Testbarer Schaltungsentwurf : eine Übersicht

Autor(en): **Jacomet, M. / Reber, A.**

Objekttyp: **Article**

Zeitschrift: **Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association Suisse des Electriciens, de l'Association des Entreprises électriques suisses**

Band (Jahr): **78 (1987)**

Heft 11

PDF erstellt am: **22.07.2024**

Persistenter Link: https://doi.org/10.5169/seals-903873

Testbarer Schaltungsentwurf – eine Übersicht

M. Jacomet und A. Reber

Mit dem Trend zu immer höheren Gatterzahlen wird das Austesten integrierter Bausteine zunehmend komplexer. Im Gegensatz zu SSI- und MSI-Schaltungen werden bei LSI- und VLSI-Schaltungen ganze Systeme integriert, die nur über wenige Pins kontrolliert und beobachtet werden können. Am Beispiel des durch das Mealy-Modell beschriebenen endlichen Automaten werden die wichtigsten Methoden für den Entwurf testbarer Schaltungen besprochen und verglichen.

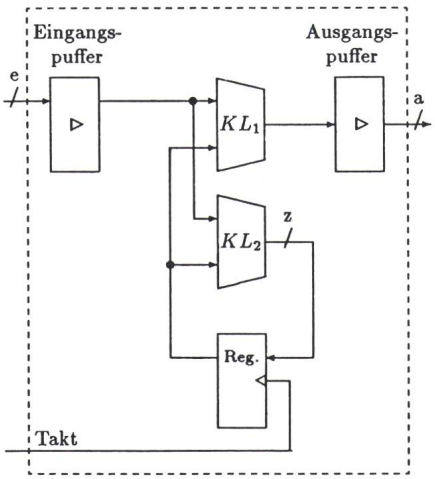
Le test des circuits intégrés devient de plus en plus compliqué en raison de nombre croissant de portes. Contrairement à ceux à petite ou moyenne échelle, les circuits à grande ou très grande échelle, avec systèmes entiers intégrés, ne peuvent être contrôlés et observés que par peu de broches. A l'exemple des automates finis, décrits dans le modèle Mealy, les méthodes les plus importantes pour le développement de circuits contrôlables sont discutées et comparées.

Adresse der Autoren

Marcel Jacomet, dipl. El.-Ing. ETH, Institut für Elektronik, ETH Zentrum, 8092 Zürich, und *Andreas Reber*, dipl. El.-Ing. ETH, HMT microelectronic AG, 2555 Brügg b. Biel.

1. Einführung

Jedes digitale System, das synchron betrieben wird, ist ein endlicher Automat mit einem endlichen Eingangs-, Ausgangs- und Zustandsalphabet. Solche Automaten können mit Hilfe eines Mealy- oder Moore-Modells beschrieben werden (Fig. 1). Die Aufgabe des Tests ist, das Zustands- und Ausgangsalphabet mit Hilfe des Eingangsalphabetes zu überprüfen. Die Ein- und Ausgangsalphabete können leicht über die Signalleitungen kontrolliert bzw. beobachtet werden. Kann der Zustandsspeicher ausserdem durch externe Steuerung in einen definierten Zustand gebracht werden, so ist auch das Zustandsalphabet überprüfbar, da dieses durch das Eingangsalphabet kontrolliert und indirekt über das Ausgangsalphabet beobachtet werden kann.



Figur 1 Mealy-Modell des endlichen Automaten

Jeder endliche Automat kann in die einfache Form des Mealy-Modells transformiert werden, bei dem Speicher (Reg.) und Verknüpfungsoperationen sauber voneinander getrennt sind.

KL_1, KL_2 kombinatorische Blöcke
Reg. Register
a, e, z Dimensionen des Eingangs-, Ausgangs- und Zustandsvektors

Der Test baut auf der Fähigkeit der *Kontrollierbarkeit* (*Kb*) und der *Beobachtbarkeit* (*Bb*) auf. Je nachdem, ob nur eine *Fehlererkennung* (Go-/No-go-Test) oder ob eine *Fehlerdiagnose* durch den Test erbracht werden soll, muss er verschieden aufgebaut sein. Damit durch hohe Kontrollier- und Beobachtbarkeit automatisch eine gute *Testbarkeit* (*Tb*) erzeugt wird, muss sämtliche Redundanz (wenigstens für den Test-Mode) eliminiert sein. In [1] wird anhand einiger Beispiele gezeigt, dass bei redundanten Schaltungen die Testbarkeit sehr stark abnehmen kann.

Die Testproblematik scheint also auf den ersten Blick gar nicht so gravierend zu sein. Bei genauerem Betrachten fällt hingegen auf, dass erstens das Eingangsalphabet nicht immer bestimmbar ist, beispielsweise im Falle von Mikroprozessoren, für welche das Alphabet aus der Menge aller Programm- und Datensätze besteht, die je einmal von diesem Prozessor verarbeitet werden müssen. Zweitens werden für ein kombinatorisches Netzwerk mit einer Eingangswortbreite e theoretisch 2^e Testvektoren für einen kompletten Test benötigt. Besitzt das Netzwerk zusätzlich einen Zustandsspeicher mit einer Wortbreite z, so steigt die Anzahl der Testvektoren, die einen logischen Test aller internen Gatter erlauben, mindestens auf $2^{(e+z)}$ (alle Kombinationen des Eingangsalphabets mal alle Kombinationen des Zustandsalphabets). Drittens wird der physikalischen Natur der möglichen Fehler, die bedingt sind durch Technologie, Topologie, Lay-Out-Regeln und Aufbau der einzelnen Gatter, keine Beachtung geschenkt.

Das klassische *Single-Stuck-at*-Fehlermodell, bei dem angenommen wird, dass alle Fehler sich dermassen auswirken, als ob ein Torein- oder -ausgang fest auf logisch Eins (stuck-at-1) oder fest auf logisch Null (stuck-at-0)

hängenbleibt, genügt nicht immer allen Anforderungen [2; 3]. Als Erweiterung finden zusätzliche Fehlermodelle Verwendung, welche das Verhalten der Transistoren und der Verbindungen untereinander miteinbeziehen. Dabei wird angenommen, dass ein Transistor immer leitend (stuck on), immer ausgeschaltet (stuck off), die Verbindung zwischen Transistoren immer offen (stuck open) ist oder einen Kurzschluss (stuck short) aufweist[1]. Diese Modellierung basiert auf Untersuchungen an defekten Bausteinen [4], die zeigen, dass Kurzschlüsse zwischen metallischen Leitern zu nahezu 40%, Unterbrüche der metallischen Leiter und Kurzschlüsse zwischen Diffusionen zu je 14% und Unterbrüche in den Diffusionen zu etwa 6% für die Fehler verantwortlich sind. Bei etwa 10% der Fehler konnte keine physikalische Ursache gefunden werden. Die restlichen Fälle wiesen grobe Fehler auf (Kratzer über den Chip usw.), so dass sie nicht in einzelne Klassen physikalischer Fehlerursachen untergebracht werden konnten. Für die Fehlermodellierung sind diese Defekte unwesentlich, weil sie mit jedem Fehlermodell entdeckt werden können.

Bei den meisten Fehlermodellen wird angenommen, dass sich immer nur ein Fehler zum betrachteten Zeitpunkt bemerkbar macht. Dies ist einschränkend und muss durch entsprechende Modifikation des Eingangsalphabets korrigiert werden.

2. Testvektorgenerierung

Ist eine integrierte Schaltung produziert worden, so muss sie auf mögliche Fabrikationsfehler untersucht werden. Der Wafer oder die verpackten Chips werden auf den Testautomaten montiert, logische Signale (Testvektoren) in die Chip-Eingänge eingespeist und deren Auswirkung an den Ausgängen beobachtet. Das Generieren von Testvektoren für VLSI-Schaltungen ist ein zeitintensiver und kostspieliger Schritt beim Entwurf einer Schaltung. Es wurden deshalb Softwarepakete, sogenannte automatische Test-Pattern-Generatoren (ATPG) entwickelt. Die Güte der Testvektoren wird mit verschiedenen Messverfahren gemessen. In diesem Abschnitt werden häufig benutzte Algorithmen zur Testvektorgenerierung und Fehlersimulation erläutert.

