Capitolo 1

Modelli di guasto

Michele Favalli

1.1 Introduzione

Come già osservato, il collaudo di un circuito digitale consiste nel verificare che il suo comportamento corrisponda a quello corretto (definito dalle specifiche) mediante l'applicazione di opportune sequenze di valori di logici (*test-patterns*) agli ingressi e l'osservazione delle sue risposte.

Per quanto riguarda il comportamento funzionale, le specifiche sono costituite dalle relazioni logiche che legano l'andamento delle uscite a quello degli ingressi (assegnate come tabella della verità nelle reti combinatorie e come tabella di transizione dello stato in quelle sequenziali). Oltre a queste specifiche funzionali vengono in generale forniti anche vincoli sul comportamento elettrico e sulle temporizzazioni dei segnali, anch'essi da verificare perchè il collaudo risulti completo. In questo capitolo, tuttavia, ci occuperemo principalmente dei problemi inerenti, le specifiche di tipo funzionale.

Da questo punto di vista, il piú immediato approccio al collaudo consiste nella verifica completa delle specifiche funzionali (collaudo esaustivo), ovvero nella verifica della correttezza di ogni stato del circuito. Nel caso venga riscontrato uno scostamento fra il valore di una o più uscite da quello corretto (*errore* sulle uscite) il circuito viene considerato come malfunzionante.

Il numero di vettori con cui è necessario stimolare il circuito per verificarne ogni stato risulta uguale a 2^n nel caso di circuiti combinatori e a 2^{n+m} nel caso di circuiti sequenziali (ove n rappresenta il numero di ingressi del circuito e m il numero di bit di memoria).

È facile verificare che con l'incremento nella complessità dei circuiti integrati, il numero di vettori necessari per il collaudo esaustivo cresce rapidamente, impli-

cando tempi (e quindi costi) di collaudo eccessivi. Ad esempio, si consideri una rete combinatoria con 64 ingressi, in questo caso il numero di vettori da applicare per collaudarla esaustivamente risulta pari a 2^{64} e supponendo di applicarli con una frequenza d1 10MHz il tempo necessario per il collaudo risulta circa pari a $2 \cdot 10^7$ giorni.

Per superare questi problemi, conviene spostare l'attenzione dal sintomo (*errore*) alle cause che possono portare a un malfunzionamento del circuito. In particolare, si assume che un *errore* sulle uscite sia sempre una conseguenza di un guasto (*fault*) interno al circuito, in modo che il collaudo debba mirare a *rivelare* la eventuale presenza di ogni guasto realisticamente ipotizzabile, invece che considerare tutti i possibili stati del circuito.

Questo approccio sotto determinate ipotesi semplificative, consente una drastica riduzione del numero dei vettori di *test* rispetto al collaudo esaustivo, riconducendo tempi e costi entro margini accettabili.

Infatti, affinchè un vettore di collaudo sia in grado di determinare se un circuito è affetto da un guasto, è necessario che la presenza di tale guasto venga rivelata come lo scostamento di un uscita dal suo valore corretto (*errore*). Quindi per collaudare in maniera completa un circuito (ovvero per verificare che questo sia privo di guasti) è in teoria necessario un numero di vettori di test almeno pari al numero di guasti considerati.

In realtà il numero di vettori necessari per il collaudo risulta, in generale, minore del numero di guasti, in quanto diffetti di diverso tipo possono avere lo stesso effetto sul comportamento del circuito. Per questo motivo, un singolo vettore può rivelare la presenza di più malfunzionamenti contemporaneamente.

Per determinare il numero di vettori necessari per collaudare un circuito in maniera completa, bisogna esaminare i possibili malfunzionamenti e difetti presenti nel circuito. In generale, il loro numero è molto grande, anche se molti di essi provocano gli stessi effetti sul comportamento del circuito.

Per descrivere tali effetti vengono introdotti dei *modelli di guasto* che devono necessariamente fare riferimento ad una specifica rappresentazione del circuito (in particolare, tra quelle descritte nel precedente capitolo).

Nei paragrafi seguenti verranno trattati diversi modelli di guasto che descrivono gli effetti di malfunzionamenti circuitali, con particolare attenzione alle tecnologie e ai circuiti tipici dei circuiti VLSI.

1.2 Modelli di guasto nei circuiti a grande scala di integrazione

Nelle metodologie di collaudo orientate alla rivelazione di guasti, per determinare se una sequenza di vettori di *test* è in grado di verificare in maniera completa il

corretto funzionamento di un circuito digitale bisogna disporre di un insieme di modelli di guasto che rappresenti in maniera il piú possibile realistica il comportamento indotto nel circuito da tutti i possibili malfunzionamenti di tipo fisico [1, 2, 3]. Questi, in generale, sono molteplici e dipendono dalle caratteristiche dei diversi processi costruttivi utilizzati nella costruzione dei circuiti, anche se è possibile identificare una serie di malfunzionamenti comuni a diverse tecnologie (bipolari, MOS, CMOS, BiCMOS).

Fra questi meritano menzione: la presenza indesiderata di materiale conduttivo (metallo, polisilicio, siliciuri ...) che connette tra loro nodi altrimenti isolati, oppure l'interruzione di una interconnessione dovuta alla mancanza di materiale conduttivo, le perforazioni dell'ossido di isolamento nei dispositivi e tra le linee di interconnessione [4, 5, 6].

I comportamenti elettrici cui questi malfunzionamenti danno luogo dipendono dalla tecnologia e dallo specifico tipo di circuito in cui il guasto è presente.

Di conseguenza, per poter rappresentare in maniera completa le possibili condizioni di malfunzionamento, dovranno essere considerati diversi modelli di guasto, dipendentemente dal meccanismo fisico che descrivono, dalla tecnologia, dall'implementazione e dalla funzionalità del circuito.

D'altra parte, la grande complessitá degli odierni circuiti digitali integrati impone di considerare modelli di guasto semplici e, naturalmente, compatibili con i simulatori (*fault simulator*) necessari per descrivere il comportamento del circuito in presenza di guasti.

Le due esigenze individuate in precedenza, in generale denominate "accuratezza" e "trattabilitá" [3] sono fra loro contrastanti; e debbono essere conciliate con una visione di tipo gerarchico del circuito, in cui a ciascun livello descrittivo (switch, gate e behavioral) vengono considerati modelli adeguati. Al momento, la ricerca in questo settore é orientata sia verso lo sviluppo di nuovi e piú accurati modelli di guasto (necessari, ad esempio, per i circuiti CMOS), sia verso nuovi algoritmi di simulazione in grado di trattare in maniera efficiente la complessitá dei circuiti sotto diverse condizioni di malfunzionamento.

L'esigenza "trattabilitá" impone ipotesi alquanto restrittive sui modelli di guasto che vengono considerati nella simulazione: ad esempio, è comune il caso in cui si ipotizza che nel circuito possa essere presente un solo guasto alla volta (ipotesi di guasti singoli) e che esso sia di tipo permanente. Queste ipotesi non sono, in generale, molto realistiche, in quanto, specie nei circuiti ad alta densitá di integrazione, é molto probabile la presenza di guasti multipli, e in casi particolari, anche di tipo intermittente. Tuttavia esse servono per contenere i tempi di simulazione entro limiti accettabili.

A questo punto, si pone il problema dell'ottenimento di un buon compromesso tra la varietà dei possibili malfunzionamenti che intende trattare (ovvero il numero dei modelli di guasto che si considerano) ed il tempo (cioè il costo) delle

simulazioni necessarie per analizzare il circuito. Ogni restrizione che si introduce per quanto riguarda il primo aspetto provoca un degrado dell'affidabilità dei risultati delle simulazioni. D'altra parte, queste devono essere ricondotte entro limiti accettabili di costo, anche in relazione alle altre fasi di sviluppo del componente.

1.2.1 Guasti del tipo *stuck-at*

Il modello di guasto, meglio conosciuto è il cosiddetto *stuck-at 0/1*, che rappresenta una linea del circuito bloccata al valore logico 0 o 1 (come conseguenza, per esempio di un corto con la massa o l'alimentazione) indipendentemente dagli stimoli applicati al circuito. Questo modello, storicamente il primo ad essere sviluppato, é tuttora il piú utilizzato e in certi casi l'unico ad essere considerato, nella maggior parte dei simulatori commerciali. Ció é dovuto alla sua semplicitá, che lo rende pienamente trattabile nei simulatori logici (in cui i segnali sono rappresentati dai valori 0 ed 1), alla sua indipendenza dal tipo di tecnologia considerato ed al fatto che rappresenta correttamente una frazione non trascurabile dei possibili malfunzionamenti circuitali anche nel caso dei circuiti VLSI (come illustrato in Fig. 1.1).

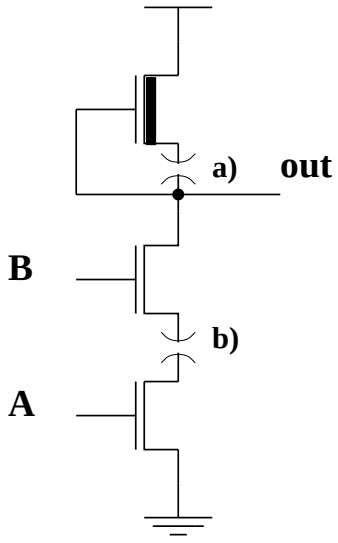


Figura 1.1: Esempio di guasti rappresentabili mediante il modello *stuck-at*: a) un interconnessione interrotta nel *pull-up* dà luogo a un guasto di tipo *stuck-at-0*; b) lo stesso difetto nel *pull-down*, invece, è causa di un guasto di tipo *stuck-at-1*.

Proprio per la sua semplicitá, il modello di tipo *stuck-at 0/1* é utilizzato per spiegare alcuni concetti che sono del tutto generali e che possono essere trasferiti agli altri modelli di guasto che tratteremo in seguito.

Come già ampiamente descritto, al fine di collaudare un circuito, una sequenza di vettori deve essere in grado di rivelare come *errori* in uscita eventuali guasti presenti nel circuito. Perché questo avvenga nel caso di un guasto del tipo *stuck-at c* (dove *c* vale *0* o *1*), la sequenza deve controllare la linea affetta dal guasto a un valore opposto a *c* (questa operazione è detta *attivazione* del guasto), inoltre il valore della linea considerata deve essere reso osservabile a un uscita del circuito (*propagazione* degli effetti del guasto). Solamente in questo caso, infatti, si avrá un valore diverso da quello che si avrebbe in assenza del guasto (ovvero un *errore*) (Fig. 1.2).

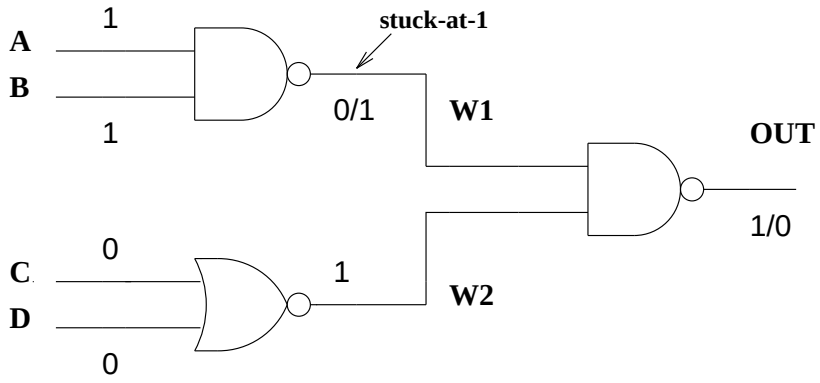


Figura 1.2: Esempio di attivazione e propagazione di un guasto. Per *attivare* lo *stuck-at-1* sulla linea **W1**, gli ingressi **A** e **B** devono valere 1 (in modo che sia $\mathbf{W1} = 0$ nel caso corretto); per propagare gli effetti del guasto all'uscita è necessario che sia $\mathbf{W2} = 0$ (e quindi $\mathbf{C} = \mathbf{D} = 0$). La notazione 1/0 (0/1) indica che la linea considerata vale 1 (0) nel circuito corretto e 0 (1) in quello guasto.

Nel caso di un circuito combinatorio, un solo vettore di collaudo é può rivelare un guasto del tipo *stuck- at 0/1*, mentre nel caso di circuiti sequenziali possono essere necessari piú vettori sia per l'attivazione che la propagazione del guasto (si veda l'esempio di Fig. 1.3).

In particolare, negli odierni circuiti VLSI é particolarmente difficile generare sequenze di *test* in grado di rivelare determinati guasti, in quanto si ha una grande complessitá interna (quantificabile in parecchie migliaia di *gate*) e, viceversa, un numero molto ridotto di pin di ingresso e uscita utililizzabili per poter controllare e propagare i guasti presenti nel circuito [1].

