) = x'(y \odot z)' + x(y \odot z) = x \odot y \odot z$$

L'ultima espressione è priva di parentesi poiché vale la proprietà associativa per l'operatore Equivalence. Nota che operando in modo leggermente diverso si ottiene la seguente:

$F_d = x'(y \oplus z) + x(y \odot z) = x'(y \oplus z) + x(y \oplus z)' = x \oplus y \oplus z$

La Fig. 5.19 mostra come queste due espressioni possono essere realizzate nella parte (a) con XOR e nella parte (b) con porte EQ.



(a)

(b) Fig. 5.19

Con procedimento analogo dalla Tab. 5.18 per F_p(x,y,z) otteniamo:

$$F_p = x'(y \odot z) + x(y \oplus z) = x'(y \odot z) + x(y \odot z)' = x \oplus (y \odot z) = x \oplus y \odot z$$

oppure

$$F_p = x'(y \odot z) + x(y \oplus z) = x'(y \oplus z)' + x(y \oplus z) = x \odot (y \oplus z) = x \odot y \oplus z$$

Il che dimostra che gli operatori XOR ed EQ commutano tra di loro. La Fig. 5.20 mostra come queste due espressioni possono essere realizzate

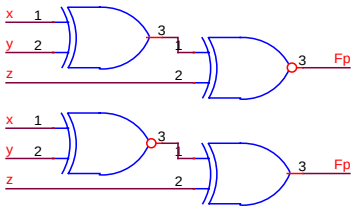


Fig. 5.20

Notare che $F_p = F'_d$ in quanto le mappe di una funzione ha zeri dove l'altra ha uni. Pertanto otteniamo l'altro risultato:

$$x \oplus y \odot z = x \odot y \oplus z = (x \oplus y \oplus z)' = (x \odot y \odot z)'$$

Es. 7 *Realizzare una funzione Booleana F_d (w,x,y,z) di 4 variabili tale che F_d = 1 quando il numero di 1 presenti nei minterm delle 4 variabili è dispari ed una funzione Booleana F_p (w,x,y,z) di 4 variabili tale che F_p= 1 quando il numero di 1 presenti nei minterm delle 4 variabili è pari (nessun 1 equivale a numero pari di 1).*

La Fig. 5.21 mostra la mappa per F_d(w,x,y,z) e quella per F_p (w,x,y,z). Nessuna semplificazione è possibile in entrambi i casi ma, per scrivere la funzione Boolena ci si può aiutare con i risultati dell'Es. 6. Per fissare le idee consideriamo la mappa per F_d(w,x,y,z).

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

$F_d = w \oplus x \oplus y \oplus z$

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

$F_p = w \odot x \odot y \odot z$

Fig. 5.21

Notiamo che questa può essere considerata come costituita di due parti:

1) in cui la variabile w = 0 e coinvolge le prime due righe. A parte il valore di w, questa è una mappa di 3 variabili x, y e z ed ha il valore 1 nei quadratini con un numero di 1 *dispari* nei minterm con 3 variabili per cui l'espressione $w'(x \odot y \odot z) = w'(x \oplus y \oplus z)$ li copre tutte e 4

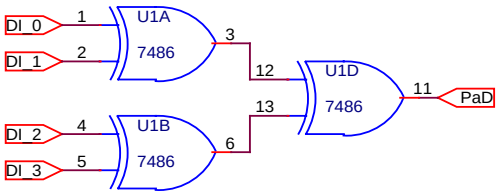


Fig. 5.22a Funzione Dispari di 4 Variabili

2) in cui la variabile w = 1 e coinvolge le righe terza e quarta. A parte il valore di w, questa è una mappa di 3 variabili x, y e z ed ha il valore 1 nei quadratini con un numero di 1 *pari* nei minterm con 3 variabili per cui l'espressione:
$w(x \odot y \oplus z) = w(x \oplus y \odot z)$
li copre tutte e 4

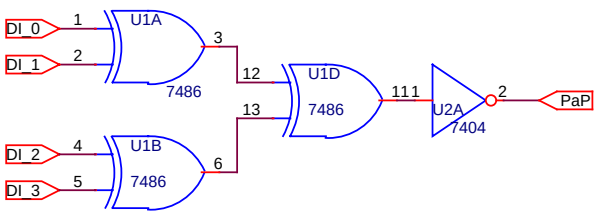


Fig. 5.22b Funzione Pari di 4 Variabili

Per coprire gli 8 termini possiamo fare la

somma logica delle due espressioni ora trovate. Per cui:

$F_d = w'(x \oplus y \oplus z) + w(x \odot y \oplus z) = w'(x \oplus y \oplus z) + w(x \oplus y \oplus z)' = w \oplus x \oplus y \oplus z$

Con procedimento analogo, dalla Fig. 5.21 per $F_p(w,x,y,z)$ otteniamo:

$F_p = w'(x \odot y \oplus z) + w(x \odot y \odot z) = w'(x \odot y \odot z)' + w(x \odot y \odot z) = w \odot x \odot y \odot z$

La Fig. 5.22a e la Fig. 522b mostrano una realizzazione pratica di Fd(4) e Fp(4) rispettivamente. Notare che la Fp(4) è stata realizzata aggiungendo un inverter all'uscita di Fd(4) (ciò è lecito sapendo che le Fp sono sempre il complemento di funzioni dispari e viceversa). La stessa realizzazione può essere interpretata dicendo che la funzione Fp(4) è stata realizzata con due XOR ed una EQ essendo (XOR)'=EQ. Notare che con l'uso di un EQ il tempo di propagazione sarebbe stato leggermente inferiore.

Indicando con $F_p(n)$ e $F_d(n)$ le funzioni Booleane che sono formate da tutti i minterm di n variabili che contegono, rispettivamente, un numero pari ed un numero dispari di 1 troviamo le seguenti formule ricorsive valide per $n > 2$:

$$F_d(n) = A' F_d(n-1) + A F_p(n-1) = A' F'_p(n-1) + A F_p(n-1) = A \odot F_p(n-1) = A' F_d(n-1) + A F'_d(n-1) = A \oplus F_d(n-1)$$

$$F_p(n) = A' F_p(n-1) + A F_d(n-1) = A' F'_d(n-1) + A F_d(n-1) = A \odot F_d(n-1) = A' F_p(n-1) + A F'_p(n-1) = A \oplus F_p(n-1)$$

Da questi risultati è facile vedere che:

1.a) *F_d (2n) può essere scritta con l'XOR di tutte le variabili.*

$F_d(2n) = A \oplus F_d(2n-1) = A \oplus B \oplus F_d(2n-2)$

1.b) *F_p (2n) può essere scritta con l'EQ di tutte le variabili.*

$F_p(2n) = A \odot F_d(2n-1) = A \odot B \odot F_p(2n-2)$

2.a) *F_d (2n+1) può essere scritta con l'XOR o l'EQ di tutte le variabili.*

$F_d(2n+1) = A \oplus F_d(2n) = A \odot F_p(2n)$

2.b) *F_p(2n+1) può essere scritta con l'EQ di tutte le variabili meno 1 combinata in XOR con la variabile mancante o con l'XOR di tutte le variabili meno 1 combinata in EQ con la variabile mancante.*

$F_p(2n+1) = A \oplus F_p(2n) = A \odot F_d(2n)$

Le funzioni XOR ed EQ sono molto utili in sistemi che richiedono rivelazione o correzione di errori. Come già discusso nel Par. 1.3.2 l'introduzione di un bit di parità è un modo per rivelare errori di trasmissione di informazioni binarie. Viene incluso nel messaggio un bit in più in modo da rendere (con il suo valore a 0 o ad 1) gli 1 del messaggio totale o pari o dispari. Il messaggio, incluso il bit di parità, è trasmesso e quindi, all'estremità ricevente, avviene il controllo se si sono verificati errori. Viene rivelato un errore se la parità trovata non corrisponde a quella trasmessa. Il circuito che genera il bit di parità all'estremità trasmittente si chiama *parity generator* (generatore di parità); il circuito che controlla la parità all'estremità ricevente si chiama *parity checker* (controllore di parità).

Per es. consideriamo un messaggio da trasmettere costituito da 3 bit più un bit di parità dispari. Il messaggio consiste di 3 variabili binarie e dobbiamo realizzare una funzione che assume il valore 1 se il numero di bit nel messaggio è pari. Per la regola (2.b), chiamati x, y, z le 3 variabili e P il bit di parità, abbiamo P = x ⊙ y ⊕ z. Lo schema di questo circuito è uno dei due rappresentati in Fig. 5.20 oppure può essere ricavato dal circuito di Fig. 5.22a, ponendo il quarto bit ad 1 o dalla Fig. 522b ponendo il quarto bit a zero.

Il messaggio, costituito dai 3 bit più il bit di parità, viene trasmesso ed all'estremità ricevente viene sottoposto al controllore di parità. Il messaggio consiste di 4 variabili binarie che deve contenere un numero dispari di 1. Ma perché ci sia un errore dobbiamo rivelare un numero pari di 1. Per la regola (1.b), chiamati x, y, z le variabili, P il bit di parità e C il risultato del controllo abbiamo: C = P ⊙ x ⊙ y ⊙ z. Lo schema di questo circuito è quello di Fig. 5.22.b.