2.1 Ad-hoc-Testvektorgenerierung

Bei kleinen Schaltungen können die Testvektoren von Hand erzeugt werden. Aber bereits bei Schaltungen mit einer Komplexität von 100 bis 300 Gattern ist man auf eine automatische Testvektorgenerierung angewiesen.

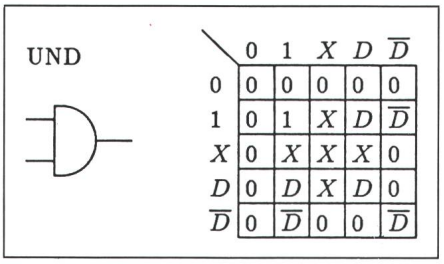
2.2 Pseudo-Random-Testvektorgenerierung

Die sogenannte Pseudo-Random-Testvektorgenerierung wird hauptsächlich in zwei Fällen angewendet: bei einfachen Softwarealgorithmen sowie beim Built-in-Test (Spezialhardware zur Testvektorgenerierung, welche im Testaufbau oder auf dem Chip mitintegriert wird) [5]. In beiden Fällen gewinnt man das Ausgangsalphabet und damit die richtige Übertragungsfunktion entweder durch eine Simultation oder durch das Beobachten einiger «fehlerfreier»[2] Schaltungen. Beim Built-in-Test werden dessen Ausgangsvektoren häufig noch auf dem Chip komprimiert. Das resultierende Datensignal wird als Signatur der Ausgangsvektoren bezeichnet.

2.3 Algorithmische Testvektorgenerierung für kombinatorische Schaltungen

Die algorithmische Testvektorgenerierung basiert auf einem Fehlermodell, dem logischen Netzwerk, sowie auf den darin enthaltenen Elementtypen.

Die meisten algorithmischen ATPG-Programme lassen sich in drei Phasen aufteilen. Anhand eines Beispieles sollen diese Schritte mittels des Single-Stuck-at-Fehlermodells illustriert werden. Um die Stuck-at-Fehler und die Testvektorgenerierung eines einfachen logischen Netzwerkes beschreiben zu können, ist eine Notation von Vorteil, welche nicht nur die logischen Zustände «0», «1» und «X» (don't care) bezeichnet, sondern auch die fehlerfreien und fehlerhaften Zustände der Schaltung. Dazu eignet sich



UND	0	1	X	D	$\overline{D}$
0	0	0	0	0	0
1	0	1	X	D	$\overline{D}$
X	0	X	X	X	0
D	0	D	X	D	0
$\overline{D}$	0	$\overline{D}$	0	0	$\overline{D}$

Figur 2 Wahrheitstabelle eines UND-Tores mit D-Notation

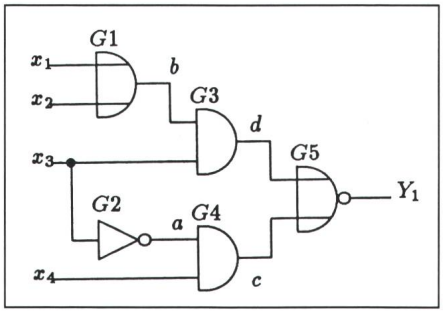
X Don't care
$D, \overline{D}$ s. Text

die sogenannte D-Notation (Fig. 2). Dabei wird dem Signal x_i ein imaginäres D-Signal «mitgegeben», das durch eventuelle Schaltungsfehler nicht beeinflusst wird. Am Eingang wird D gesetzt, falls $x_i = 1$, und $\overline{D}$ falls $x_i = 0$ ist. Ein Fehler manifestiert sich nun auf irgendeinem Zweig b_j dadurch, dass dort die Kombination $x_j = 1$ und $\overline{D}$ oder die Kombination $x_j = 0$ und D auftritt.

D-Algorithmus

Für den Single-Stuck-at-1-Fehler (s-a-1) am Knoten b in Figur 3 soll die Testvektorgenerierung anhand des D-Algorithmus aufgezeichnet werden.

Das Ziel der *Initialisierungsphase* ist, die Eingänge des Netzwerkes so zu beschalten, dass das zu den Stuck-at-Fehlern inverse logische Signal auf der fehlerhaften Leitung erscheint. Um am Beispiel von Figur 3 einen S-a-1-Fehler am Gatterausgang $G1$ zu entdekken, müssen somit die Eingänge x_1 und x_2 auf logisch 0 gesetzt werden. Dem Ausgang von Gatter $G1$ wird somit die Variable $\overline{D}$ zugewiesen. (Zu diesem Zeitpunkt weiss man noch nicht, ob am Knoten tatsächlich ein S-a-1-Fehler vorhanden ist.)



Figur 3 Beispiel eines logischen Netzwerkes für die Testvektorgenerierung

Bei einem Stuck-at-1-Fehler am Ausgang von G_1 antwortet der Ausgang Y_1 auf den Testvektor $\boldsymbol{x} = (0, 0, 1, \boldsymbol{x})$ mit einer 0 statt mit einer 1.

[1] In diesem Heft behandelt der Aufsatz von *J.-F. Perotto* «Test des circuits logiques combinatoires CMOS» ein solch komplexeres Fehlermodell.

[2] Gemeint sind Schaltungen, die sich gleich verhalten und deshalb als fehlerfrei klassiert werden können.

In der *Ausbreitungsphase* wird für den (potentiellen) Fehler ein Weg zu einem Ausgang gesucht und die Signale dem Pfad entlang für die Ausbreitung des Fehlers entsprechend spezifiziert. Der Pfad für den Fehler in Figur 3 geht über Gatter *G3* nach Gatter *G5* zum Ausgang Y_1. Dazu muss der Eingang x_3 auf 1 und die Leitung c auf 0 gesetzt werden. Die Ausgänge der Gatter *G3* und *G5* nehmen die Zustände $\overline{D}$ und D an. Nimmt der Ausgang Y_1 den logischen Wert 0 an, so ist gemäss der D-Notation der Single-Stuck-at-1-Fehler am Knoten b vorhanden.

In der *Justierungsphase* müssen die in der Ausbreitungsphase gesetzten Signale auf die primären Eingänge zurückgeführt werden. Dazu werden für die Leitung $c = 0$ die Eingänge $x_3 = 1$ oder $x_4 = 0$ gesetzt. Dem Eingang x_3 ist der logische Wert 1 schon in einem früheren Schritt zugeordnet worden, so dass x_4 einen beliebigen Wert annehmen darf.

Die beiden mit dem D-Algorithmus gewonnenen Testvektoren für den S-a-1-Fehler der Leitung b lauten somit:

$$\boldsymbol{x} = (x_1,x_2,x_3,x_4) = (0, 0, 1, 0)$$

$$\boldsymbol{x} = (x_1,x_2,x_3,x_4) = (0, 0, 1, 1)$$

Es existieren verschiedene Programme, welche auf dem Prinzip des D-Algorithmus beruhen. Eines der ältesten Programme, FALTGEN [6], kann für die Testvektorgenerierung von kleinen kombinatorischen Netzwerken (<300 Gatter) eingesetzt werden. Das bekanntere Programm PODEM [7] soll bei LSSD-Netzwerken (s. Kapitel 4.2) von bis zu 4000 Gattern Testvektoren mit einer Fehlerabdeckung von 90% in einer vernünftigen Rechenzeit erreichen. Eine Weiterentwicklung ist das Programm FAN [8]. Nach Angaben der Entwickler soll es etwa dreimal schneller als PODEM arbeiten.