Come si é visto, un simulatore di guasti determina quali fra tutti quelli pos-

[1] La facilitá con cui un guasto puó essere rivelato viene in generale definita come testabilitá e dipende sia dalla sua controllabilitá (ovvero la facilitá con cui si può controllare la linea del circuito su cui é presente il guasto per permetterne l'attivazione), che dalla osservabilitá (ovvero la facilitá con cui può essere propagato).

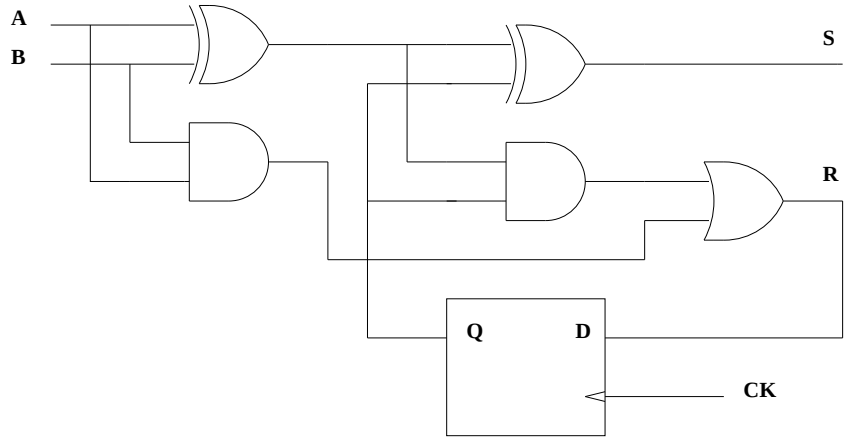


Figura 1.3: Per rivelare un guasto nel *serial-adder* illustrato *stuck-at-0* sulla linea **R** è necessario applicare prima il vettore di collaudo $\mathbf{A} = 1$, $\mathbf{B} = \mathbf{1}$ in modo da attivare il guasto e poi un secondo vettore (qualsiasi data la struttura del circuito) per rendere osservabili gli effetti all'uscita **S**.

sibili presenti all'interno di un circuito sono rivelati dai vettori di *test* applicati agli ingressi, verificando per ciascuno di essi le condizioni di attivazione e propagazione. In generale, in circuiti di grandi dimensioni la quantitá di guasti del tipo *stuck-at* da considerare é molto grande e poiché la loro simulazione é un processo particolarmente costoso, dal punto di vista dei tempi di calcolo conviene ridurne il numero di quelli che devono essere effettivamente simulati.

In particolare, si definiscono come guasti *equivalenti*, quelli rivelati dai medesimi vettori di *test*. In altri termini, due guasti F_1 ed F_2 equivalenti hanno per definizione lo stesso effetto sulle uscite del circuito per qualunque sequenza di vettori di ingresso. In conseguenza di questa proprietà, i guasti del tipo *stuck-at 0/1* possono essere divisi in classi di equivalenza, il che consente di ridurre in in maniera significativa, il numero di casi da considerare (pari al numero di classi di equivalenza).

Le classi di equivalenza per guasti del tipo *stuck-at* nel semplice circuito di Fig. 1.4 sono elencate in Tab. 1.1. Ad esempio, i guasti **W2**, **I3**, **I4** *stuck-at-1* sono rivelati dallo stesso vettore di collaudo ($\mathbf{I1}, \mathbf{I2}, \mathbf{I3}, \mathbf{I4} = 1100$).

Inoltre, nel caso in cui tutti i vettori di test che rivelino un guasto F_1 rivelano anche F_2, ma non viceversa, si dice che F_1 *domina* F_2. In conseguenza di questa definizione, dal punto di vista della generazione di vettori di test si possono considerare solamente i guasti dominanti (F_1).

Suddividendo il circuito in classi di dominanza, si riduce il numero di vettori di test (ulteriormente rispetto al numero di classi di equivalenza) che devono essere generati per un determinato circuito.

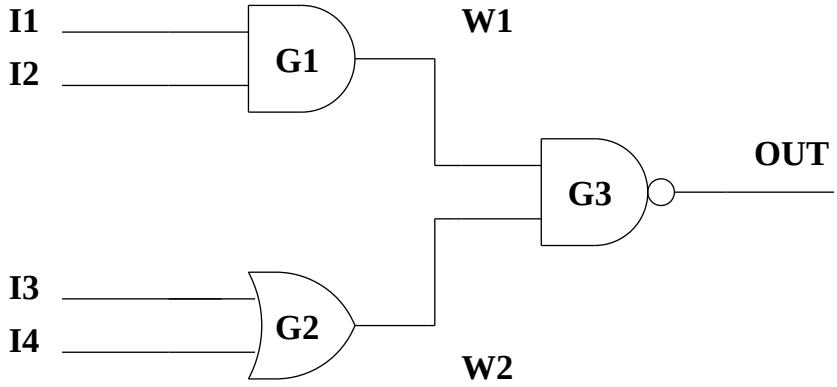


Figura 1.4: Esempio di semplice circuito combinatorio utilizzato per alcuni degli esempi di questo capitolo.

classe	linea	guasto
1	OUT	stuck-at-1
	W1	stuck-at-0
	W2	stuck-at-0
	I1	stuck-at-0
	I2	stuck-at-0
2	OUT	stuck-at-0
3	W1	stuck-at-1
4	W2	stuck-at-1
	I3	stuck-at-1
	I4	stuck-at-1
5	I1	stuck-at-1
6	I2	stuck-at-1
7	I3	stuck-at-0
8	I4	stuck-at-0

Tabella 1.1: Classi di guasti equivalenti nel circuito di Fig. 1.4.

Considerando sempre il circuito di Fig. 1.4 come esempio, si può osservare che il guasto "I1 stuck-at-1" domina "W1 stuck-at-1" e OUT "stuck-at-0"; infatti, i vettori di test che rivelano il primo guasto sono in grado di rivelare anche gli altri, ma non viceversa (vedi Tab. 1.2). La lista completa delle classi di dominanza per

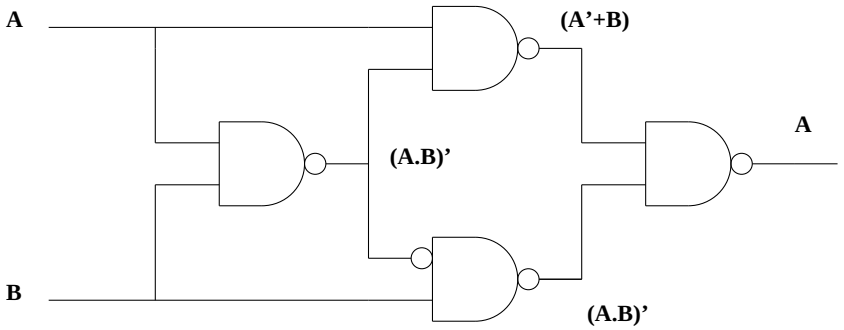


Figura 1.5: Esempio di rete ridondante da un punto di vista logico (come si può vedere dalle espressioni relative a ciascun segnale) in cui la linea di ingresso **A** non è mai osservabile in uscita e quindi qualsiasi guasto di tipo *stuck-at-0/1* su di essa non è rivelabile.

questo circuito è riportata in Tab. 1.3. Nell'esaminarla, si tenga presente che, diversamente dal caso delle classi di equivalenza, un guasto può essere dominato da più di un altro e pertanto può comparire in più di una classe di dominanza.

Un ultimo problema da analizzare è dato da guasti di tipo *stuck-at* non rivelabili: questo accade in presenza di ridondanze logiche nel circuito, che rendono non attuabili le condizioni di attivazione (ad esempio la linea sede del guasto può non essere controllabile al valore opposto a quello del guasto) o propagazione (la linea sede del guasto può non essere mai osservabile in uscita). In Fig. 1.5 è riportato un esempio di rete ridondante cui alcuni guasti di tipo *stuck-at* non sono rivelabili.

Nonostante questi guasti non alterino le funzioni svolte dal circuito, costituiscono ugualmente un problema in quanto: a) le cause fisiche del guasto possono comunque portare a un degrado dell'affidabilità del circuito; b) in presenza di sequenze di guasti (guasti multipli) possono dare luogo a notevoli problemi di collaudo. Pertanto questo tipo di guasti deve essere tipicamente rimossi durante la sintesi dei circuiti.

1.2.2 Guasti del tipo *bridging*

Molti malfunzionamenti e difetti di tipo fisico non possono essere rappresentati con un modello di tipo *stuck-at 0/1* (Fig. 1.6) [7, 3]. Fra questi, assumono particolare importanza quelli dovuti alla presenza di connessioni indesiderate tra linee normalmente isolate Questo tipo di guasto, spesso impropriamente denominato cortocircuito (*short*), ma più propriamente noto con il termine di *bridging*, può essere dovuto a ragioni fisiche diverse, quali l'elettromigrazione delle metallizzazioni, il disallineamento fra maschere ed eventuali perforazioni dell'ossido di

I1 I2 I3 I4	guasti rivelati
0101	I1 stuck-at-1, W1 stuck-at-1, OUT stuck-at-0
0110	”
0111	”
0011	W1 stuck-at-1, OUT stuck-at-0

Tabella 1.2: Vettori di collaudo che rivelano guasti appartenenti a una classe di dominanza nel circuito di Fig. 1.4.

classe	linea	guasto
1	OUT	stuck-at-1
	W1	stuck-at-0
	W2	stuck-at-0
	I1	stuck-at-0
	I2	stuck-at-0
2	*W2*	stuck-at-1
	I3	stuck-at-1
	I4	stuck-at-1
3	I1	stuck-at-1
	W1	stuck-at-1
	OUT	stuck-at-0
4	I2	stuck-at-1
	W1	stuck-at-1
	OUT	stuck-at-0
5	I3	stuck-at-0
	W2	stuck-at-0
	OUT	stuck-at-1
6	I4	stuck-at-0
	W2	stuck-at-0
	OUT	stuck-at-1

Tabella 1.3: Classi di equivalenza e dominanza nel circuito di Fig. 1.4. I guasti dominati sono riportati in corsivo.

isolamento tra metallizzazioni di diversi livelli.

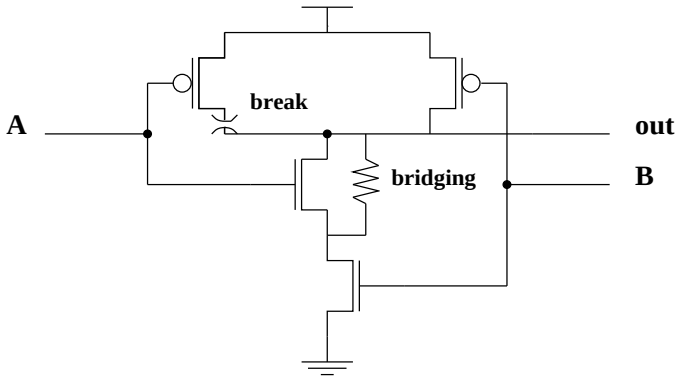


Figura 1.6: Esempio di *gate* CMOS con guasti non descrivibili mediante il modello *stuck-at*. Come si vedrà in questa e nella sezione seguente, l'interruzione di una connessione (*break*) dà luogo a un comportamento di tipo sequenziale del *gate*. La presenza di una connessione indesiderata *bridging* può fare sì che l'uscita si porti a un valore di tensione intermedio non rappresentabile con valori logici standard.

La possibile presenza di guasti del tipo *bridging* viene ad avere importanza crescente con l'aumento della densitá di integrazione, in quanto le sempre minori distanze fra metallizzazioni aumentano notevolmente la probabilitá di formazione di connessioni indesiderate all'interno di circuiti.

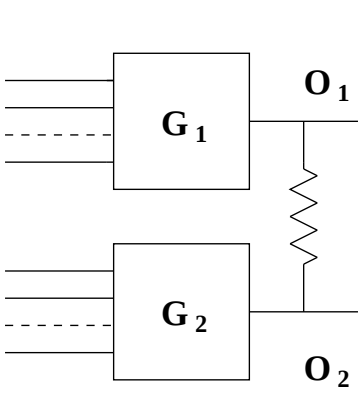
La particolaritá di questo tipo di guasti é quella di mettere in conflitto reti che, normalmente, piloterebbero valori logici diversi (Fig. 6), per cui le due linee coinvolte dal guasto vengono ad assumere un valore intermedio fra quelli alto (V_H) e basso (V_L), dipendentemente dal rapporto di conduttanze fra le reti connesse.

In alcune tecnologie, ad esempio in quelle MOS a rapporto, é sempre il valore basso a prevalere, poichè le reti di *pull-up* che collegano le uscite dei *gate* all'alimentazione sono meno conduttive di quelle di *pull-down*.

Questa caratteristica viene descritta dal modello di guasto di tipo *wired-and* (oppure *wired-or* nel caso opposto, in cui prevale il valore alto) che consente di determinare senza ambiguità il risultato del conflitto logico provocato dalla presenza del guasto (Fig. 1.7) [8, 9, 10].

Per questo motivo il guasto può essere trattato al livello logico, in modo analogo al caso degli *stuck-at*.