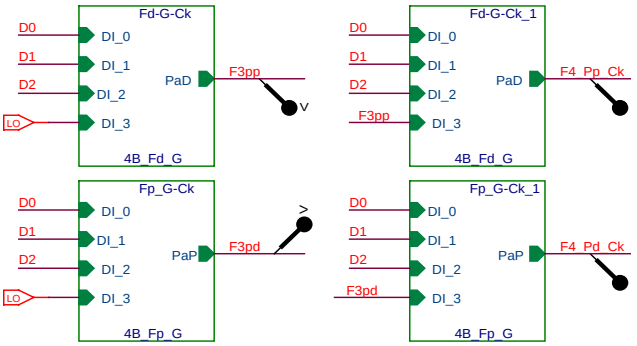


Fig. .5.23

La Fig.5.23 mostra, a sinistra, i circuiti di Fig. 5.22a e 5.22b utilizzati rispettivamente come generatore di parità *dispari* e *pari* di un messaggio formato da 3 bit (il quarto bit è posto a zero) e quindi, nella parte destra, gli stessi circuiti usati come rivelatore di errore di parità, rispettivamente dispari e pari. Il timing che rappresenta il loro funzionamento è dato in Fig. 5.24.

Si noti che entrambi i circuiti di rivelazione di errore mostrano livelli pari a zero con la presenza di spikes in corrispondenza alle transizioni delle forme d'onda relative al generatore

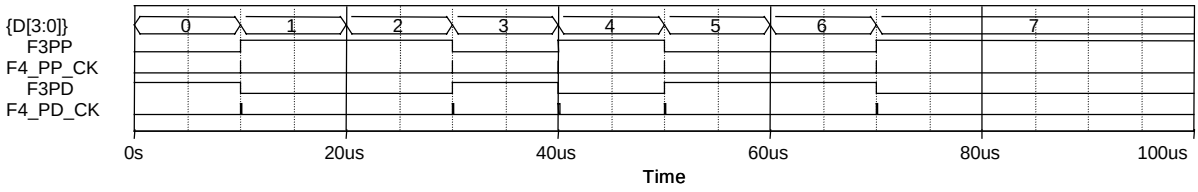


Fig. 5.24

di parità. Ciò è dovuto al fatto che il bit di parità arriva in leggero ritardo rispetto a quelli di ingresso, che sono usati anche per gli ingressi del circuito rivelatore di errore. A causa del fatto che il bit di parità pari è realizzato con un tempo di propagazione maggiore di quello dispari gli impulsi spuri durano più a lungo. In generale, quando la presenza di impulsi spuri nella forma d'onda, potrebbe danneggiare il funzionamento dei circuiti in cascata e la loro durata massima è nota, la forma d'onda non è immediatamente utilizzata ma viene fatta passare attraverso una porta logica AND che è in stato controllato per tutto il tempo della durata dei glitch e abilitata subito dopo.

5.6.2 Simulazione di generatori/checker di parità

La Fig. 5.25 mostra i circuiti per la simulazione delle funzioni che generano o testano la parità. Notare che le EQ (74LS266) hanno, in uscita, un resistore di pull-up in quanto essi sono a collettore aperto. Sono stati usati due differenti packages di XOR e cioè il 74LS86A e il 74LS386A. Dal timing della Fig. 5.26 notare che, quando il numero delle variabili di ingresso è pari, l'uso di tutte XOR produce funzioni dispari (genera un bit di parità pari o genera un bit di errore per il test di parità pari), mentre l'uso di tutte EQ produce funzioni pari (genera un bit di parità dispari o genera un bit di errore per il test di parità dispari)