2.4 Algebraische Testvektorgenerierung für kombinatorische Schaltungen

Auf algebraischem Wege lassen sich die Testvektoren mit Hilfe der Boolschen Ableitung berechnen (s. Fenster). Eine kombinatorische Schaltung realisiere die Funktion $f(\bar{x}) = f(x_1,x_2, \ldots x_n)$. Es seien die folgenden Abkürzungen eingeführt:

$$f_i(1) \equiv f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n) \quad (1)$$

$$f_i(0) \equiv f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n)$$

Die kombinatorische Schaltung $f(\bar{x}) = f(x_1,x_2, \ldots x_n)$ weise den Stuck-at-1-Fehler α am Eingang x_i auf. Die Schaltungsfunktion unter der Fehlerbedingung lautet somit:

$$f_\alpha(\boldsymbol{x}) = f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n) \quad (2)$$
$$= f_i(1)$$

Um einen Fehler entdecken zu können, müssen die beiden Übertragungsfunktionen $f(\boldsymbol{x})$ und $f_\alpha(\boldsymbol{x})$ verschiedene Werte liefern. Dies ist der Fall, wenn eine EXOR-Verknüpfung der beiden Funktionen 1 ergibt. Die Menge der Testvektoren, welche den Fehler α mit Signal x_i s-a-1 entdecken, ist somit bestimmt durch

$$T = f(\boldsymbol{x}) \oplus f_\alpha(\boldsymbol{x}) \quad (3)$$
$$= f(\boldsymbol{x})\overline{f}_\alpha(\boldsymbol{x}) + \overline{f}(\boldsymbol{x})f_\alpha(\boldsymbol{x})$$

an der Stelle $T = 1$. Mit dem Erweiterungstheorem[3] und für $f_\alpha(x) = f_i(1)$ erhält man aus (3)

$$T = (x_i f_i(1) + \overline{x}_i f_i(0)) \oplus f_i(1) \quad (4)$$
$$= \overline{x}_i f_i(0)\overline{f}_i(1) + \overline{x}_i f_i(0) f_i(1)$$
$$= \overline{x}_i(f_i(0) \oplus f_i(1))$$

Der Faktor $f_i(0) \oplus f_i(1)$ in der obigen Gleichung wird als die partielle Boolesche Differenz von f nach x_i bezeichnet und lässt sich mit $\partial f/\partial x_i$ abkürzen. Die Menge aller Testvektoren, welche den Fehler $a(x_i$ s-a-1) entdecken, ist somit:

$$\overline{x}_i \frac{\partial f}{\partial x_i} \quad (5a)$$

Entsprechend kann die Menge der Testvektoren für den Fehler x_i s-a-0 angegeben werden:

$$x_i \frac{\partial f}{\partial x_i} \quad (5b)$$

Im Beispiel der Figur 3 lässt sich nun der S-a-1-Fehler am Knoten b folgendermassen algebraisch berechnen:

Mit der Abkürzung $b = (x_1+x_2)$ schreibt sich die Funktion der Schaltung:

Die Boolesche Differenz

Die Änderung, welche eine Funktion

$$f(\boldsymbol{x}) = f(x_1,\ldots,x_i,\ldots,x_n) \quad (f1)$$

erfährt, wenn eine seiner Variablen x_i invertiert wird, kann mit Hilfe der Exclusiv-ODER-Funktion aus den beiden Zuständen vor und nach der Inversion berechnet werden. Das Resultat kann nur die beiden Werte 1 (Änderung) und 0 (keine Änderung) annehmen. In Anlehnung an die Ableitung der Analysis bezeichnet man diese Operation als Boolesche Differenz und schreibt

$$\frac{\partial f(\boldsymbol{x})}{\partial x_i} \overset{\text{def}}{=} f(x_1,\ldots,x_i,\ldots,x_n) \oplus f(x_1,\ldots,\overline{x}_i,\ldots,x_n) \quad (f2)$$

Mit $x_i = 0$ und

$$f_i(0) \equiv f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n) \quad (f3)$$

$$f_i(1) \equiv f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n) \quad (f4)$$

lässt sich vereinfacht schreiben

$$\frac{\partial f(\boldsymbol{x})}{\partial x_i} = f_i(0) \oplus f_i(1) \quad (f5)$$

Da die Boolesche Algebra kein Vorzeichen kennt, gilt die Identität

$$\frac{\partial \overline{f(\boldsymbol{x})}}{\partial x_i} = \frac{\partial f(\boldsymbol{x})}{\partial x_i} \quad (f6)$$

[3] Eine Funktion kann um eines oder um mehrere ihrer Argumente erweitert werden. Die Erweiterung der Funktion f um ihr Argument x_i lautet $f(x_1, \ldots, x_i, \ldots, x_n) = x_i f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) + \overline{x}_i f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$

$$\overline{f} = bx_3 + \overline{x}_3 x_4 \qquad (6)$$

daraus folgt mit der Identität $\partial\overline{f}/\partial x = \partial f/\partial x$:

$$\frac{\partial f}{\partial b} = x_3 \qquad (7)$$

Der S-a-1-Fehler auf der Leitung b findet sich mit jedem Vektor aus der Menge:

$$\overline{b} \cdot \frac{\partial f}{\partial b} = \overline{(x_1 + x_2)} \cdot x_3 = \overline{x}_1 \overline{x}_2 x_3 \qquad (8)$$

Erwartungsgemäss führt dies zu demselben Resultat, wie bei der Herleitung mit dem *D*-Algorithmus. Die Methode der Booleschen Ableitung hat allerdings nur einen akademischen Wert, da sie bei komplexeren Schaltungen zu aufwendig wird.

2.5 Fehlersimulation

Neben der normalen Simulation von Netzwerken ist es oft wünschenswert, eine Simulation unter Fehlerbedingungen durchzuführen. Die Gründe dafür können sein:

- Bewertung einer Testvektorsequenz im Hinblick auf deren Fähigkeit, Fehler zu entdecken oder zu lokalisieren,
- Untersuchung des Verhaltens einer Schaltung unter Fehlerbedingungen.

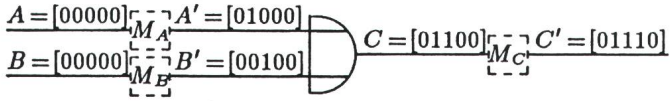
Fehler können in einer Schaltung ein unerwartetes Verhalten bewirken. Zum Beispiel können Hazards[4] auftreten, kombinatorische Schaltungen können ein sequentielles Verhalten aufweisen (insbesondere bei CMOS-Schaltungen), synchrone Schaltungen können asynchron werden, die Initialisierung einer Schaltung kann verhindert werden usw. Um eine zuverlässige Fehlersimulation zu erhalten, ist ein wirklichkeitsgetreues Simulationsmodell unerlässlich. Neben dem häufig verwendeten Stuck-at-Fehlermodell kann je nach Technologie auch das Verhalten der Schaltung unter anderen Fehlerbedingungen (Kurzschlüsse, Unterbrüche, Mehrfachfehler) nötig sein.

[4] Hazards sind dynamische Wertänderungen in Schaltungen, die auf unterschiedliche Signalverzögerungen zurückzuführen sind.

Figur 4
Prinzip der parallelen Fehlersimulation
Die Maske markiert die Bitposition, bei welcher der Fehlerwert einzusetzen ist.
A, B Eingangsvektoren

Bit Position	Fehler
1	fehlerfrei
2	A s-a-1
3	B s-a-1
4	C s-a-1
5	C s-a-0

Maske	Fehlerwert
M_A=[01000]	[01000]
M_B=[00100]	[00100]
M_C=[00011]	[00010]



Die älteste Fehlersimulationsmethode ändert die Schaltung dem Fehler entsprechend ab und simuliert diese neue Struktur. Da diese Methode äusserst zeitintensiv sein kann, haben sich weitere Fehlersimulationsmethoden eingebürgert. Sie lassen sich hauptsächlich in drei Kategorien einteilen:

- parallele (parallel) Fehlersimulationen [9],
- gleichzeitige (concurrent) Fehlersimulationen [10],
- herleitende (deductive) Fehlersimulation [11].