Nei circuiti CMOS, viceversa, le conduttanze delle reti di *pull-up* e di *pull-down* sono comparabili e, poichè il valore di tensione cui si portano le linee connesse dal *bridging* dipende dal rapporto fra le loro conduttanze, questo non può essere determinato a priori (Fig. 1.8) [11, 12, 13, 14, 15].



fault-free	wired-and	wired-or
O_1 O_2	O_1 O_2	O_1 O_2
00	00	00
01	00	11
11	11	11
10	00	11

Figura 1.7: Valori assunti da due linee di uscita di *gate* cortocircuitate secondo il modello di guasto del tipo *wired-and/or* confrontati con quelli presenti nel caso privo di guasti (fault-free).

Per guasti del tipo *bridging* le condizioni di attivazione e propagazione consistono rispettivamente nel pilotare le linee cortocircuitate a valori logici diversi e nel propagare verso le uscite del circuito il valore logico della linea che fra le due ha un cambiamento di stato come conseguenza del guasto.

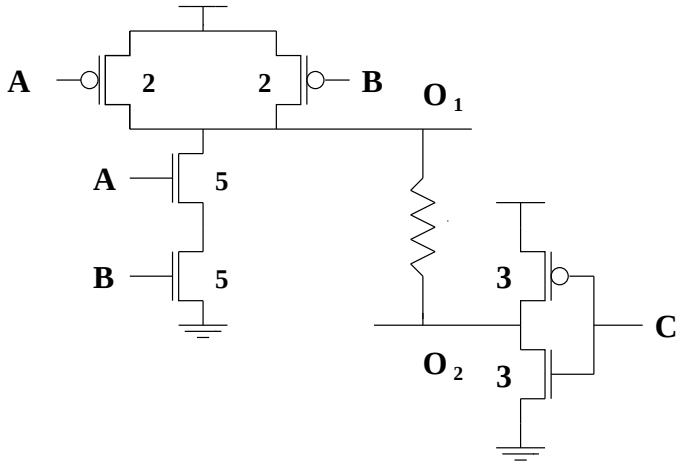
Diversamente dai casi in cui si può utilizzare il modello di tipo *wired-and/or*, nei circuiti CMOS, per determinare quale delle linee connesse dal guasto cambia di stato rispetto al caso corretto, è necessario conoscere la posizione dei valori di tensione intermedi assunti da tali linee *bridging* rispetto alla soglia logica dei rispettivi *gate* di *fan-out* (V_{LT}).

Infatti, il guasto è rivelabile solamente se una delle due linee di segnale (chiaramente, quella pilotata dalla rete meno conduttiva) si porta dalla parte opposta di V_{LT} rispetto al suo valore privo di guasti. Se tale linea è osservabile, il suo valore di tensione intermedio verrà normalizzato a un valore logico errato dopo la propagazione attraverso pochi *gate* (a causa dell'alto guadagno dei *gate* CMOS) dando così luogo a un errore logico sulle uscite del circuito (vedi esempio di Fig. 1.9).

Naturalmente, anche nel caso dei *bridging* possono esistere guasti non rivelabili ciò accade ad esempio se la linea che cambia di stato non è mai osservabile nelle condizioni in cui il guasto è attivato (vedi Fig. 1.10).

Oppure, nel caso in cui il valore di resistenza del *bridging* non sia trascurabile rispetto alle conduttanze di uscita dei *gate*. In tali condizioni, infatti, è possibile che a causa della caduta di tensione sulla resistenza del *bridging*, entrambe le linee rimangano dalla stessa parte della soglia logica rispetto al loro valore corretto.

Per quello che riguarda quest'ultimo problema, è importante notare che la presenza di *bridging* con valori di resistenza non trascurabili è stata riscontrata



ABC	O_1, O_2 (fault-free)	$V(O_1) \simeq V(O_2)$ (faulty)
000	11	-
001	10	$> V_{LT}$
010	11	-
011	10	$< V_{LT}$
100	11	-
101	10	$< V_{LT}$
110	01	$> V_{LT}$
111	00	-

Figura 1.8: Esempio di guasto del tipo cortocircuito i cui effetti non possono essere rappresentati mediante il modello di tipo *wired-and/or*. La soglia logica dei *gate* di *fan-out* di O_1 e O_2 è V_{LT}; inoltre, si consideri la resistenza del corto trascurabile rispetto a quella dei transistori accesi; i numeri a fianco dei transistori indicano i relativi valori di conduttanza normalizzata. Nella tabella sono riportati, rispettivamente, i valori di O_1 e O_2 nel caso privo di guasti e quelli ottenuti tramite simulazione elettrica in presenza del guasto. Come si può vedere, questi ultimi non sono riconducibili al modello *wired-and/or* perchè, dipendentemente dal fatto che nel *pull-up* del NAND CMOS siano attivi 1 o 2 transistori canale p, prevale il valore alto o quello basso. Questo comportamento non è tenuto in conto dai tradizionali simulatori a livello *switch* per i quali la conduttanza di tale rete rimane la stessa indipendentemente dal fatto che ci siano uno o due transistori accesi.

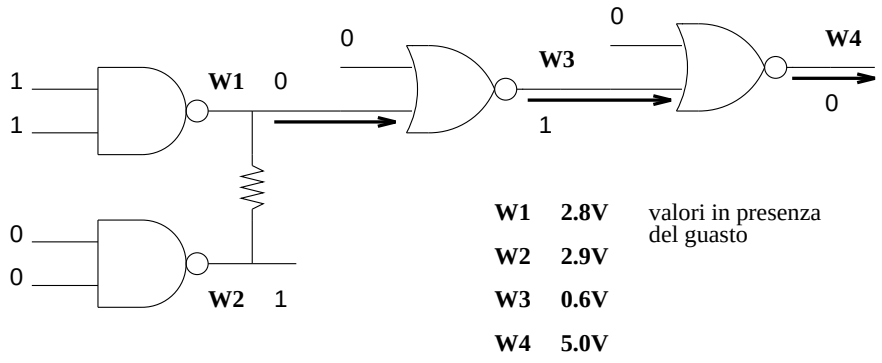


Figura 1.9: Esempio di propagazione degli effetti di un guasto di tipo *bridging* in un circuito CMOS. Si suppone che la soglia logica dei *gate* sia $\simeq 2.5V$; per cui il valore di tensione intermedio ($2.8V$) cui si porta la linea **W1** viene propagato come se fosse un 1 logico (e quindi un errore). Il valore della linea **W2** ($2.9V$), invece, è in pratica equivalente a un valore privo di guasti.

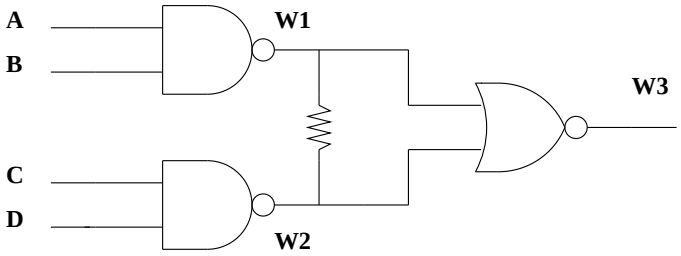


Figura 1.10: Esempio di guasto di tipo *bridging* non rivelabile. Il guasto è attivato se l'uscita dei due NAND sono diverse: in queste condizioni nel circuito privo di guasti l'uscita del NOR (**W3**) vale 0. Se si fa l'ipotesi che le reti di *pull-up* siano sempre più conduttive di quelle di *pull-down* in modo che prevalga sempre un valore di tensione alto (ovvero al di sopra della soglia logica), allora anche in presenza del guasto l'uscita **W3** rimane a un valore basso, indistinguibile da quello corretto.

sperimentalmente con una distribuzione di probabilità per le resistenze dei guasti che va da valori trascurabili a qualche $k\Omega$ [14] e costituisce un problema che non può essere trascurato nella modellistica e simulazione di guasto.

In particolare, i *bridging* sono guasti che coinvolgono un valore continuo della resistenza e sono, pertanto, detti di tipo *parametrico*. Per guasti di questo tipo, la rivelabilità o meno dipende dal valore assunto da tale parametro.

Per studiare la rivelabilità di un *bridging*, sarebbe pertanto necessario conoscere il valore di tale resistenza, per altro non deterministicamente noto (in quanto

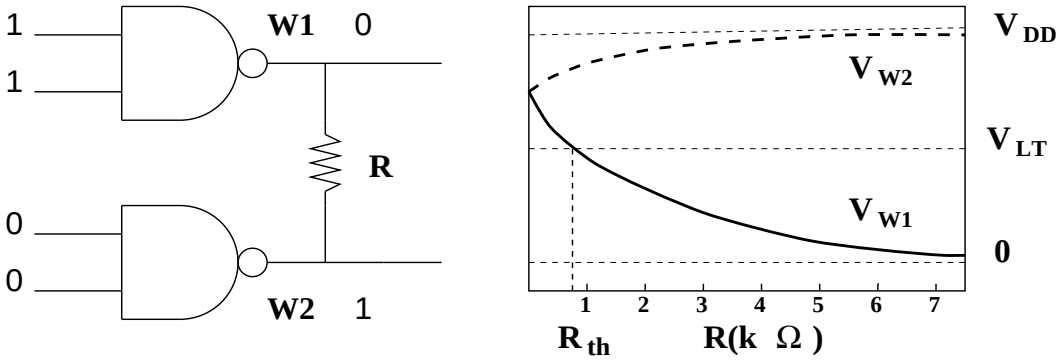


Figura 1.11: Andamento della tensione di uscita di un *gate* in presenza di un *bridging* in funzione della resistenza del guasto. Come si può vedere, nell'intervallo $[0, R_{th}]$, la tensione di uscita rimane dalla parte opposta del valore corretto permettendo di rivelare il guasto. Per valori di resistenza superiori a questo il valore di tensione di **W1** viene propagato come un valore corretto.

dipendente sia dalle modalità di guasto che dal *layout* del circuito), e diventa importante determinare un valore di soglia per tale parametro al di sotto del quale il guasto dà luogo a un errore logico (Fig. 1.11).

Una volta note tali soglie e la distribuzione dei valori di resistenza, diventa possibile determinare in modo statistico la copertura di guasti (o meglio, il suo valore atteso).

D'altra parte, occorre ricordare come nonostante alcuni di questi guasti non siano rivelabili, da un punto di vista logico, la loro presenza può alterare consistentemente le prestazioni dinamiche della parte di circuito affetta dal guasto e quindi, dipendentemente dalla frequenza di *clock* a cui opera il circuito, ciò può dare luogo o meno a errori di campionamento da parte dei *flip-flop* [14].

Infatti, si consideri una transizione su una linea affetta da un *bridging* che si propaga alle uscite del circuito: se il valore intermedio raggiunto dalla tensione di tale linea (una volta esaurito il transitorio) giace dalla stessa parte del valore corretto rispetto alla soglia logica dei *gate* di *fan-out*, allora si avrà una transizione corretta dell'uscita anche se fortemente ritardata perchè: a) la rete connessa dal *bridging* alla linea considerata sottrae corrente alla capacità di carico (vedi Fig. 1.12); b) il valore finale della linea considerata non è comunque quello corretto e pertanto i *gate* di *fan-out* attraverso cui si propaga la transizione avranno essi stessi degli incrementi nei ritardi di propagazione a causa dell'insufficiente tensione sopra soglia dei propri transistori [16]. Questo fatto è illustrato, nel caso del circuito di Fig. 1.12, dove in presenza di un *bridging* una transizione si propaga attraverso due NAND di *fan-out*, come nel caso del circuito corretto perchè il valore di tensione cui si porta la linea **W2** è $> V_{LT}$ (altrimenti non si

propagherebbe alcuna transizione e il guasto sarebbe rivelabile come uno *stuck-at*). Come si può vedere dalle forme d'onda dei segnali, non si incrementa solo il ritardo di propagazione attraverso il *gate* guasto, ma anche quello attraverso i *gate* di *fan-out*.

Guasti del tipo descritto in precedenza (che non danno luogo a errori logici) possono essere rivelati in due diversi modi. Nel primo si cerca di rivelare i *bridging* come guasti di tipo ritardo.

La seconda tecnica utilizzabile nei circuiti CMOS [17] si basa sull'osservazione che questo tipo di circuiti presentano, nel caso privo di guasti un consumo di corrente statica trascurabile mentre in presenza di un *bridging* si può avere un notevole assorbimento di corrente statica (indipendentemente dal fatto che il guasto sia rivelabile con un collaudo di tipo funzionale o meno) che può essere rilevato da della circuiteria di *sensing* (esterna al *chip*, oppure direttamente integrata nel *chip* stesso [18]).