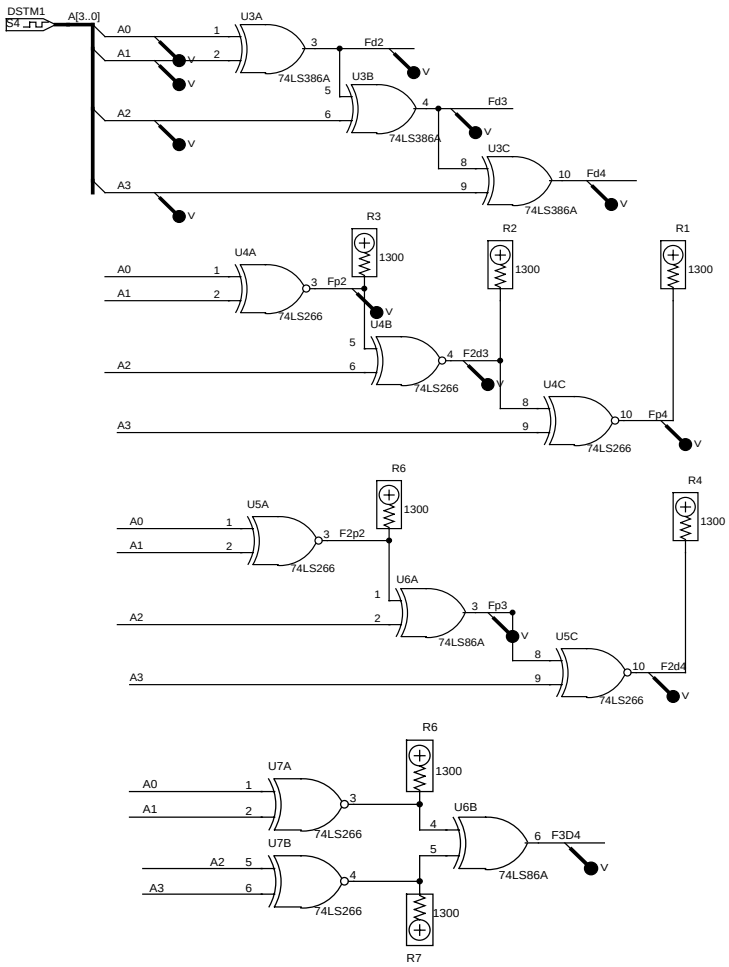


Fig. 5.25

Quando il numero di bit di

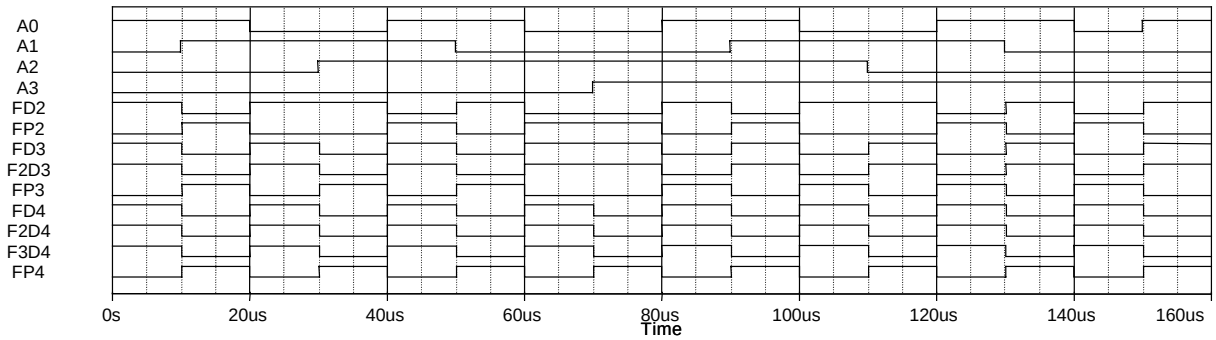


Fig 5.26

ingresso è dispari l'uso di tutte XOR o di tutte le EQ produce una funzione dispari. Per ottenere una funzione pari di un numero dispari di variabili è necessario usare almeno una XOR e le altre porte EQ o viceversa una porta EQ e le altre XOR. Il confronto tra le traccie per F2D4 e F3D4 dimostrano la legge associativa dell'EQ e dell'XOR

5.7 Circuiti di interfaccia

Del problema generale dell'interfaccia tra apparecchiature di diverso tipo si è già parlato nel Par. 1.1. Ci interesseremo ora, in particolare, del problema di interfaccia in cui è coinvolta una sola linea. Un esempio è l'adattamento dei livelli logici tra le diverse famiglie o dell'uscita analogica che deve pilotare un ingresso digitale per es. una porta logica. Introdurremo inizialmente i principi di funzionamento di due circuiti analogici: *il circuito integratore ed il circuito discriminatore*. Nel seguito supporremo che sia le funzioni di ingresso che di uscita siano delle forme d'onda di tensione.

5.7.1 Circuito Integratore

Il circuito integratore, a cui siamo interessati, si comporta come tale soltanto nelle vicinanza delle transizioni dei segnali di ingresso. Motivi pratici, come il valore della tensione di alimentazione, limitano i valori minimi e massimi che le tensioni di uscita possono assumere. Quando il valore di tensione, che la funzione integrale dovrebbe assumere, esce da questi limiti l'uscita di un integratore assume un valore di tensione costante: pari al massimo, V_s, o al minimo 0 Volt. Un integratore in queste condizioni si dice essere in condizione di *saturazione*.

Supponiamo che la tensione di ingresso sia un segnale logico che per $t < t_0$ $V_i = 0$ (0 logico), che per $t_0 < t < t_1$ $V_i = V_0$ (1 logico) e che per $t_1 < t$ $V_i = 0$ (0 logico). Al tempo t_0 si ha una transizione positiva. L'uscita del circuito integratore tende al valore di saturazione V_s con legge lineare $V_{out} = \alpha\ t$, come mostrato in Fig. 5.27 (l'asse orizzontale è quello dei tempi). Quando il valore della tensione di uscita raggiunge il valore di saturazione allora $V_{out} = V_s$ = costante. Chiamiamo t'_0 il tempo in cui tale valore viene raggiunto ($t'_0 = V_s/\alpha$). Per $t > t_1$ la nostra tensione V_i passa dal valore V_0 a 0 Volt. Allora V_{out} tende al valore inferiore di saturazione con legge lineare: $V_{out} = V_s - \alpha\ t$.