Als Beispiel sei die Arbeitsweise eines parallelen Fehlersimulators skizziert. Will man verschiedene Fehler simulieren, so lässt sich diese Aufgabe immer in n Probleme unterteilen, welche parallel simuliert werden können. Bei dieser Simulationsmethode wird die Tatsache ausgenutzt, dass die Verarbeitung von 1-bit und n-bit breiten Worten mit Computern einen ähnlich grossen Zeitaufwand benötigen. Die Fehlersimulation wird deshalb prinzipiell mit Hilfe n-Bit breiter Worte durchgeführt. Jede Bitposition dieser Worte repräsentiert ein Problem (Fehlerzustand), welches von den anderen zwar völlig unabhängig ist, aber trotzdem parallel mit den anderen simuliert werden kann. Die zu simulierenden Fehler werden in Masken (M) hineininjiziert, welche die Worte an den entsprechenden Bitstellen verändern. Damit können mehrere Fehler parallel verarbeitet werden. In Figur 4 ist gezeigt, wie die Eingangsworte A und B durch die Masken M_A und M_B in die Worte A' und B' transformiert werden. Das entsprechende gilt für das Wort C. Die Bitpositionen 2 bis 4 des Wortes C' weichen von der fehlerfreien ersten Bitposition ab, was eine Erkennung der drei entsprechenden Fehler zulässt.

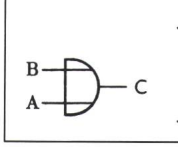
2.6 Messung der Testbarkeit

Es gibt verschiedene Möglichkeiten, die Testbarkeit einer Schaltung zu charakterisieren, ohne zuerst die Testvektoren generiert zu haben.

- Bei der *Bewertungsmethode* werden die einzelnen Schaltungselemente nach deren Eigenschaften bezüglich der Testbarkeit bewertet. Fehlt beispielsweise einer Schaltung die Initialisierungsmöglichkeit (Kontrollierbarkeit!), so stellt dies eine gravierende Verschlechterung der Testbarkeit dar. Sind lineare Schieberegister vorhanden, so wird hingegen die Testbarkeit verbessert. Der Vorteil dieser Bewertungsmethode liegt in ihrer Einfachheit, der Nachteil in ihrer Aussagekraft.
- Die *algorithmischen Methoden* quantifizieren die Testbarkeit jedes einzelnen Schaltungsknotens und erlauben somit das Auffinden von schlecht testbaren Schaltungseinheiten. Die Testbarkeit (Tb) kann als Funktion der Kontrollierbarkeit (Kb) und der Beobachtbarkeit (Bb) gesehen werden. Im CAMELOT-Projekt wird die Testbarkeit definiert als $Tb = Kb \cdot Bb$, wobei die Kontrollierbarkeit und Beobachtbarkeit wiederum aus einer Funktion verschiedener Einflussgrössen erhalten werden [12; 13]. Weitere Ansätze algorithmischer Methoden für die Testbarkeitsmessung sind SCOAP [14], TEST/80 [15], ENTROPY [16], TESTSCREEN [17] und VICTOR [18], um nur einige zu nennen.

3. Reduktionsverfahren

Sucht man beispielsweise die Testvektoren aller Stuck-at-Fehler für ein einfaches ODER-Gatter mit den Eingängen A, B und dem Ausgang C, so stellt man fest, dass jeweils mehrere Fehler von demselben Testvektor ent-



A	B	Fehlerklassen
0	0	$a/1 \Leftrightarrow b/1 \Leftrightarrow c/1$
0	1	$b/0 \Rightarrow c/0$
1	0	$a/0 \Rightarrow c/0$
1	1	$c/0$

Figur 5 Klassen der Stuck-at-Fehler eines ODER-Tores

a, b, c Signale an den Anschlüssen A, B, C
$a/1$ Signal a stuck-at-1
$\Leftrightarrow$ äquivalente Fehler
$\Rightarrow$ dominante Fehler

deckt werden können (Fig. 5). Dieses sogenannte Zusammenfallen (Collapsing) von Fehlern kann zu systematischen Reduktionsverfahren führen. In einer ersten Näherung lässt sich das Fault Collapsing in drei Stufen unterteilen:

- Eine erste Stufe, die sich den äquivalenten Fehlern widmet (Fault Equivalence).
- Eine zweite Stufe, die dominante Fehler aufdeckt (Fault Dominance).
- Eine dritte Stufe, welche die Beziehung zwischen Einzel- und Mehrfach-Fehler betrachtet (Single Faults, Multiple Faults).

3.1 Äquivalente Fehler

Bei einer kombinatorischen Schaltung, welche die Funktion f realisiert, stellt $T_\alpha = f \oplus f_\alpha$ die Menge der Testvektoren dar, welche den Fehler α aufdeckt. Der Fehler β wird entsprechend durch die $T_\beta = f \oplus f_\beta$ beschrieben. Die Testvektormenge, welche die beiden Fehler α und β unterscheiden kann, ist somit durch $f_\alpha \oplus f_\beta$ definiert. Falls $f_\alpha = f_\beta$ ist, so existieren keine Testvektoren, welche die beiden Fehler voneinander unterscheiden können. Solche Fehler werden als äquivalent bezeichnet, und jeder Testvektor, welcher den Fehler α entdeckt, findet ebenso den Fehler β und umgekehrt. Somit ist eine Fehlerdiagnose für diese Fehler, nicht aber eine Fehleranalyse möglich. Bei der Testvektorgenerierung muss nur einer von allen äquivalenten Fehlern betrachtet werden.

3.2 Dominante Fehler

Ist man nur an der Fehlerdiagnose interessiert, so lässt sich mit der oben erwähnten zweiten Stufe eine weitere Fehlerreduktion durchführen. Die Menge der Testvektoren T_α für den Fehler α dominiert den Fehler β, falls T_β in T_α enthalten ist ($T_\beta \subset T_\alpha$). Jeder Testvektor für den Fehler β entdeckt ebenso den Fehler α, aber nicht umgekehrt. Für die Testvektorgenerierung muss somit nur der Fehler β berücksichtigt werden (Fig. 5).

3.3 Mehrfachfehler

Bis jetzt wurde immer von der Voraussetzung ausgegangen, dass nur ein Fehler pro Schaltung (Single Fault) auftreten kann. Werden Mehrfachfehler (Multiple Faults) betrachtet, so können neben dem exponentiellen Anwachsen der Komplexität noch weitere Probleme auftreten. Eines davon stellt das Maskierungsphänomen (Fault Masking) dar. Zur Illustration stelle man sich ein UND-Gatter (Eingänge x_1 und x_2) mit den Fehlern α s-a-1 am Eingang x_1 vor. Der Testvektor $x = (0,1)$ deckt diesen Fehler zwar auf, tritt jedoch am Ausgang des UND-Gatters gleichzeitig ein s-a-0-Fehler β auf, so wird der Vektor $x = (0,1)$ zur Fehlererkennung nicht mehr nützlich sein. Man spricht in solch einem Fall von der Maskierung von α durch β. Dieses Verdecken kommt nicht mehr vor, falls der Test $x = (1,1)$ angewendet wird.