Questa tecnica, generalmente denominata I_{ddq} *testing* [17, 18, 19, 20, 21], in teoria è in grado di rivelare tutti i *bridging*. In realtà presenta anch'essa alcuni notevoli problemi: 1) il livello di corrente originato dal guasto deve essere distinguibile da quello dovuto alle normali perdite in un circuito privo di guasti (ma questo non è un serio problema per valori tipici di resistenza); 2) la misura di corrente statica deve essere fatta quando tutte le correnti dovute ai transitori sono divenute trascurabili. Queste esigenze pongono dei vincoli sui tempi di collaudo, (in particolare sulla frequenza con cui possono essere applicati i vettori di collaudo che è tipicamente da uno a due ordini di grandezza inferiore a quella utilizzabile nel collaudo di tipo funzionale) e sull'accuratezza del circuito di *sensing*.

Tutti i guasti di tipo *bridging* considerati fino ad ora collegano fra loro le uscite di due *gate*. Questo insieme costituisce una frazione notevole di quelli possibili *bridging* in un circuito, ma non esaurisce senz'altro tutte le possibilità.

Infatti, in teoria è possibile la presenza di connessioni fra due qualsiasi nodi del circuito; in pratica, sono molto comuni quelle fra il terminale di controllo *gate* di un transistore e quello di *source* o *drain* (questo a causa di perforazioni nell'ossido di *gate*) o i guasti interni ai singoli *gate* CMOS.

Le considerazioni fondamentali svolte per i *bridging* fra uscite di *gate* valgono anche in questi casi, anche se è importante notare che la simulazione di guasti di tipo *bridging* interni ai *gate* CMOS richiede l'utilizzo di una rappresentazione del circuito tipicamente a livello *switch*: in particolare, è necessario che i due nodi connessi dal guasto siano, nel circuito corretto connessi ad alimentazioni opposte e che uno di essi sia connesso all'uscita del *gate* (che deve a sua volta essere osservabile alle uscite del circuito).

La prima condizione è di per sè sufficiente per rivelare il guasto mediante I_{ddq} *testing*, naturalmente con le limitazioni già ricordate riguardo al valore della corrente indotta dal guasto e dalle velocità delle prove. ESEMPIO

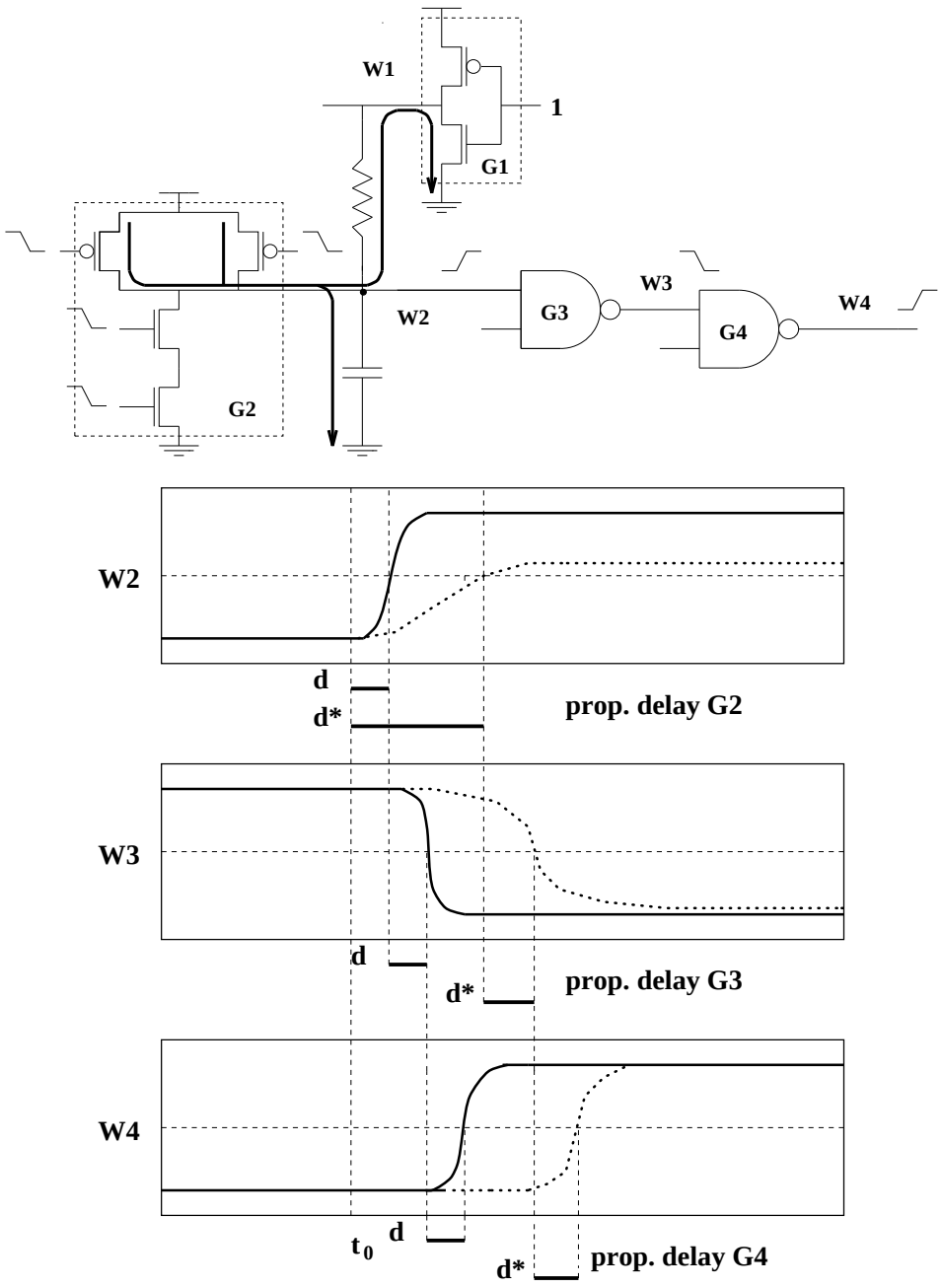


Figura 1.12: Esempio degli effetti di un *bridging* sul comportamento dinamico di un circuito: nel caso privo di guasti il *gate* **G2** ha una transizione da un valore basso a uno alto; a causa del *bridging*, invece, la sua uscita raggiunge un valore di poco superiore alla soglia logica. Quindi la transizione viene ugualmente propagata, ma con un notevole ritardo. Infatti, il transitorio di **G2** è rallentato dalla corrente che viene sottratta tramite il *bridging*, inoltre, come si può vedere dal diagramma in cui sono riportate le forme d'onda (a tratto pieno per il circuito privo di guasti e tratteggiate per quello con il *bridging*) anche i ritardi di propagazione di **G3** e in piccola parte di **G4** sono incrementati.

Dal punto di vista della simulazione questo tipo di guasti é particolarmente problematico, sia per la presenza di valori di tensione intermedi difficilmente modellabili al livello logico, sia perché, pur limitandosi ai connessioni fra due linee, tra le l presenti in un circuito, si hanno l^2 casi da simulare. È evidente che questo numero diventa estremamente alto per ualunque valore realistico di l.

In questo caso, una possibile strategia consiste nel considerare un insieme ridotto di tutti i possibili *bridging*, scelto in base alla probabilità di occorrenza (valutabile però solo se è noto il *layout* [5]). Esempi significativi in questo senso sono i *bridging* fra le linee di ingresso del circuito o fra le linee di ingresso di ciascun *gate*.

1.2.3 Guasti tipici della tecnologia CMOS

Le logiche CMOS, caratterizzate dalla sostanziale assenza di assorbimento di corrente in condizioni stazionarie, presentano condizioni di malfunzionamento che non sono in generale riconducibili a modelli di guasto del tipo *stuck-at 0/1*.

In particolare, gli aspetti più importanti da questo punto di vista sono: a) il comportamento complementare delle reti di *pull-up* e *pull-down* dei *gate*, per cui in condizioni statiche una delle due reti non conduce, mentre l'altra connette l'uscita del *gate* alla massa o all'alimentazione; b) i valori non trascurabili di capacità parassite presenti fra le uscite dei *gate* e la massa.

Per trattare adeguatamente i malfunzionamenti in questo tipo di logiche vengono considerati modelli di guasto del transistore, che rappresentano malfunzionamenti dovuti a imperfezioni o rotture nei contatti e degradazioni nei dispositivi (ad es. alterazioni della soglia di conduzione). Questi debbono essere necessariamente descritti a un livello più basso di quello di *gate*. In particolare, a questo proposito si utilizza largamente il livello *switch* in cui il transistore MOS viene visto come un interuttore digitale. I modelli di guasto utilizzati in questo ambito sono i seguenti:

Transistor stuck-on

Questo modello rappresenta un dispositivo che conduce indipendentemente dalla differenza di tensione fra i terminali di *gate* e di *source*.

La caratteristica principale del *transistor stuck-on* è la possibilità di alterare il funzionamento complementare dei gate CMOS. Infatti, se attivato (Fig. 1.13) tale guasto fa sì che l'uscita di un gate sia contemporaneamente collegata all'alimentazione e alla massa, a causa della presenza di un cammino conduttivo attraverso dispositivo guasto (il quale sarebbe aperto in condizioni corrette, similarmente a quanto accade nel caso dei *bridging* interni).

Come conseguenza, l'uscita del *gate* si porta a un valore di tensione diverso

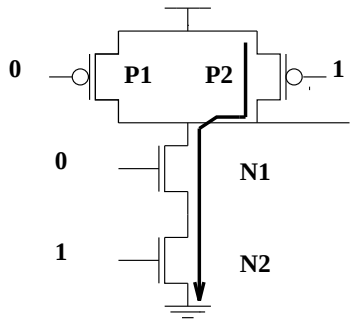


Figura 1.13: Esempio di guasto di tipo *transistor stuck-on* in un *gate* NAND CMOS. Il transistore $N1$ è bloccato acceso nonostante sia pilotato da un valore basso, pertanto si ha un cammino conduttivo fra l'alimentazione e la massa e l'uscita assume un valore di tensione intermedio.

da quelli corretti alto (V_{DD}) e basso (V_{SS}), che dipende dal rapporto fra le conduttanze della rete guasta e di quella che si trova correttamente in conduzione.

Questo tipo di guasto ha un comportamento simile a quello di un *bridging* tra i nodi di *source* e *drain* del dispositivo *stuck-on*, che presenta una resistenza tipicamente simile a quella del transistore correttamente acceso. Per questo motivo, i guasti del tipo *transistor stuck-on* possono essere considerati come casi particolari di *bridging* da cui vengono distinti unicamente per motivi storici e comodità di classificazione. Inoltre, molti di essi risultano non rivelabili (in percentuale maggiore rispetto al caso generale di guasti di tipo *bridging*).

D'altra parte, la presenza di un cammino conduttivo fra la massa e l'alimentazione, dovuta a un transistore guasto, provoca un assorbimento di corrente in condizioni statiche (che, come noto, non é presente nei *gate* CMOS), che puó essere rilevato mediante misure di corrente statica.

1.2.4 Transistor stuck-open

Questo modello di guasto descrive un transistore MOS che non riesce ad andare in conduzione (ad esempio a causa di un contatto aperto) nemmeno quando pilotato con valori di tensione di controllo più alti della sua soglia di conduzione.

Per attivare un *transistor stuck-open* in un *gate* CMOS, bisogna cercare di controllare l'uscita del *gate* con il guasto (ai valori basso o alto) attraverso un cammino (verso la massa o l'alimentazione) che includa il dispositivo *stuck-open*. Poichè quello è acceso nel circuito corretto, ma spento in quello guasto, l'uscita del *gate*, anzichè portarsi al valore atteso, si porta in uno stato ad alta impedenza (poichè uno dei due cammini è normalmente spento mentre l'altro non conduce a causa del guasto). Come conseguenza degli effetti capacitivi non trascurabili connessi ai nodi di segnale, l'uscita del gate guasto conserverà il valore di tensione

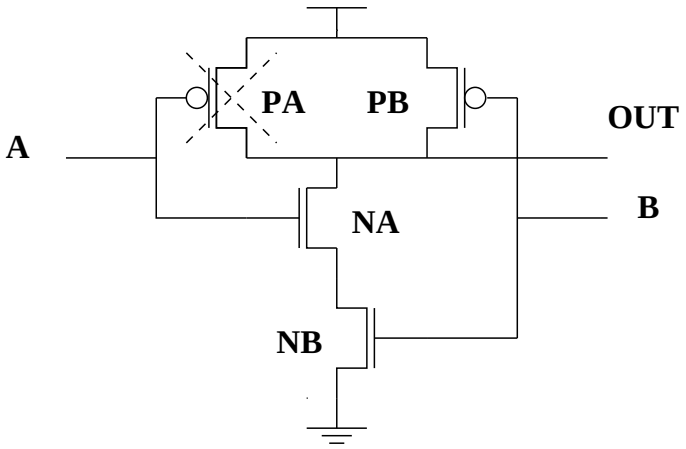


Figura 1.14: Guasto di tipo *transistor stuck-open* in un *gate* CMOS. Per riverarlo occorre: a) inizializzare l'uscita a un valore basso (apllicando il vettore di ingresso $\mathbf{A} = 1$, $\mathbf{B} = 1$; b) tentare di caricare l'uscita attraverso il transistore guasto **PA**. In presenza del guasto, l'uscita rimane a un valore basso e è osservabile, il guasto viene rivelato.

che aveva prima dell'attivazione del transistor *stuck-open*. Per questo motivo, lo *stuck-open* trasforma un circuito combinatorio in uno sequenziale.