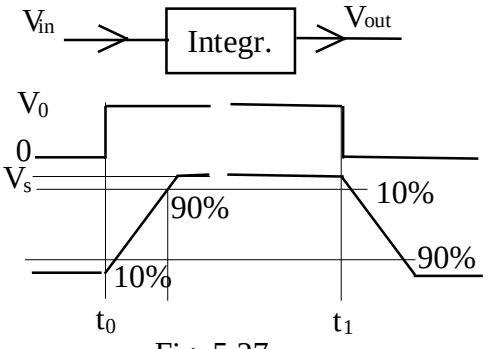


Fig. 5.27

La forma d'onda che ne risulta presenta delle transizioni simmetriche verso i valori di saturazione. Si definisce *tempo di salita*, t_s, il tempo necessario a che una forma d'onda passi dal 10% al 90% del suo valore massimo e *tempo di discesa*, t_d, il tempo necessario a che una forma d'onda passi dal 90% al 10% del suo valore massimo. Questa definizione permette di trattare anche forme d'onda con transizioni non simmetriche cioè con $t_s \neq t_d$, ed eventualmente con legge diversa della lineare (le transizioni reali delle forme d'onda seguono, in generale, una legge esponenziale).

$$V_{out} = V_0\left(1 - e^{-t/\tau s}\right) \qquad \text{Fronte di salita}$$

$$V_{out} = V_0 e^{-t/\tau d} \qquad \text{Fronte di discesa}$$

Dove τ_s e τ_d prendono il nome di *costante di tempo di salita e di discesa* rispettivamente. Nel seguito per semplicità supporremo transizioni lineari e simmetriche.
Nel Par. 2.3.2 abbiamo detto che i segnali logici non possono assumere valori tra quelli che corrispondono allo 0 ed all'1 logico, ma possono soltanto effettuare delle transizioni tra i due livelli. Il tempo in cui queste transizioni debbono verificarsi deve essere inferiore ad un certo massimo che dipende dalla famiglia logica. Se il tempo di salita del segnale di ingresso ad una porta logica supera tali limiti si ha che, durante il tempo di salita, la tensione di uscita della porta può presentare delle oscillazioni tra i due livelli logici. Essendo questa una condizione indesiderata bisogna evitare che succeda. Questo problema di interfaccia può essere parzialmente risolto con un circuito discriminatore.

5.7.2 Circuito Discriminatore e Generatori di Ritardo

Un circuito Discriminatore è il primo passo verso il superamento del problema presentato nel precedente paragrafo. Esso, per quanto ci interessa, è essenzialmente un circuito che accetta, in ingresso, un segnale analogico e fornisce, in uscita, un segnale logico. Le transizioni tra i due livelli logici hanno tempi di salita e discesa molto piccoli. Dal punto di vista funzionale presenta due ingressi analogici ed una uscita logica.

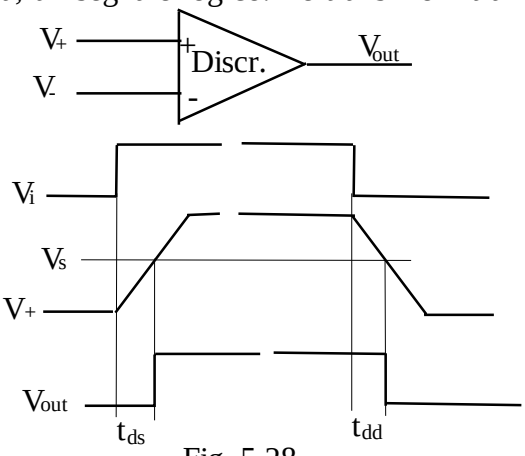


Fig. 5.28

I due ingressi sono marcati una con il segno + e l'altro con il segno -. La scelta di tale convenzione simbolica è giustificata dal suo funzionamento: se la tensione all'ingresso + supera quella dell'ingresso - allora la tensione di uscita ha un valore di tensione corrispondente al livello logico 1, viceversa al livello logico 0. Se all'ingresso - forniamo un valore di tensione costante, V_s, chiamato *tensione di soglia*, e all'ingresso + forniamo una tensione analogica come la tensione di uscita da un circuito integratore, la tensione di uscita, V_{out}, del discriminatore, avrà l'andamento disegnato in Fig. 5.28 dove: V_i rappresenta la tensione applicata al circuito integratore, V_+ rappresenta la sua tensione di uscita e V_s rappresenta la tensione di soglia all'ingresso - del discriminatore. Dal diagramma temporale della stessa Fig. 5.28 notiamo che V_{out} è un segnale logico, con tempi di salita e discesa molto rapidi, che riproduce il segnale logico di ingresso con un ritardo temporale, t_{ds}, per il fronte di salita (è il tempo che la tensione V_+ impiega a raggiungere il valore di soglia in salita) e con un ritardo temporale, t_{dd}, per il fronte di discesa (è il tempo che la tensione V_+ impiega a raggiungere il valore di soglia in discesa). Se la tensione di soglia V_s è a metà strada, tra il livello massimo ed il livello minimo di V_+, abbiamo che: $t_{ds} = t_{dd}$.