4. Testbarer Schaltungsentwurf

4.1 Elemente zur Untersuchung testbarer Strukturen

Damit eine Schaltung gut testbar ist, muss schon zu Beginn der Entwurfsphase die Teststrategie festgelegt werden. Je nach verwendeter Methode muss mehr oder weniger testspezifische Hardware eingebaut werden, d.h. Hardware, welche die Struktur der Schaltung, nicht aber deren logische Funktion beeinflusst. Bevor auf die eigentlichen Testmethoden eingegangen wird, werden kurz die wichtigsten Schaltungselemente beschrieben, welche heute zur Bildung testbarer Strukturen eingesetzt werden [19]:

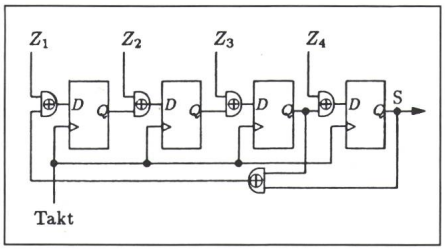
- Testpunkte (Test Points)
- Schieberegister (Shift Register)
- Linear rückgekoppelte Schieberegister (Linear Feedback Shift Register)
- Bilbo-Elemente (Built-in Logic Block Observation)

Mit den *Testpunkten* wird versucht, interne, schwer zu kontrollierende oder zu beobachtende Punkte über eine Schnittstelle zur Aussenwelt zu führen. Die Auswahl der Testpunkte erfolgt häufig heuristisch, weshalb diese Methode zu den Ad-hoc-Methoden gezählt werden kann. Sie eignet sich nur für kleine Komplexitäten und zeichnet sich meistens durch einen geringen zusätzlichen Hardwareaufwand sowie eine schlechte Fehlerabdeckung aus.

Mit *Schieberegistern* lassen sich Daten seriell, d.h. über eine möglichst geringe Anzahl Pins aus der integrierten Schaltung auslesen, sie können in zwei Modi betrieben werden. In einer ersten Betriebsart stellen sie ein Register mit einer Anzahl paralleler Ein- und Ausgänge dar. Mit Hilfe eines Mode-Signals kann das Element in ein Schieberegister übergeführt werden, bei dem die Daten seriell ein- bzw. ausgelesen werden können.

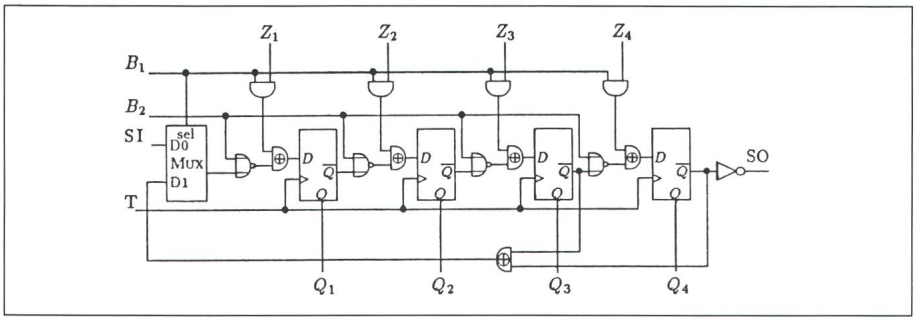
Ein *linear rückgekoppeltes Schieberegister (LFSR)* besteht aus einem gewöhnlichen Schieberegister, welches zusätzlich eine lineare[5] Rückkopplung aufweist. Ein solches Element eignet sich sowohl für die Komprimierung von Daten (Signaturanalyse) als auch für schaltungsinterne Erzeugung von Testvektoren (Pseudo Random). Bei der Verwendung des linear rückgekoppelten Schieberegisters für die Signaturanalyse können die zu komprimierenden Daten entweder seriell oder parallel (Fig. 6) eingelesen werden. Die Signatur, welche eine Funktion der komprimierten Daten ist, kann nach einer genügend grossen Anzahl Taktzyklen seriell ausgelesen werden. Wird das linear rückgekoppelte Schieberegister als Testvektorgenerator eingesetzt, so kann nach jedem Taktzyklus ein Testvektor an den Flip-Flop-Ausgängen abgelesen werden. Da die Schieberegister linear rückgekoppelt sind, erhält man eine maximal lange Folge von Vektoren (2^n mit n = Anzahl der Flip-Flops).

[5] Linearität in der Booleschen Algebra bedeutet, dass keine Variablenprodukte auftreten dürfen. Eine lineare Rückkopplung wird durch die ausschliessliche Verwendung von Exor-Gattern als Verknüpfungseinheiten erreicht.



Figur 6 Linear rückgekoppeltes 4-bit-Schieberegister für die Signaturanalyse

$Z_1, ..., Z_4$ Eingänge
S Signatur



Figur 7 4-bit-BILBO-Struktur

$Z_1, \ldots Z_4$ Eingänge
$Q_1, \ldots, Q_4$ Ausgänge
B_1, B_2 Steuereingänge
$B = (1,1)$ Normaler (Register-)Modus
$B = (0,0)$ Schieberegister-Modus
$B = (1,0)$ LFSR mit parallelen Eingängen (Testvektorerzeugung)
$B = (0,1)$ Reset Modus (Initialisierung der Eingänge)
SI Scan In
SO Scan Out
T Takt

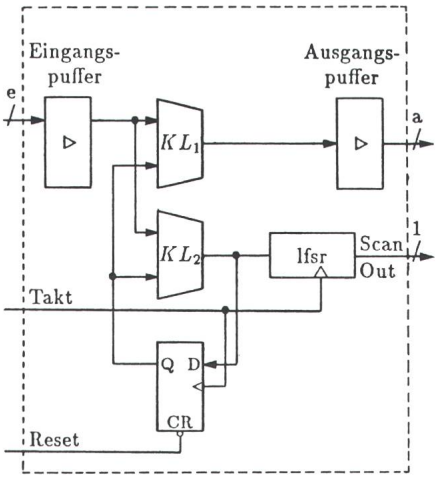
Das *Built-in Logic Block Observation*-Element (BILBO, Fig. 7) stellt ein multifunktionales Testelement dar, welches Funktionen verschiedener Baublöcke wie Register (Eingänge parallel), Schieberegister (Eingang seriell) und linear rückgekoppelte Schieberegister (Eingänge parallel) erfüllen kann, die über die Steuereingänge ***B*** angewählt werden können. Die Bilbo-Elemente eignen sich aufgrund ihrer universellen Funktionen gut als Teststrukturen und lassen sich in zahlreichen Konfigurationen einsetzen.

4.2 Testmethoden

Die Testmethoden lassen sich in einer ersten Annäherung in vier Klassen unterteilen:
- Ad-hoc-Methoden
- Scan-Methoden
- Strukturspezifische Methoden
- Selbsttest-Methoden

4.2.1 Ad-hoc-Methoden

Ad-hoc-Methoden werden so genannt, weil sie keine systematischen Entwurfsregeln kennen (fehlende Kontrollier- und Beobachtbarkeit). Zu ihnen werden z.B. Testpunktverfahren und teilweise auch die Signaturanalyse (bei fehlender Kontrollierbarkeit) gezählt. Bei der Signaturanalyse wird ein linear rückgekoppeltes Schieberegister für die Beobachtung der gewünschten Punkte verwendet, wobei an dessen Eingängen die Sequenz der zu beobachtenden Punkte eingespeist wird (Fig. 8). Neben Testpunktverfahren und Signaturanalyse wird oft das Aufteilen der Schaltung in Unterblöcke (Partitioning) zu den Ad-hoc-Methoden gezählt, da keine Regeln für das Aufteilen und den Entwurf der Unterblöcke existieren [20; 21]. Fehlerfreie Schaltungen im Testmodus zum Oszillieren zu bringen, stellt eine weitere, jedoch etwas rudimentäre Testmethode dar (Self-oscillating). Ein sinnvollerer Ansatz besteht im Vergleich von ähnlichen Schaltungsblöcken (Self-comparison). Dabei wird beim Entwurf der Schaltung darauf geachtet, dass sich Blöcke mit ähnlichen Funktionen unter Testbedingungen identisch verhalten. So können die Ausgänge mit Exor-Gatter verglichen und die Funktionen überprüft werden [22].