A questo proposito, è evidente che, poichè il valore logico presente sul nodo d'uscita del *gate* prima dell'attivazione del guasto può essere uguale a quello corretto, il guasto può non essere rivelato.

Per superare questo problema, bisogna applicare due vettori di *test*: il primo per inizializzare l'uscita a un valore opposto a quello che sarebbe imposto nel caso corretto dalla rete (*di pull-up* o di *pull-down*) affetta dal guasto, il secondo vettore per cercare di fare assumere all'uscita del *gate* un valore opposto a quello inizializzato (Fig. 1.14).

A questo punto, tuttavia, vale la pena di osservare come una coppia di vettori di *test*, potenzialmente in grado di rivelare un guasto di tipo *transistor stuck-open*, possa essere invalidata (cioè non riuscire a rivelare il guasto) a causa di [22, 23]:

1. *glitch* fra i segnali di ingresso durante la commutazione fra il primo e il secondo vettore di collaudo (che possono attivare temporaneamente cammini conduttivi nella rete contenente il transistore guasto tali da rimuovere il valore di inizializzazione);

2. fenomeni di ripartizione di carica (per cui il valore di inizializzazione memorizzato nella capacità di uscita viene degradato).

Per risolvere questo tipo di problemi sono state proposte opportune tecniche di progettazione orientata al collaudo che per brevità non possono essere descritte in questo capitolo.

Le considerazioni fino ad ora svolte valgono nel caso di *gate* CMOS statici, nel caso di circuiti CMOS dinamici (come, ad esempio, quelli denominati *Domino*), i guasti di tipo *transistor stuck-open* sono, invece, rivelabili da un singolo vettore di collaudo, in quanto l'inizializzazione dell'uscita del *gate* viene comunque fornita durante la fase di precarica.

1.2.5 Il modello di guasto del tipo *stuck-at 0/1* e le altre classi di guasti

Come si è visto, il modello del tipo *stuck-at* rappresenta solo una parte dei possibili malfunzionamenti che possono essere presenti in un circuito a grande densità di integrazione, in particolare di quelli realizzati con tecnologia CMOS.

D'altra parte, i programmi di simulazione e di generazione automatica di vettori di collaudo implementati negli odierni simulatori logici commerciali considerano, in generale, soltanto guasti di tipo *stuck-at*.

Diventa, quindi, importante determinare l'efficacia di sequenze di collaudo generate per rivelare guasti del tipo *stuck-at* nei confronti degli altri modelli di guasto per circuiti CMOS.

In relazione a questo problema, in Fig. 1.15 vengono rappresentate le percentuali di guasti del tipo *stuck-at*, *bridging*, *stuck-open* e *stuck-on* rivelati da una sequenza di *test* generata per rivelare guasti del tipo *stuck-at 0/1* [24]. Tale percentuale è calcolata in funzione del numero di vettori di *test*, nel caso di un ALU a 16 *bit* realizzata in tecnologia CMOS.

La sequenza di *test*, generata con un algoritmo deterministico ottiene una copertura (ovvero la percentuale di guasti di un certo tipo rivelati dall'insieme di vettori di *test* considerato) prossima al 100% nei confronti di guasti del tipo *stuck-at 0/1*, mentre per quanto riguarda gli altri tipi guasto si ottengono valori significativamente minori, in particolare per i *transistor stuck-open* e *stuck-on*.

Questo risultato, sufficientemente rappresentativo per tipici circuiti CMOS combinatori, mostra come nello sviluppo di simulatori e di algoritmi per la generazione di sequenze di *test*, sia necessario tenere conto esplicitamente anche dei guasti del tipo *bridging* e di quelli sui transistori.

1.3 Guasti di tipo ritardo

Le sempre più stringenti specifiche in termini di velocità degli odierni circuiti integrati hanno portato l'attenzione verso quei malfunzionamenti che, pur non

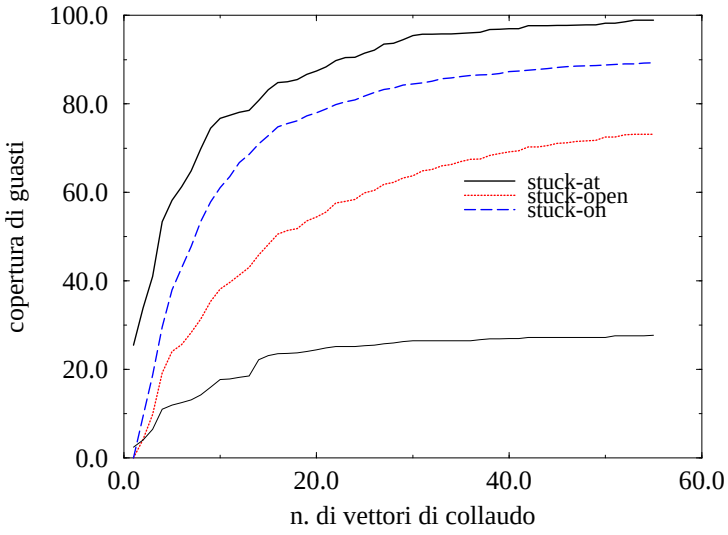


Figura 1.15: Copertura di guasti tipici dei circuiti CMOS in una ALU a 16 bit, confrontata comparata con quella ottenuta per i guasti di tipo *stuck-at* cui è orientata la sequenza di collaudo utilizzata.

alterando le funzioni logiche dei componenti del circuito, danno luogo a un degrado delle loro prestazioni dinamiche. A sua volta, tale degrado di questi che, dipendentemente dalle condizioni di funzionamento del circuito, può dar luogo ad errori logici (dovuti ad esempio al campionamento da parte dei *flip-flop* di segnali che non hanno ancora raggiunto il loro stato stabile) difficili da rivelare mediante *test* orientati alla rivelazione di guasti che alterano il comportamento del circuito in condizioni stazionarie, di qui l'esigenza di sviluppare specifiche tecniche di generazione dei vettori di collaudo per tali guasti.

Fra le possibili cause che possono dare luogo a questo comportamento, oltre ai *bridging*, esistono, anche altri difetti: ad esempio, variazioni nella geometria o nei parametri tecnologici dei transistori che ne riducono fortemente la transconduttanza.

Il modello più semplice in grado di tenere conto di questo tipo di malfunzionamenti consiste nel considerare che la transizione di uscita di un *gate* nel circuito corretto, non possa avere luogo in presenza del guasto.

Questo modello, che è in pratica la trasposizione nel caso dinamico dello *stuck-at*, è chiamato *transition-fault* o *conditional-stuck-at*. Per l'uscita di ciascun *gate* si hanno quindi due *transition-fault* denominati *slow-to-rise* e *slow-to-fall* che condizionano, rispettivamente, i transitori in salita e discesa.

Per rivelare un guasto di questo tipo in un circuito combinatorio, è necessario applicare due vettori di collaudo: il primo dei quali inizializza opportunamente

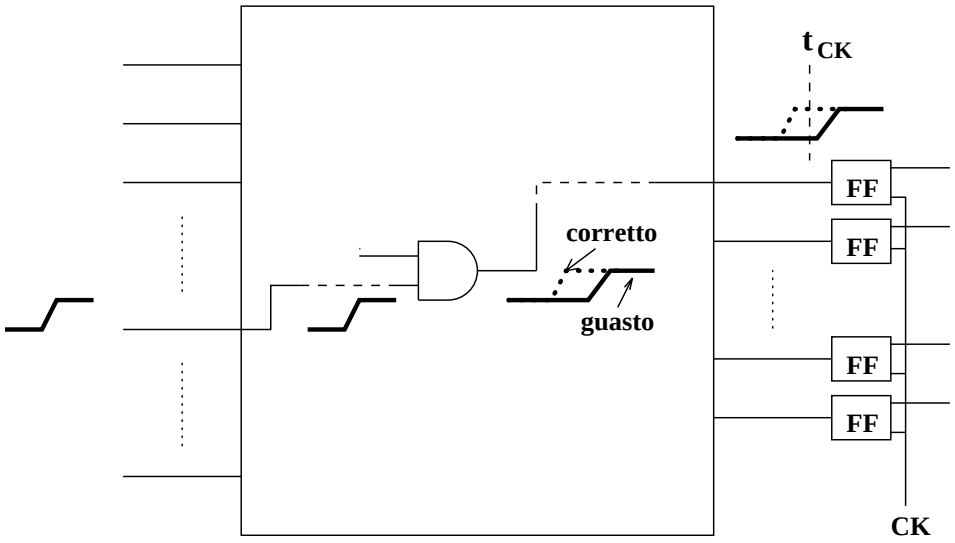


Figura 1.16: Esempio di rivelazione di guasto ritardo in un circuito di tipo combinatorio. Il ritardo di propagazione in salita del *gate* AND è maggiore di quello ammesso, dalle specifiche di progetto e tale da dare luogo al campionamento da parte di un *flip-flop* di un valore erroneo (0) qualora venga propagata una transizione attraverso il *gate*.

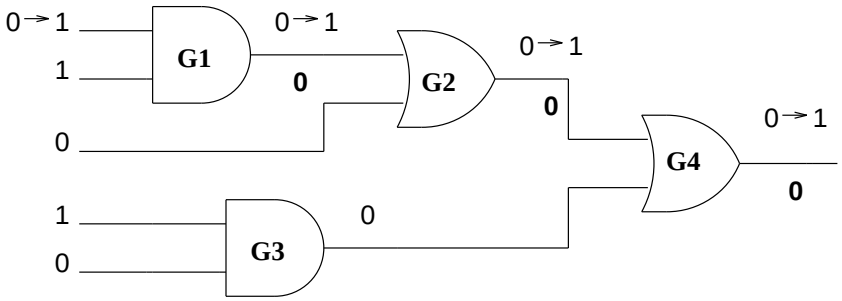


Figura 1.17: Effetti di un *transition fault*. Nel circuito corretto si propaga una transizione attraverso i *gate* **G1**, **G2** e **G4**; in presenza di un *transition fault* su **G1**, l'uscita di tale *gate* rimane a 0 e il guasto è rivelabile come uno *stuck-at-0* (nella figura, i valori del circuito guasto sono indicati in grassetto).

l'uscita del *gate* (a 0/1 se si vuole rivelare un guasto di tipo *slow-to-rise/fall*) e il secondo rivela lo *stuck-at* corrispondente al valore cui il guasto ha mantenuto tale linea, impedendone la transizione (0/1 nel caso di *slow-to-rise/fall*).

Questo modello, ha come ipotesi di partenza che il ritardo aggiuntivo dato dal malfunzionamento abbia durata almeno pari all'intervallo di tempo che intercorre fra l'applicazione di un vettore di collaudo e il campionamento delle uscite, permettendo così di prescindere dai ritardi nel circuito. Questo consente delle procedure di simulazione molto efficienti (in pratica, le stesse utilizzate nel caso di guasti di tipo *stuck-at*), ma non è particolarmente realistico, in quanto il ritardo dovuto a questo tipo di difetti è, in generale, minore da quanto imposto dal modello *transition-fault*. A questo riguardo, è stato introdotto il modello di guasto del tipo *ritardo* (*delay fault*) che fa, infatti, ipotesi più generali sull'entità del malfunzionamento (considerandola variabile in un certo intervallo). In questo caso bisogna considerare:

a) il valore dell'incremento del ritardo di propagazione del *gate* affetto dal guasto (questa quantità viene comunemente denominata *delay defect size*);

b) le tempistiche del circuito (ritardi dei *gate*, istanti di applicazione dei vettori di ingresso e delle uscite).

Per comprendere meglio queste esigenze, si consideri l'esempio di Fig. 1.18 che riporta una rete combinatoria (in cui a ciascun *gate*, secondo un modello molto semplice, sono associati ritardi di propagazione in salita e discesa) e l'andamento dei segnali nel caso privo di guasti.

Come si può vedere dalle forme d'onda dei segnali (Fig. 1.18) nel caso in cui il *gate* **G1** abbia un guasto di tipo ritardo (*slow-to-fall*) di *size* pari a $3ns$, il guasto è rivelabile in quanto all'uscita (W3) viene campionato un valore errato.

Lo stesso guasto nel *gate* G6, non è però rivelabile all'uscita (W5) in quanto la transizione dell'uscita avviene prima dell'istante di campionamento nonostante il ritardo (Fig. 1.20).

Come nell'esempio descritto, in gnerale, la rivelabilità dei guasti di tipo ritardo dipende non solo dal valore di un parametro *size* che li caratterizza, ma anche dalla loro posizione all'interno del circuito.