La caratteristica ora illustrata trasforma la tensione di uscita dell'integratore ad uno idoneo a pilotare un circuito logico e può essere utilizzata per ottenere segnali logici di uscita ritardati rispetto a quelli di ingresso. Il ritardo temporale può essere programmato scegliendo sia il tempo di salita e di discesa del circuito integratore che il valore di soglia del discriminatore.
Se V_s è collegata all'ingresso + e, l'uscita del circuito integratore è collegata all'ingresso - del discriminatore, otterremo un segnale di uscita, che ha le stesse caratteristiche di commutazione, ma che è il complemento del segnale logico di ingresso.

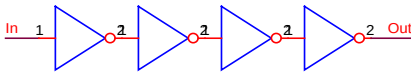


Fig. 5.29

Un altro modo per ottenere segnali logici ritardati è quello di utilizzare il tempo di propagazione delle porte logiche, in particolare quelli dei buffer o degli inverter (vedi Fig. 5.29). Se si usano un numero dispari di inverter si ottiene anche una complementazione. Valori di ritardo piccoli sono realizzati con questa tecnica, mentre ritardi più lunghi sono ottenuti per mezzo di circuiti che funzionalmente si comportano come quello illustrato in precedenza composto da un integratore e un discriminatore. Quando, in un circuito logico, vogliamo usare un circuito di ritardo useremo il simbolo di Fig. 5.30 indipendentemente dalla sua realizzazione interna.

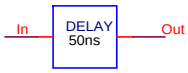


Fig.5.30

5.7.3 Trigger di Schmitt

In circuito chiamato *Trigger di Schmitt* fa parte, a tutti gli effetti, delle funzioni disponibili nelle varie famiglie logiche. Esso è realizzato a partire dal circuito discriminatore già illustrato con l'aggiunta di un collegamento di reazione. Noi non ci addentreremo nei dettagli realizzativi ma ci accontenteremo di illustrare il suo funzionamento. Esso è funzionalmente caratterizzato da:

1) Un solo ingresso in cui accetta segnali analogici, entro i valori di alimentazione della famiglia logica a cui appartiene. L'ingresso per la tensione di soglia non è disponibile.
2) Esistono due valori di soglia (generate al suo interno) V_{T+} e V_{T-} con $V_{T+} > V_{T-}$. La differenza $V_{T+} - V_{T-}$ prende il nome di *tensione di isteresi*. Lo stato logico dell'uscita determina quale soglia è attiva. Le tensioni di soglia ora definite, in generale, coincidono con i valori di tensione inferiore e superiore che definiscono l'intervallo di tensione di ingresso "proibito" per la famiglia logica a cui appartiene il discriminatore. La Fig. 5.31 presenta sia il simbolo logico che la caratteristica di funzionamento. In ascisse ed in ordinata sono riportate rispettivamente

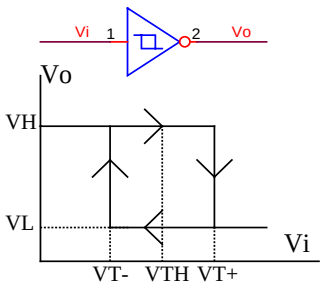


Fig. 5.31

la tensione di ingresso, V_i, e la tensione di uscita, V_o. Le frecce servono per risolvere le ambiguità sul valore assunto da V_o per V_i compreso tra le tensioni di *soglia inferiore* = V_{T-}, e *soglia superiore* = V_{T+}.

Per meglio capire il grafico bisogna pensare di percorrerlo seguendo le frecce notando, contemporaneamente, i valori di V_o e V_i. Partiamo dal punto $V_i = 0$ e $V_o = V_H$. Seguendo le frecce vediamo che: fino a quando V_i risulta inferiore a V_{T+} la tensione di uscita è al livello alto $V_o = V_H$. Non appena si supera V_{T+}, V_o precipita al valore basso = V_L e se V_i aumenta, mantiene questo valore seguendo la freccia verso destra. Quando si inverte il senso di marcia (V_i diminuisce) V_o resta a livello basso fino a che $V_i > V_{T-}$ cioè pur avendo riattraversato V_{T+}. Quando V_i diventa inferiore a V_{T-} la tensione di uscita si riporta ancora a V_H. La caratteristica che abbiamo percorsa prende il nome di *ciclo di isteresi* ed è formalmente simile alla isteresi studiata nel ferromagnetismo.