Figur 8 Anwendung des linear rückgekoppelten Schieberegisters als Signaturanalyseelement

e, a, 1 Dimensionen des Eingangs- und Ausgangsvektors sowie des seriellen Schieberegisterausgangs

4.2.2 Scan-Techniken

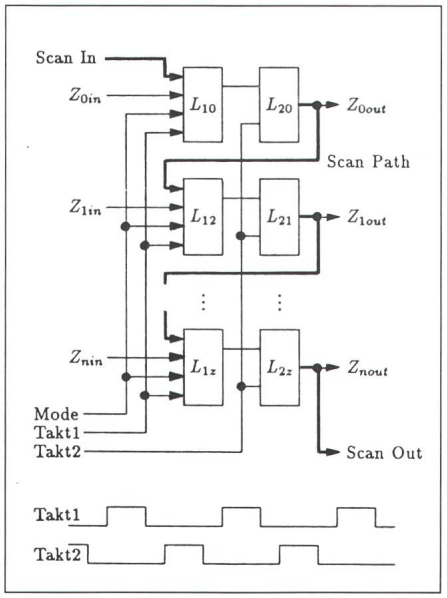
Bei den Scan-Techniken wird die hohe Kontrollier- und Beobachtbarkeit dadurch erreicht, dass der Zustandsvektor von aussen sowohl gesetzt (kontrolliert) als auch gelesen (beobachtet) werden kann. Je nach der Struktur des Zustandsspeichers werden drei Methoden unterschieden:

- LSSD (Level Sensitive Scan Design)
- Scan-Path (Edge Sensitive)
- Scan-Set (Edge Sensitive)

Die Unterschiede zwischen diesen Techniken resultieren vor allem aus der Verwendung verschiedener Speicherelemente.

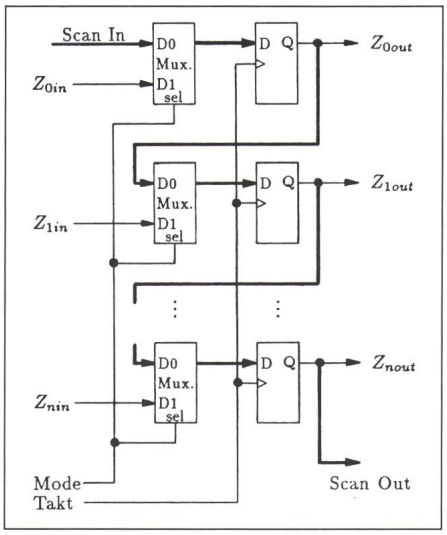
Bei der *LSSD-Methode* wird der einzelne Zustandsspeicher in zwei separat getaktete Latchs[6] aufgeteilt (Fig. 9). Da bei diesen die Eingangsdaten bei aktivem Pegel des Steuersignales an den Ausgang weitergeleitet werden, sind Latchs zustandssensitiv (Level Sensitive), während Flip-Flops, welche auf die Taktflanke schalten, als flankensensitiv (Edge Sensitive) bezeichnet werden.

[6] Ein Latch ist ein speicherndes Element, das bei aktivem Steuersignal (Takt) für die Daten transparent erscheint; der Ausgang folgt damit jeder Signaländerung am Eingang. Bei inaktivem Steuersignal bleibt der Zustand unabhängig vom Eingangssignal erhalten.



Figur 9 Organisation des Zustandspeichers für LSSD-Technik

$Z_{0in}, \ldots, Z_{nin}$ Ausgang der kombinatorischen Logik
$Z_{0out}, \ldots, Z_{nout}$ Zustandsvektor



Figur 10 Organisation des Zustandspeichers für Scan-Path-Technik

$Z_{0in}, ..., Z_{nin}$ Ausgang der kombinatorischen Logik
$Z_{0out}, ..., Z_{nout}$ Zustandsvektor

Bei der *Scan-Path-Methode* (Fig. 10) wird die einzelne Zelle des Zustandsspeichers (Flip-Flop) mit einem flankensensitiven Takt angesteuert. An den Eingängen der Speicherelemente werden über einen Multiplexer, je nach Modus, die Daten des Scan-Pfades oder des Ausganges der Zustandslogik eingespeist.

Das *Scan-Set-Verfahren* (Fig. 11) benötigt wie alle Scan-Techniken ein Schieberegister, das entweder parallel oder seriell geladen werden kann. Im Gegensatz zu den beiden obenerwähnten Verfahren liegt das Schieberegister aber ausserhalb der Datenflusspfade. Es dient dazu, die Werte einzelner Testpunkte sichtbar zu machen. Dabei wird das Schieberegister mit den Daten der Testpunkte parallel geladen (in Fig. 11 entsprechen diese gerade dem Zustandsvektor) und danach seriell ausgelesen. Um die Kontrollierbarkeit zu unterstützen, können mit den im Schieberegister eingegebenen Daten auch die Werte einzelner Speicherelemente gesetzt oder aber die Struktur der Schaltung für den Test beeinflusst werden[7].

[7] Die Idee, die Testantworten wie bei der Signaturanalyse zu komprimieren, liegt auch der Syndrom-Test-Technik [23] sowie der Testmethode der Verifikation der Walsh-Koeffizienten [24] zugrunde. In beiden Methoden wird eine kombinatorische Schaltung durch einen kompletten Satz von Eingangsvektoren gespeist, wobei in einem Register das Syndrom bzw. die Summe einiger Walsh-Koeffizienten abgespeichert werden.

4.2.3 Strukturspezifische Testmethoden

In der digitalen Schaltungstechnik werden für häufig auftretende Funktionen spezielle, optimierte Baublöcke angeboten. Dies können z.B. Speicherelemente, PLA[8] (Programmable Logic Array) oder Mikroprozessoren sein.

Die Regularität solcher Bauelemente erlaubt, die Topologie verstärkt in den Test mit einzubeziehen. Anhand des PLA soll eine strukturspezifische Testmethode vorgestellt werden.

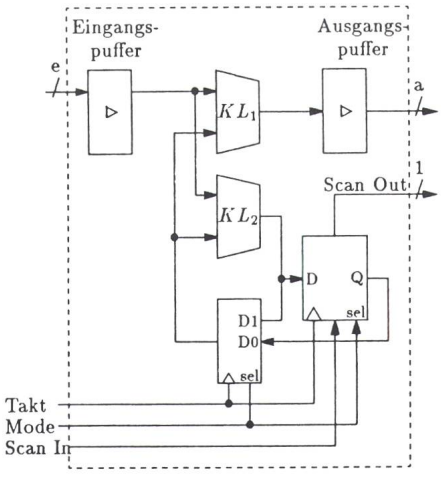
Bei PLA müssen nicht nur die klassischen Fehler wie Stuck-at-Fehler, Kurzschlüsse (Shorts) und Unterbrechungen (Opens) berücksichtigt werden, sondern auch die sogenannten Kreuzpunktfehler (Crosspoint Faults). Sie werden durch die An- bzw. Abwesenheit von Transistoren in den Schaltmatrizen (Und- bzw. Oder-Ebene) hervorgerufen. Die Bestimmung der kompletten Testvektormenge ist ein schwieriges und rechenintensives Unterfangen [25], das durch schaltungstechnische Massnahmen vereinfacht werden kann. Zur Illustration sei ein solches Vorgehen erläutert, welches als Basis zahlreicher weiterer Ideen gelten kann [26].

Zusätzlich zur PLA-Struktur sind die folgenden Hardware-Einheiten vorzusehen (Fig. 12):

- Mit einem Schieberegister (S_1, ..., S_5) lassen sich die einzelnen Produktterme (vertikale Linien) beliebig selektieren bzw. deselektieren, indem eine logisch 1 bzw. 0 zum entsprechenden Produkt «geschoben» wird.
- Eine zusätzliche Kolonne in der Und-Ebene und eine zusätzliche Zeile in der Oder-Ebene wird so programmiert, dass jeweils die totale Anzahl der Verknüpfungen ungerade wird; sie erlaubt das Auffinden von Fehlern durch Beobachten der Paritäten.
- Durch zusätzliche Steuereingänge (Y_1, Y_2) an der Eingangsstufe kann die absolute Kontrolle der einzelnen Eingangswerte erreicht werden.
- Zwei kaskadierte Exor-Gatter-Ketten zeigen an ihren Ausgängen Z_1, Z_2 Einzelfehler in den Und- und Oder-Ebenen an.