In particolare, tanto più la transizione affetta dal guasto verrà propagata lungo un cammino il cui ritardo in condizioni nominali è grande, tanto più il guasto sarà rivelabile per valori anche piccoli del *size*. Infatti, poichè nel circuito corretto l'intervallo (*slack*) fra l'arrivo della transizione alle uscite del circuito e l'istante di campionamento assume i valori minori proprio per i cammini di propagazione più lenti, che risultano quindi più sensibili a ritardi aggiuntivi.[2] È quindi facile

[2] (L'istante di campionamento viene calcolato per superare di poco, ma con un certo margine, l'arrivo della transizione con il ritardo massimo corrispondente al cammino logico più lento dentro il circuito (*critical path*).

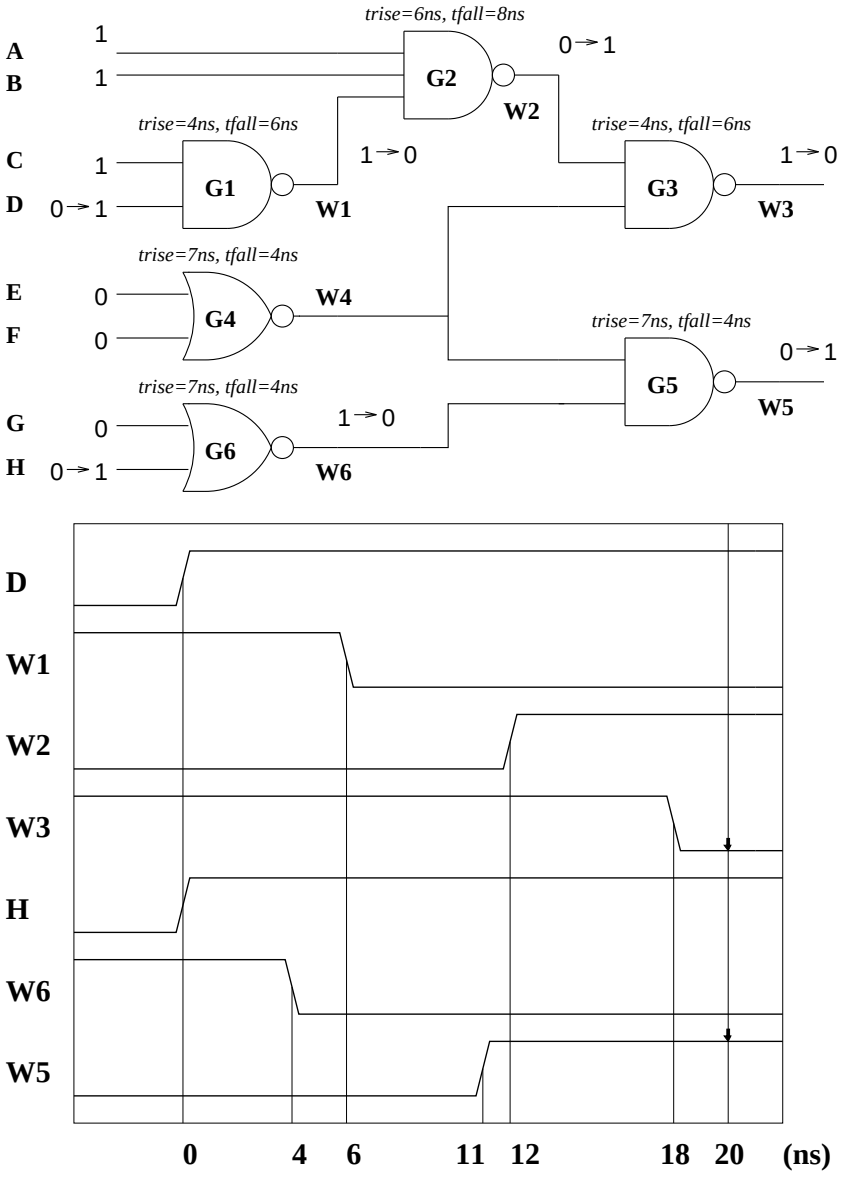


Figura 1.18: Esempio di valutazione delle forme d'ondea dei segnali in un circuito combinatorio ai cui *gate* sono stati associati ritardi di propagazione in salita e in discesa (*trise* e *tfall*) in condizioni nominali. Nell'esempio illustrato al tempo $t = 0$ vengono applicate due transizioni alle linee di ingresso **D** e **H** che si propagano alle uscite **W3** e **W5**, rispettivamente. Tali segnali vengono poi campionati da elementi di memoria a $t = 20ns$.

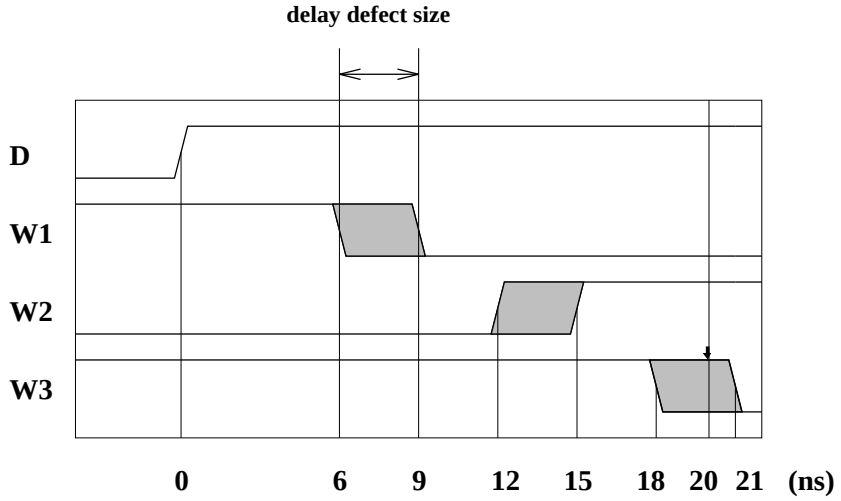


Figura 1.19: Forme d'onda nel circuito di Fig. 1.18, nel caso in cui il *gate* **G1** presenti un *delay fault* sul fronte di discesa di $size = 3ns$. Come si può vedere gli effetti di tale guasto si propagano fino all'uscita **W3** la cui transizione avviene a $t = 21ns$, invece che a $t = 18ns$; in questo modo viene campionato un valore logico alto invece che basso.

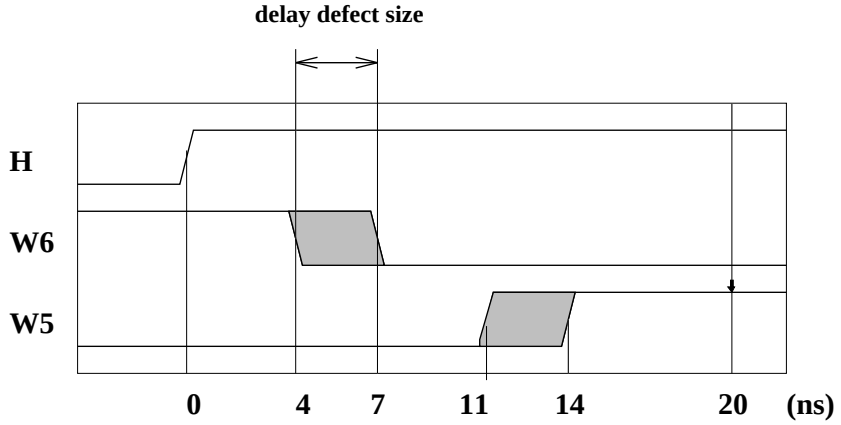


Figura 1.20: Forme d'onda nel circuito di Fig. 1.18, nel caso in cui il *gate* **G6** presenti un *delay fault* sul fronte di discesa di $size = 3ns$. Come si può vedere, nonostante gli effetti di tale guasto si propaghino fino all'uscita **W5**, questa ha una transizione ritardata a $t = 14ns$ che è comunque prima dell'istante di campionamento.

che con il contributo aggiuntivo dovuto al guasto, la transizione avvenga dopo il campionamento permettendo così la rivelazione del guasto.

Il modello di guasto fino ad ora considerato prevede che tutto il ritardo aggiuntivo sia concentrato in un singolo *gate* (ed è pertanto denominato *gate delay fault model*). Come alternativa, per rappresentare gli effetti di perturbazioni del processo in grado di danneggiare intere regioni del *chip*, dando luogo a un degrado delle prestazioni dei singoli *gate*, ma ritardi eccessivi lungo cammini di propagazione, è stato introdotto il *delay path fault model*, in cui il ritardo è distribuito su un intero cammino di propagazione.

Il problema principale di questo modello è dato dal numero di cammini da considerare, che in circuiti di grandi dimensioni con riconvergenza di *fan-out* può diventare intrattabile.

1.4 Modelli di guasto nelle strutture regolari

Le PLA e le memorie rappresentano tipici esempi di strutture regolari che trovano larghissimo utilizzo nella realizzazione di sistemi digitali. Inoltre esiste una crescente tendenza della forte tendenza alla realizzazione di architetture strutturate e di tipo gerarchico, che fanno uso di parti e componenti a struttura regolare.

Dal punto di vista del collaudo, i circuiti con struttura regolare pongono, in generale, problemi specifici, perchè possono presentare difetti al livello fisico e di layout che inducono malfunzionamenti diversi da quelli tipici dei circuiti non strutturati (generalmente definiti come *random logic*).

Per questi motivi, non è conveniente descrivere i malfunzionamenti associati alle PLA ed alle memorie con gli stessi modelli illustrati nel paragrafo precedente. D'altra parte, data la complessità dei circuiti considerati, non è evidentemente possibile un collaudo di tipo esaustivo. Di conseguenza, la strategia di *testing* non può che essere fondata su specifici modelli di guasto. Come esempi di questo tipo di problemi, considereremo nel seguito i casi delle PLA e delle memorie.

Programmable Logic Arrays

Le PLA sono largamente utilizzate nell'ambito dei microcircuiti VLSI per implementare funzioni combinatorie mediante una struttura regolare a 2 livelli (NOR-NOR in tecnologia nMOS) (concettualmente equivalente a un espressione SP AND-OR della funzione stessa). Come noto, la struttura di una PLA consiste di due matrici di linee di connessione (in Fig. 1.21 è illustrata l'implementazione mediante una PLA di tipo statico delle funzione di Fig. 1.22, che verrà utilizzata come esempio nel resto di questo paragrafo), di cui la prima, denominata piano AND, ha come righe le linee che implementano i termini prodotto presenti

nell'espressione della funzione realizzata e come colonne la decodifica degli ingressi. La seconda matrice è denominata piano OR ed ha come righe le linee prodotto e come colonne le linee di uscita della PLA ovvero i termini somma.

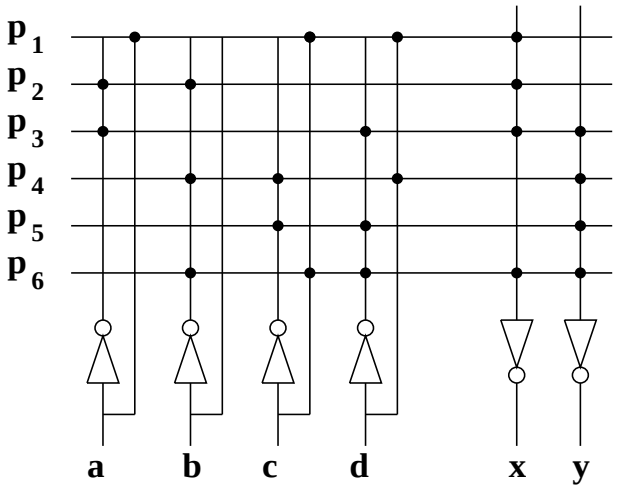


Figura 1.21: Esempio di struttura di una PLA (NOR-NOR) che implementa la funzione descritta in Fig. 1.22. La presenza del simbolo • denota la presenza di un transistore in un punto di incrocio.

La struttura delle PLA può essere programmata per implementare un qualsiasi funzione combinatoria, connettendo due linee che si incrociano mediante un transitore MOS. In particolare, il punto di incrocio di due linee viene definito come *cross-point* e su questo puó essere presente o meno una connessione realizzata con un transistore MOS.

Così come in un generico circuito digitale, le linee che portano i segnali (ingressi, decodifica degli ingressi, prodotto e somma) possono naturalmente avere guasti del tipo *stuck-at 0/1* o *bridging*.

Nelle PLA, però, queste classi di guasto non rappresentano il modo più efficiente per descrivere i malfunzionamenti più comuni, che sono costituiti dalla presenza o assenza, non volute, di un transistore MOS in un *cross-point*.