I livelli di tensione nominali di V_H e V_L sono rispettivamente i valori dell' 1 e dello 0 logico della famiglia a cui il particolare trigger di Schmitt appartiene. Se il segnale di ingresso è un segnale logico il circuito si comporta come un normale inverter. Generalmente i valori di soglia dipendono dalla tensione di alimentazione e sono tali che il loro valore medio coincide con la metà della tensione di alimentazione.

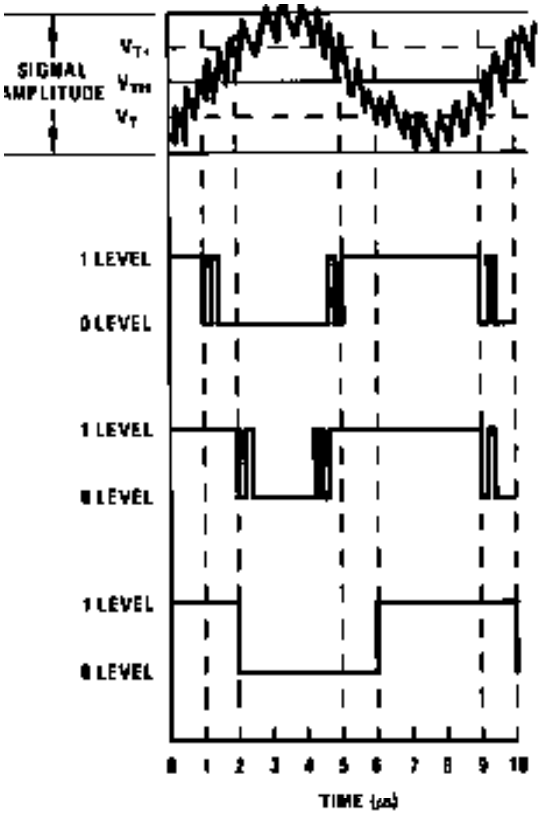


Fig. 5.32

La Fig. 5.32 (tratta dall'Application Note AN140 della Fairchild Semiconductors), presenta il comportamento di un trigger di Schmitt quando la tensione di ingresso analogica ha del rumore. Notare che l'uso di un normale discriminatore con tensione di soglia, V_{TH}, pari a metà della escursione della tensione di ingresso, avremmo attenuto più impulsi invece di uno solo, se usiamo un discriminatore con la soglia pari a V_{T+} avremmo ottenuto in uscita non solo più impulsi ma anche un impulso positivo che dura meno del negativo. Quando usiamo il trigger di Schmitt gli impulsi multipli spariscono per effetto della isteresi e l'unico impulso che si ottiene ha durata della parte positiva uguale a quella negativa e la sua differenza, rispetto a quello che si sarebbe ottenuto con un discriminatore privo di isteresi e soglia V_{TH} e nessun rumore, è solo un ritardo. Proprio per tale motivo un circuito trigger di Schmitt è preferito al normale discriminatore e costituisce il circuito di interfaccia standard tra il mondo analogico e quello digitale. Notare infine che, se il segnale di ingresso è

l'uscita dell'integratore, di cui abbiamo già parlato, l'uscita del trigger di Schmitt non differisce da quello di un normale discriminatore.

Fa parte delle famiglie logiche anche una funzione NAND a quattro ingressi. Ciascuno degli ingressi ha le stesse caratteristiche del trigger di Schmitt finora discusso ed il suo simbolo grafico è quello di Fig. 5.33.