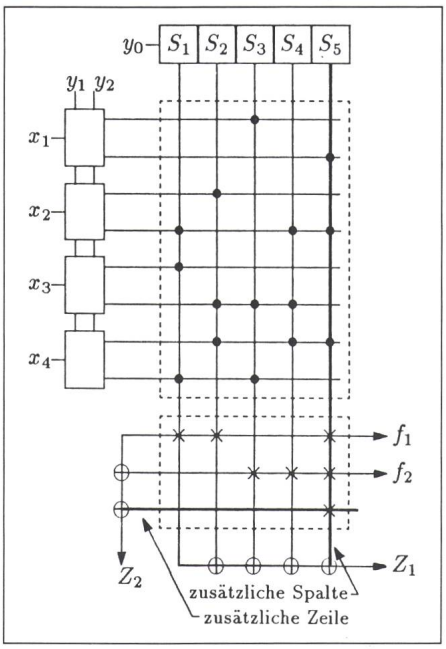
In Figur 13 ist der universelle Test-Set für das PLA mit n Eingängen und m Produkttermen aufgelistet. Als Beispiel sei das Auffinden eines Kreuzungspunktfehlers in der Oder-Ebene gezeigt: Die Eingänge der Und-Ebene werden auf logisch 1 gesetzt (x =

[8] Ein PLA ist ein integrierter Baustein mit einer programmierbaren zweistufigen Logik (Und- sowie Oder-Ebene).

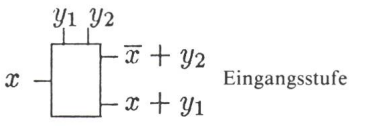


Figur 11 Beispiel für Scan-Set-Technik im Mealy-Modell

$e, a, 1$ Dimensionen des Eingangs- und Ausgangsvektors sowie des seriellen Ausgangs



Figur 12 Erweitertes PLA für universellen Test



⊕ EXOR-Verknüpfung
● UND-Verknüpfung (vertikal)
× ODER-Verknüpfung (horizontal)
x_i Eingangsvektor
S_i Selektionsvektor
f_1, f_2
Z_1, Z_2 Ausgänge der EXOR-Gatter-Ketten

	$x_1 \cdots x_i \cdots x_n$	y_1	y_2	$S_1 \cdots S_j \cdots S_m$	Z_1	Z_2
I_1	$X \cdots X \cdots X$	X	X	$0 \cdots 0 \cdots 0$	0	0

Für $j = 1, 2, \cdots, m$

I^0_{2j}	$0 \cdots 0 \cdots 0$	$1\ 0\ 0 \cdots 010 \cdots 0$	1 1
I^1_{2j}	$1 \cdots 1 \cdots 1$	$0\ 1\ 0 \cdots 010 \cdots 0$	1 1

Für $j = 1, 2, \cdots, n$

I^0_{3j}	$1 \cdots 101 \cdots 1$	$0\ 1\ 1 \cdots 1 \cdots 1$	ε_m –
I^1_{3j}	$0 \cdots 010 \cdots 0$	$1\ 0\ 1 \cdots 1 \cdots 1$	ε_m –

$$\varepsilon_m = \begin{cases} 0 & \text{falls } m \text{ ungerade} \\ 1 & \text{falls } m \text{ gerade} \end{cases}$$

Figur 13 Universeller Test Set

n Dimension Eingangsvektor
m Anzahl Produktterme

Mit den Testvektoren 2^0_j und 2^1_j wird die j-te Spalte des PLA selektiert. Durch das Beobachten von Z_2 lassen sich Kreuzungspunktfehler in der j-ten Spalte der ODER-Ebene feststellen. Durch Z_1 werden Stuck-at-0-Fehler der j-ten Spalte, die Stuck-at-1-Fehler der anderen Spalten sowie die Stuck-at-0-Fehler der Zeilen der UND-Ebene angezeigt. Mit den Testvektoren 3^j_j können alle Kreuzungspunkt-Fehler und Stuck-at-1-Fehler in der $2j$-ten Zeile der UND-Ebene durch Beobachten von Z_1 gefunden werden.

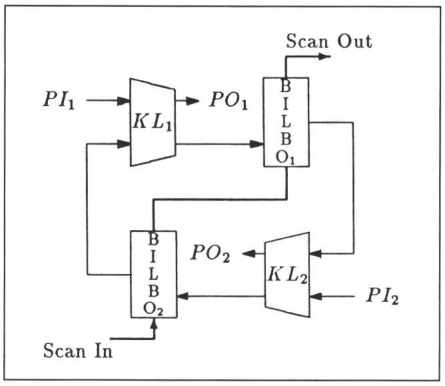
(0,0,0,0), y = (1,0). Mit (S = (1,0,0,0)) wird ausschliesslich die erste Spalte (Und-Term) selektiert. Ist ein Kreuzungspunktfehler vorhanden, so nimmt der Ausgang Z_2 den falschen Wert 0 an.

Diese Testmethode bietet eine hundertprozentige Fehlerabdeckung für Einfachfehler an. Die Testvektoren lassen sich unabhängig von der realisierten Funktion generieren, das heisst, sie sind universell. Die Testlänge wächst linear mit der Grösse des PLA ($2m + 3n$). Ausgehend von der skizzierten Testmethode sind weitere Vorschläge für das Testen von PLA-Strukturen entstanden [27; 28; 29].

4.2.4 Einbaubare Selbst-Test-Strukturen

Bis jetzt wurde nur auf das externe Prüfen von Schaltungen eingegangen, wobei ein externes Testgerät (ATE) die Testvektoren liefert und die Antwortsequenzen beobachtet. In diesem Abschnitt soll nun auf die einbaubaren Selbst-Test-Schaltungen (Built-in-Selftest) eingegangen werden. Motivierend für die Entwicklung von Selbst-Test-Methoden sind die kostenintensiven Testapparaturen sowie der limitierte Zugriff zu schaltungsinternen Knoten mit den damit verknüpften zeitintensiven Testabläufen.

Eine der populärsten Selbst-Test-Methoden ist der BILBO-Ansatz (Built-in Logic Block Observer) [30]. Dieser einbaubare Beobachtungsblock kombiniert den Scan-Design mit der Signaturanalyse, indem ein linear rückgekoppeltes Schieberegister (LFSR) auf dem Chip die Funktion eines Testvektorgenerators und Testvektoranalysators in einem übernimmt. Der Einsatz der BILBO-Strategie eignet sich besonders gut bei hochintegrierten, modularen Strukturen, in denen verschiedene kombinatorische Funktionseinheiten über Register mit-



Figur 14 Schaltungsentwurf in der BILBO-Technik

PI, PO Primäre Ein- und Ausgänge der kombinatorischen Logik-Blöcke

einander verbunden sind (Fig. 14). Der Testablauf für das Prüfen des kombinatorischen Blockes 1 umfasst die folgenden Punkte:

- $BILBO_1$ wird in den Signaturanalyse-Modus gebracht.
- $BILBO_2$ wird in den Testvektorgenerator Modus (Pseudo Random) betrieben.
- Eine Anzahl Pseudo-Zufallsvektoren werden über die primären Eingänge PI_1 und die $BILBO_2$-Ausgänge in den Logik-Block 1 eingespeist.
- Die Ausgangsvektoren des Logik-Blocks 1 werden als Signatur im $BILBO_1$ akkumuliert.
- Bei Testende wird die Signatur im Scan-Path-Modus aus dem $BILBO_1$ hinausgeschoben.

Entsprechend lässt sich der zweite kombinatorische Block überprüfen.