Questo tipo di guasti, generalmente denominati *cross-point defect*, possono essere classificati a seconda della loro posizione (sul piano AND o OR) e ricevono una denominazione che dipende da come alterano la funzione della PLA [25, 26, 27]. In particolare, si distinguono i seguenti casi:

- l'assenza (non programmata) di un transistore in un *cross-point* nel piano AND, che provoca un aumento delle dimensioni del termine prodotto corrispondente (*growth*);

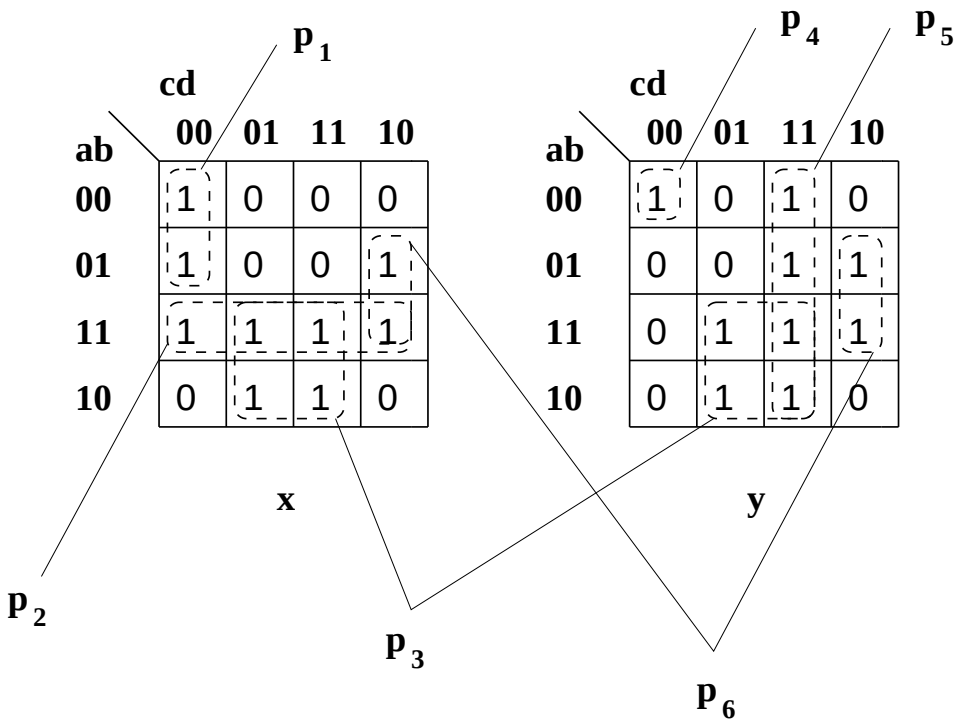


Figura 1.22: Funzione (a 4 ingressi e 2 uscite) implementata dalla PLA di Fig. 1.21.

- la presenza (non programmata) di un transistore in un *cross-point* del piano AND, che provoca una diminuzione delle dimensioni del termine prodotto corrispondente (*shrinkage*);
- l'assenza (non programmata) di un transistore in un *cross-point* nel piano OR, che provoca la scomparsa di un termine prodotto della somma corrispondente (*disappearence*);
- la presenza (non programmata) di un transistore in un *cross-point* del piano OR, che provoca la comparsa di un termine prodotto nella somma corrispondente (*appearence*).

Per meglio comprendere l'effetto di questi guasti, si può considerare un esempio per ciascuno di essi nella PLA di Fig. 1.21 (vedi Fig. 1.23). La Fig. 1.24 illustra per ciascuno di questi guasti gli effetti sulla funzione implementata dalla PLA.

In generale, una sequenza di *test* in grado di rivelare tutti i possibili guasti del tipo *stuck-at* in una PLA può non essere in grado di rivelare tutti i *crosspoint*

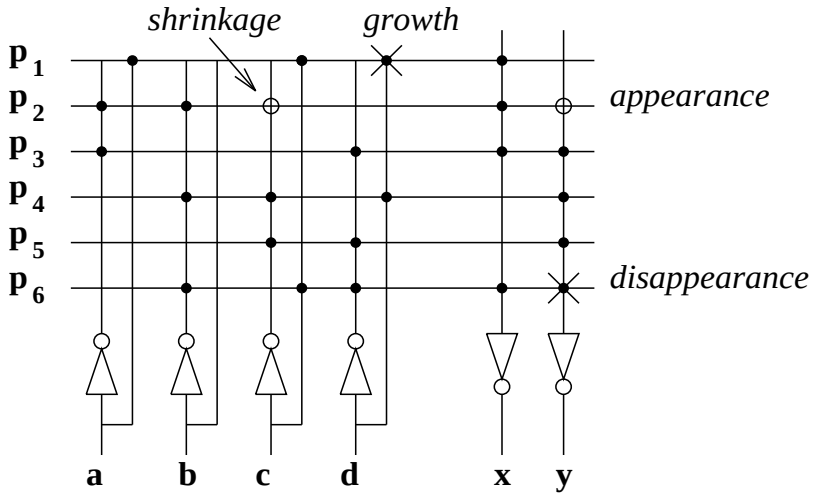


Figura 1.23: Esempi di possibili *crosspoint defect* in una PLA. Il simbolo $\circ$ indica la presenza indesiderata di un transistore in un punto di incrocio; il simbolo $\times$, invece, denota l'assenza di un transistore da un punto di incrocio.

defects. Viceversa, è stato dimostrato [28] che un insieme di vettori di collaudo in grado di rivelare tutti i guasti di questo tipo rivela anche tutti gli *stuck-at* e i *bridging*.

Per questo motivo esistono sia algoritmi per la generazione automatica di vettori di *test* che per la simulazione di questo tipo di guasti. Questi strumenti, risultano particolarmente efficenti a causa della regolarità e semplicità delle PLA.

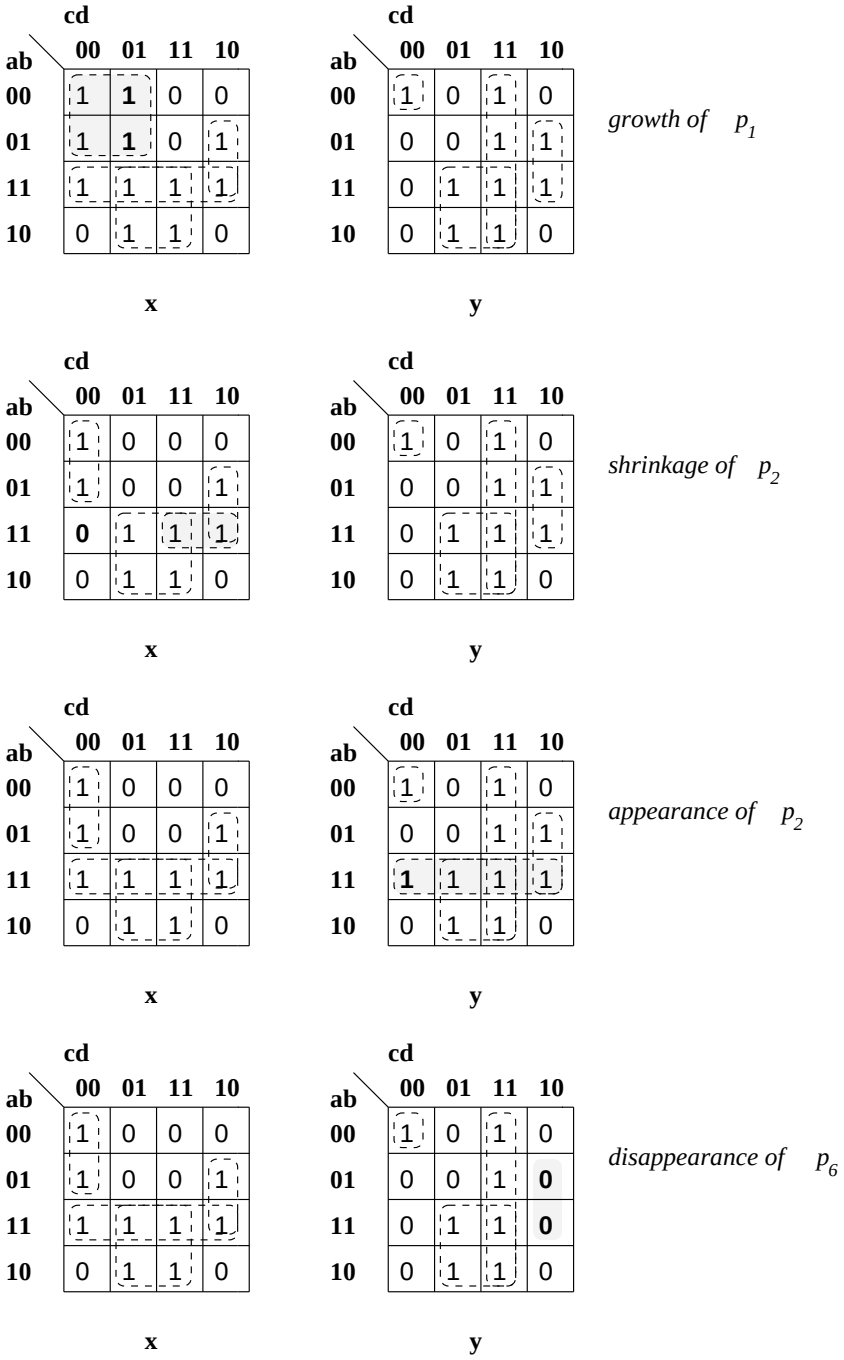


Figura 1.24: Effetto dei guasti illustrati in Fig. 1.23 sulla funzione svolta dalla PLA. I valori logici (0/1) in grassetto corrispondono a configurazioni di ingresso per le quali il valore della funzione con il guasto differisce da quello corretto.

1.4.1 Memorie

Nell'ambito dei sistemi digitali, le memorie sono fra i componenti più importanti dal punto di vista sia applicativo che tecnologico. Dal punto di vista del collaudo esse presentano problemi del tutto particolari [29] dovuti alle peculiari caratteristiche della loro architettura e tecnologia.

Prima di descrivere i modellistica di guasto utilizzati per questo caso, conviene ricordare brevemente una semplice classificazione delle memorie che possono essere suddivise sulla base della permanenza (1) o meno (2) dell'informazione a circuito non alimentato in:

1. non volatili (come ad esempio le ROM, ma anche le PLA viste nel capitolo precedente);

2. volatili (fra cui il tipo più diffuso è costituito dalle RAM (*Random Access Memory*), caratterrizzate dalla possibilità di leggere e scrivere i dati con ritardi comparabili. Le RAM, a loro volta, essere suddivise in memorie dinamiche (DRAM, Dynamic-RAM) e statiche (SRAM,Static-RAM), a seconda che le informazioni vengano memorizzate rispettivamente come carica all'interno di un condensatore o in un circuito rigenerativo (*flip-flop*).

In questo capitolo ci si occuperà del collaudo di memorie volatili (e in particolare delle RAM), poichè il collaudo di quelle non volatili è tipicamente più semplice e può essere generalmente ricondotto al caso di quelle volatili.

Per quello che riguarda l'architettura di una memoria, sono possibili diverse soluzioni, in questo capitolo si farà riferimento a uno schema molto generale Fig. 1.25. I cui componenti essenziali sono: una logica di indirizzamento (essenzialmente una serie di elementi di decodifica o *decoder*); una matrice di celle di memoria; nonchè una logica di scrittura e una di lettura.

Le modalità di indirizzamento, e quindi la struttura della logica di decodifica, riflettono da un punto di vista funzionale l'organizzazione della memoria (numero di bit per parola) permettendo di scrivere o leggere una parola per volta (tipicamente di 16 o 32 bit). Da un punto di vista interno, invece, l'organizzazione è tipicamente per righe e per colonne, in cui ciascuna cella di una colonna è collegata (con un pass-transistor attivabile autonomamente dagli altri della stessa colonna) a una bit line e le singole colonne sono poi multiplate in ingresso alla circuiteria di lettura e scrittura.

La struttura delle singole celle caratterizza il tipo di memoria. Nelle DRAM ciascuna di essa è costituita da un transistore MOS e da un condensatore, nelle SRAM, invece, ciascuna cella è costituita da un *flip-flop*.

Dal punto di vista del collaudo interessa notare una importante differenza con il tipo di circuiti (realizzati mediante gate) esaminati in precedenza. Questi

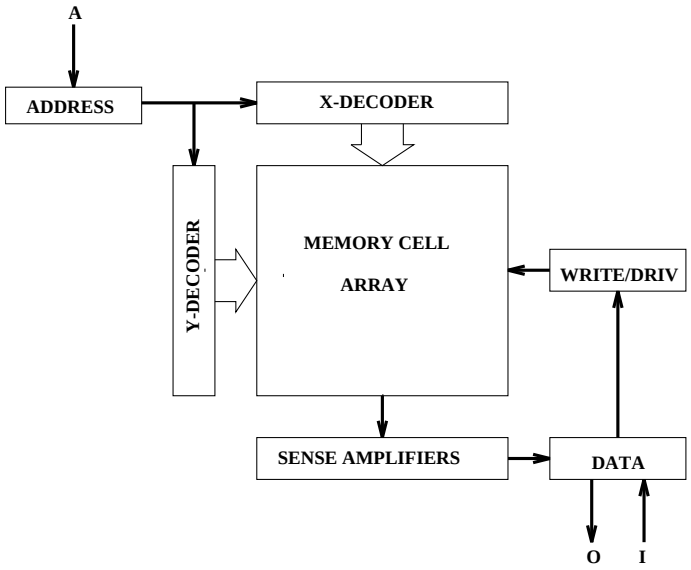


Figura 1.25: Rappresentazione schematica dei principali blocchi di una SRAM.

circuiti presentano margini di immunità ai disturbi e caratteristiche di funzionamento che li rendono abbastanza "robusti" rispetto a problemi di tipo *soft* rispetto a problemi quali, ad esempio, rumori sulle alimentazioni o perturbazioni statistiche dei parametri, mentre sono sensibili a difetti di tipo fisico come *break* e *bridging* descrivibili coi modelli esaminati in precedenza.