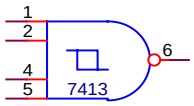


FIG. 5.33

5.7.3 Uso di circuiti Generatori di Ritardo come rivelatori di transizioni di un segnale logico periodico (Clock)

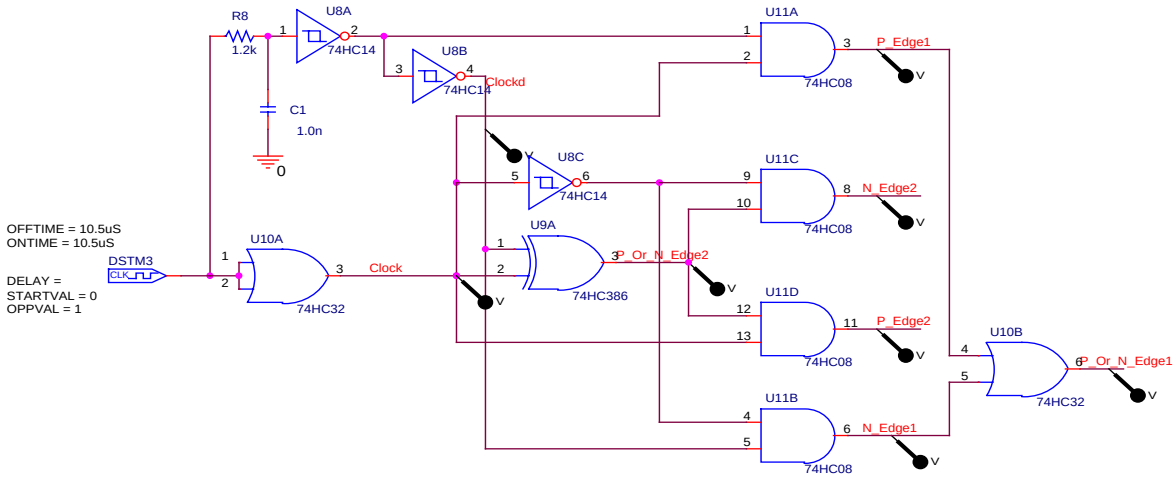


Fig. 5.34

La Fig. 5.34 mostra l'uso di un gruppo integratore RC per generare un clock ritardato ed il suo successivo uso in un rivelatore di transizione: solo positivi, solo negativi e positivi e negativi (duplicatore di frequenza), della forma d'onda di ingresso.

La coppia RC, realizza un filtro passa basso pilotato da un generatore di clock, DSTIM3. L'uscita del filtro (vedi parte inferiore analogica del timing di Fig. 5.35) va al trigger di Schmitt, U8A. Questo essenzialmente squadra la forma d'onda in ingresso, rigenera la forma d'onda del generatore, ritardata di un tempo che è proporzionale alla costante di tempo del circuito RC e al valore di soglia superiore del trigger di Schmitt. L'uscita (piedino 4) dell'inverter U8B (clockd) fornisce una forma d'onda con la stessa polarità del clock (vedi Fig. 5.35). La OR, U10A, è usata soltanto per evitare il display del clock nella parte analogica del timing.

Il resto del circuito mostra due modi diversi di realizzazione di un circuito rivelatore di fronti o solo positivi o solo negativi o sia positivi sia negativi. Il primo è realizzato con le porte U11A e U11B e U10B, il secondo con le porte U8C, U9A, U11C e U11D. Dal timing si può

notare che in corrispondenza ad ogni fronte del clock c'è un impulso la cui durata è pari al ritardo di *clockd*.

Notare che la forma d'onda *N_Edge2* presenta, in corrispondenza ai fronti di salita del clock, degli impulsi spuri messi in evidenza dalla simulazione con l'opzione "*worst-case*". Questo fatto evidenzia la possibilità che con l'uso di quel circuito possono verificarsi delle corse. Siccome nelle altre uscite non sono presenti spikes si evince che, in questo caso, l'uso delle porte AND è preferibile a quello della XOR.

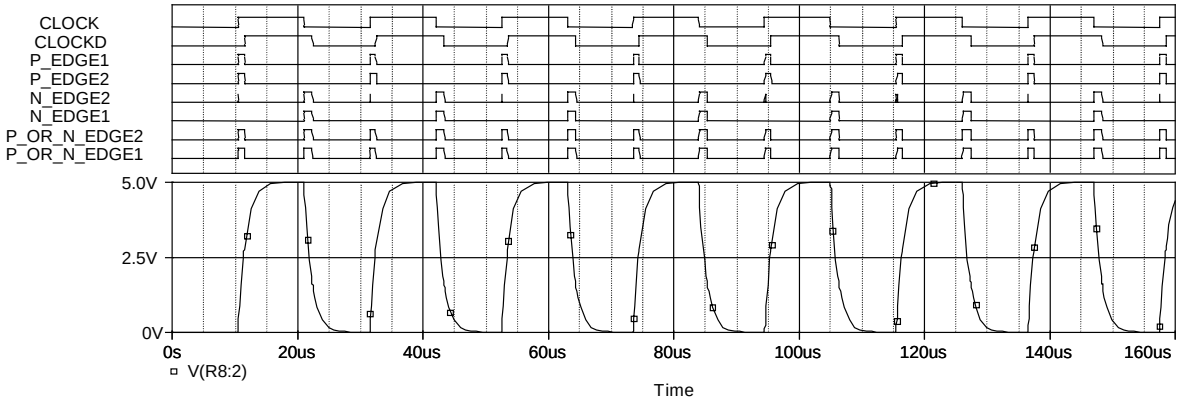


Fig. 5.35

5.7.4 Corse

Si noti che noi abbiamo realizzato questi circuiti utilizzando una unità di ritardo appositamente progettata. Gli stessi fenomeni possono avvenire quando questi ritardi sono presenti e non voluti. In tali casi quello che è qui chiamato rivelazione di fronti vengono chiamate ***spikes*** o ***glitchs*** e sono spiegati come risultato delle ***corse*** presenti tra i segnali che pervengono ai piedini di ingresso di una porta logica; uno arriva in ritardo rispetto all'altro per effetto, per es., di un livello logico, interposto su di una linea, ma non nell'altra.

PROBLEMI