Le memorie, invece, sono particolarmente sensibili a disturbi di tipo *soft*, perchè la ricerca di migliori prestazioni e capacità è sempre ottenuta spingendo il progetto fino ai limiti consentiti dalla tecnologia (chiaramente a scapito dei margini di immunità ai disturbi e della sensibilità del circuito). Nel caso dei microprocessori, invece, altri aspetti (in questo caso di tipo architetturale) giocano un ruolo dominante.

Questo, è a maggior ragione vero nelle DRAM, in cui l'informazione viene memorizzata come una piccola quantità di carica all'interno delle singole celle e in cui il circuito che legge tale informazione (*sense-amplifier*) è un componente essenzialmente analogico. Di conseguenza, quindi evidente come rumori sulle alimentazioni, perdite dei dispositivi sotto soglia (*leakage*), radiazioni e asimmetrie nei componenti, possono dar luogo ad errori nei bit letti. Poichè rimane comunque possibile la presenza di guasti *hard*, è evidente come sia necessario considerare molteplici condizioni di guasto.

Come si vedrà nel seguito, questa maggiore sensibilità delle memorie dà luogo a nuovi modelli di guasto, in parte diversi da quelli visti nel caso di logiche realizzate mediante *gate* che sono immediatamente riconducibili a diffetti di tipo

fisico nella struttura del circuito. [3]

La rappresentazione schematica di Fig. 1.25 consente una prima classificazione dei modelli di guasto come riguardanti:

a) la logica di indirizzamento;

b) la matrice di celle;

c) la logica di lettura e scrittura;

Fra questi, chiaramente i più interessanti sono quelli della matrice di memoria, anche perchè è stato dimostrato [30] che quelli nella logica di indirizzamento o nella logica lettura/scrittura possono essere ricondotti a guasti (possibilmente) multipli interni alla matrice di memoria.

Nell'esaminare questi tipi di guasto nel seguito, si farà riferimento ad una memoria statica [31, 32] per poi discutere le principali differenze col caso delle DRAM.

a) *guasti nel decoder*: La logica di indirizzamento è tipicamente combinatoria; pertanto, può essere trattata mediante i modelli di guasto visti in precedenza (*stuck-at*, *bridging* ...). Nel collaudo delle memorie, però, e sono di particolare interesse gli effetti sull'indirizzamento di guasti interni al *decoder*, classificati come:

- con un dato indirizzo nessuna cella viene indirizzata;
- con un dato indirizzo si accede a più celle contemporaneamente (creando così condizioni di conflitto elettrico sulle bit-line;
- una certa cella può essere accessibile con più indirizzi;

b) *guasti nelle celle di memoria*: All'interno della matrice di celle, si possono avere diverse tipi di guasti, alcuni dei quali sono descrivibili mediante modelli di tipo tradizionale, altri sono peculiari delle memorie. Tra i guasti delle celle meritano un cenno particolare:

- *stuck-at*: una cella è permanentemente bloccata ai valori logici 0 e 1;
- *stuck-open*: a causa di un break o di un transistore bloccato spento, la cella rimane isolata elettricamente dalla bit-line; la rivelabilità del guasto dipende dalla struttura e dalla tecnologia del *sense-amplifier* in maniera relativamente complessa, perchè il guasto può essere rivelato come uno *stuck-at*, oppure possono anche entrare in gioco problemi di ritenzione da parte della circuiteria di uscita del dato letto precedentemente;

[3] Proprio a causa della minore affidabilità delle memorie rispetto ad altri tipi di circuiti digitali, si introducono ridondanze nell'informazione memorizzata, mediante l'utilizzo di codici a correzione o rivelazione d'errore.

- *transition fault*: in fase di scrittura la cella commuta lentamente in modo da non rendere disponibile il nuovo dato a una successiva lettura;
- *data retention fault*: la cella guasta non è in grado di conservare il suo valore logico per un tempo adeguatamente lungo;
- *coupled state fault*: una cella (A) può essere accoppiata (a causa di fenomeni capacitivi o *bridging*) A una delle celle adiacenti (B). In questo caso, un'operazione di scrittura su B può dare luogo luogo a un cambiamento di stato di A. In particolare, se lo stato di A segue quello di B, si parla di accoppiamento idempotente, se invece A ha una transizione opposta a B si parla di accoppiamento invertente.

c) *guasti nella logica di lettura e scrittura*: In pratica si tratta di guasti nei bus dati nei *sense amplifier* o nei *buffer* di scrittura. Questi possono risultare in *stuck-at* (singoli o multipli) sulle uscite, in guasti *stuck-open* sulle uscite stess (che esibiscono un comportamento in cui il guasto si può manifestare o meno dipendentemente dallo stato assunto in precedenza dal circuito); oppure possono aversi accoppiamenti fra i diversi bit.

Per quello che riguarda le memorie dinamiche, i modelli di guasto, sono molto simili a quelli descritti in precedenza; il *data retention fault*, in questo caso criticamente dipendente dal *leakage* dei dispositivi nonchè da possibili guasti nel ciclo di *refresh*, è comunque di particolare importanza. In questo tipo di circuiti occorre ricordare l'importanza di guasti di tipo intermittente o transiente.

Al riguardo della notevole molteplicità di possibili modelli di guasto occorre infine rivelare che gli algoritmi utilizzati per ricavare i vettori di collaudo da applicare alle memorie non sono orientati alla rivelazione di specifici guasti. Essi, infatti, sfruttando la regolarità del circuito, consistono in una serie di operazioni di lettura e scrittura delle diverse celle della memoria con valori ottenuti sulla base di considerazioni funzionali (o euristiche), che tentano comunque di garantire una buona copertura sui guasti fino ad ora elencati.

Bibliografia

[1] J. Galiay, Y. Crouzet, and M. Vergniault, "Physical Versus Logical Fault Models in MOS LSI Circuits: Impact on Their Testability", *IEEE Transaction on Computers*, vol. 29, pp. 527 – 531, 1980.

[2] C. Timoc and et al., "Logical Model for Physical Failures", in *Proc. of IEEE Int. Test Conf.*, pp. 34 – 41, 1983.

[3] J. A. Abraham and W. K. Fuchs, "Fault and Error Models for VLSI", *Proc. of the IEEE*, vol. 74, pp. 639 – 654, 1986.

[4] P. Banerjee and J. Abraham, "Characterization and Testing of Physical Failures in MOS Logic Circuits", *IEEE Design & Test*, vol. 1, pp. 76 – 86, Aug 1984.

[5] J. Shen, W. Maly, and F. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits", *IEEE Design & Test*, pp. 33 – 26, Dec. 1985.

[6] W. Maly, "Realistic Fault Modeling for VLSI Testing", in *Proc. of Design Automation Conf.*, pp. 173 – 180, 1987.

[7] J. M. Soden and C. F. Hawkins, "Test Considerations for Gate Oxide Shorts in CMOS ICs", *IEEE Design & Test*, vol. 3, pp. 56 – 63, Aug 1986.

[8] K. Mei, "Bridging and Stuck-at Faults", *IEEE Transaction on Computers*, vol. C-23, pp. 720 – 727, 1974.

[9] M. Abramovici and P. R. Menon, "A Practical Approach to Fault Simulation and Test Generation for Bridging Faults", *IEEE Transaction on Computers*, vol. C-34, pp. 658 – 663, Aug 1985.

[10] S. Millman and E. M. Cluskey, "Detecting Bridging Faults with Stuck-at Test Sets", in *Proc. of IEEE Int. Test Conf.*, pp. 773 – 778, 1988.

[11] R. Rajsuman, Y. Malaiya, and A. Jayasumana, "On Accuracy of Switch-Level Modeling of Bridging Faults in Complex Gates", in *Proc. of Design Automation Conf.*, pp. 244 – 250, 1987.

[12] F. Ferguson, M.Taylor, and T. Larrabee, "Testing for Parametric Faults in Static CMOS Circuits", in *Proc. of IEEE Int. Test Conf.*, pp. 436 – 443, 1990.

[13] T. Storey and W. Maly, "CMOS Bridging Fault Detection", in *Proc. of IEEE Int. Test Conf.*, pp. 842 – 851, 1990.

[14] H. Hao and E. McCluskey, ""Resistive Shorts" within CMOS Gates", in *Proc. of IEEE Int. Test Conf.*, pp. 292 – 301, 1991.

[15] M. Favalli, P. Olivo, M. Damiani, and B. Riccò, "Fault Simulation of Unconventional Faults in CMOS ICs", *IEEE Transaction on CAD*, vol. 10, pp. 677 – 682, 1991.

[16] M. Favalli, P. Olivo, and B. Riccò, "Dynamic Effects in the Detection of Bridging Faults in CMOS ICs", *Jou. of Electronic Testing: Theory and Application*, vol. 3, pp. 197 – 205, 1992.

[17] Y. K. Malaiya and S. Y. H. Su, "A New Fault Model and Testing Technique for CMOS Devices", in *Proc. of IEEE Int. Test Conf.*, pp. 25 – 34, 1982.

[18] D. Feltham, P. Nigh, R. Carley, and W. Maly, "Current Sensing for Built-In Testing of CMOS circuits", in *Proc. of IEEE Int. Conf. On Computer Design*, pp. 454 – 457, 1988.

[19] L. Horning, J. Soden, R. Fritzmeier, and C. Hawkins, "Measurements of Quiescent Power Supply Current for CMOS ICs in Production Testing", in *Proc. of IEEE Int. Test Conf.*, pp. 300 – 309, 1987.

[20] C. Crapuchettes, "Testing CMOS i_{DD} on Large Devices", in *Proc. of IEEE Int. Test Conf.*, pp. 310 – 315, 1987.

[21] C. Hawkins, J. Soden, R. Fritzemeier, and L. Horning, "Quiescent Power Supply Current Measurement for CMOS IC Defect Detection", *IEEE Trans. on Industrial Electronics*, vol. 36, pp. 211 – 218, 1989.

[22] S. Reddy, V. Agrawal, and M. Reddy, "Robust Tests for Stuck-Open Faults in CMOS Combinational Logic Circuits", in *Proc. of Int. Fault Tolerant Comp. Symp.*, pp. 44 – 49, 1984.

[23] S. Reddy and M. Reddy, "Testable Realization for CMOS Stuck-Open Faults in CMOS Combinational Logic Circuits", *IEEE Transaction on Computers*, vol. C-35, pp. 742 – 754, Aug 1986.

[24] M. Favalli, P. Olivo, F. Somenzi, and B. Riccò, "Fault Simulation for General FCMOS ICs", *Jou. of Electronic Testing: Theory and Application*, vol. 2, pp. 181 – 190, 1991.

[25] D. Osatpko and S. Hong, "Fault Analysis and Test Generation for Programmable Logic Arrays (PLA's)", *IEEE Transaction on Computers*, vol. 28, pp. 617 – 626, 1979.

[26] J. Smith, "Detection of Faults in Programmable Logic Arrays", *IEEE Transaction on Computers*, vol. 28, pp. 845 – 853, 1979.

[27] F. Somenzi and S. Gai, "Fault Detection in Programmable Logic Arrays", *Proc. of the IEEE*, vol. 74, pp. 655 – 668, 1986.

[28] M. Ligthart and R. Stans, "A Fault Model for PLAs", in *Proc. of IEEE Eur. Test Conf.*, pp. 252 – 260, 1989.

[29] A. J. V. de Goor, *Testing Semiconductor Memories, Theory and Practice.* Wiley & Sons, 1991.

[30] R. N. nd S. Thatte and J. Abraham, "Efficient algorithms for testing semiconductor random-access memories", *IEEE Transaction on Computers*, vol. 27, pp. 572 – 576, 1978.

[31] F. B. R. Dekker and L. Thjssen, "A Realistic Fault Model and Test Algorithms for Static Random Access Memories", *IEEE Transaction on CAD*, vol. 9, pp. 567 – 562, 1990.

[32] A. J. V. D. Goor, "Using march tests to test SRAMs", *IEEE Design & Test*, pp. 8 – 14, March 1993.