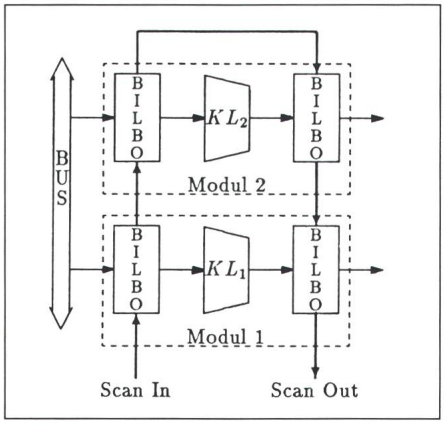
Ein weiteres Anwendungsfeld für die BILBO-Technik stellen modular aufgebaute Strukturen dar, welche über Register an einen Bus gekoppelt sind (Fig. 15, [30]). Der Vorteil dieser Testmethode gegenüber den Scan-Path-Techniken liegt in der Reduktion der Testzeiten. Bedingt durch die Generierung der Testvektoren und Komprimierung der Testanworten auf dem Chip selbst fällt das zeitintensive serielle Ein- und Auslesen der Vektoren weg. Zum Überprüfen von PLA-Strukturen eignet sich die BILBO-Technik nicht, da Gatter mit zahlreichen Eingängen nur sehr ineffizient durch Pseudo-Zufallsvektoren getestet werden können[9].

Bei der Entwicklung von sogenannten autonom testbaren Schaltungen [32] werden im wesentlichen die bisher beschriebenen Techniken miteinander verknüpft:

- Aufteilen der Schaltung in Funktionseinheiten, welche untereinander über Multiplexer kontrollier- und beobachtbar sind,
- Einsatz von Testvektorgeneratoren (Zähler oder linear rückgekoppelte Schieberegister),
- Verwendung von Komprimierungsmechanismen für die Testanworten,
- Einsatz von Vergleichseinheiten zur Überprüfung der komprimierten Daten. Diese Vergleicher können direkt das Signal *Chip fehlerfrei* produzieren.

Die zeitintensive Testvektorgenerierung und die kostspieligen Testapparaturen fallen bei einem solchen Entwurf weg. Der Nachteil liegt in der grösseren Entwicklungszeit der Schaltung sowie in der zusätzlich benötigten Siliziumfläche, welche sich negativ auf die Ausbeute auswirkt.

[9] Wie Zufallsvektoren mit nichtlinear rückgekoppelten Schieberegistern für PLA-Strukturen generiert werden können, ist in [31] gezeigt.



Figur 15 Modularer, busorientierter Entwurf mit BILBO-Technik

5. Vergleich der verschiedenen Testmethoden

Um die verschiedenen Testmethoden miteinander vergleichen zu können, müssen Kriterien aufgestellt werden. Aus der Sicht des Anwenders sind dies:

- kleiner Anteil defekter Bausteine pro Los,
- möglichst keine Pins, die nur für Testzwecke gebraucht werden,
- maximale Geschwindigkeit der Technologie, sie soll nicht aufgrund zusätzlicher Test-Hardware reduziert werden.

Für den Design- und Testingenieur gelten etwas andere Kriterien:

- Die gewählte Teststrategie soll leicht zu implementieren sein.
- Die Schaltung muss simulationsfreundlich aufgebaut sein, um einerseits kurze Entwicklungszeiten zu erreichen und andererseits die Funktionstüchtigkeit der Schaltung beweisen zu können.
- Bei zu hohen Ausbeuteverlusten muss die Fehlerquelle eruiert werden können.
- Die Siliziumfläche soll möglichst klein sein.
- Die Testzeiten sollen kurz sein und eine maximale Fehlerabdeckung ergeben.

Zur Siliziumfläche muss bemerkt werden, dass bei einer Vergrösserung der Chipfläche um einen Faktor a das Produkt wesentlich mehr als um diesen Faktor verteuert wird. Dies folgt aus der exponentiellen Abhängigkeit der Ausbeute Y von der Chipfläche A. Meistens wird heute eine Kopplung von zwei Fehlermodellen, dem Murphy-Modell [33]

$$Y = \left(\frac{1-e^{-A \cdot D}}{A \cdot D}\right)^2 \qquad (9)$$

und dem Seeds-Modell [34]

$$Y = e^{-\sqrt{A \cdot D}} \qquad (10)$$

angenommen, welche die Defektrate D (Defekt pro Flächeneinheit) und die Chipfläche A einbeziehen. Dadurch nehmen neben den reinen Chip-Preisen auch die Testzeiten zu. Dazu ist zu bemerken, dass die Ein- und Ausgangs-Pattern mit sehr hohen Taktraten an den Prüfling gebracht werden können (typisch 10 bis 50 MHz). Problematisch sind die Strommessungen, besonders für CMOS-Schaltungen, da Ruhe- und Eingangsleckströme in der Grössenordnung von 100nA gemessen werden müssen. Anderseits kann bei CMOS-Technologien anhand dieser Strommessungen der Prozess gut bewertet werden; insbesondere lassen sich Kurzschlüsse oder offene Gattereingänge leichter feststellen als durch Zustandsbeobachtung.

Literatur

[1] *J. Savir:* Good controllability and observability do not guarantee good testability. IEEE Trans C 32(1983)12, p. 1198...1200.

[2] *R.L. Wadsack:* Fault modeling and logic simulation of CMOS and MOS integrated circuits. Bell Syst. Techn. J. 57(1978)5, p. 1449...1474.

[3] *S.M. Reddy, M.K. Reddy* and *V.D. Agrawal:* Robust tests for stuck-open faults in CMOS combinational logic circuits. IEEE Proceedings of the 14th International Fault-Tolerant Computing Systems Conference 1984, p. 44...49.

[4] *J. Galiay, Y. Crouzet* and *M. Vergniault:* Physical versus logical fault models MOS LSI circuits: impact on their testability. IEEE Trans. C 29(1980)6, p. 527...531.

[5] *K.D. Wagner, C.K. Chin* and *E.J. McCluskey:* Pseudorandom testing. IEEE Trans. C 36(1987)3, p. 332...343.

[6] *J.P. Roth:* Diagnosis of automated failures: A calculus and a method. IBM Journal of Research and Development 10(1966)4, p. 278...291.

[7] *P. Goel:* An implicit enumeration algorithm to generate tests for combinational logic circuits. IEEE Trans C 30(1981)3, p. 215...222.

[8] *H. Fujiwara* and *T. Shimono:* On the acceleration of test generation algorithms. IEEE Trans. C 32(1983)12, p. 1137...1144.

[9] *S.A. Szygenda, D.M. Rouse* and *E.W. Thompson:* A model and implementation of a universal time delay simulator for large digital nets. AFIPS Conference Proceedings of the Spring Joint Computer Conference, May 5...7, 1970, Atlantic City, p. 207...216.

[10] *E.G. Ulrich* and *T.E. Baker:* Concurrent simulation of nearly identical digital networks. IEEE Computer 7(1974)4, p. 39...44.

[11] *D.B. Armstrong:* A deductive method for simulating faults in logic circuits. IEEE Trans. C 21(1972)5, p. 464...471.

[12] *R.G. Bennetts, C.M. Maunder* and *G.D. Robinson:* Camelot: a computer-aided measure for logic testability. IEE Proceedings, Part E 128(1981)5, p. 177...189.

[13] *R.G. Bennetts:* Design of testable logic circuits. London a.o., Addison-Wesley, 1984.

[14] *L.H. Goldstein:* Controllability/Observability analysis of digital circuits. IEEE Trans. CAS 26(1979)9, p. 685...693.

[15] *M.A. Breuer* and *A.D. Friedman:* Test/80–A proposal for an advanced automatic test generation system. Proceedings of the IEEE International Automatic Testing Conference 1979, p. 305...312.

[16] *J.A. Dussault:* A testability measure. Proceedings of the IEEE Semiconductor Test Conference 1978, p. 113...116.

[17] *P.G. Kovijanic:* Testability analysis. IEEE 1979 Semiconductor Test Conference, October 23...25, 1979, Cherry Hill/N.J. Digest of papers, p. 310...316.

[18] *I.M. Ratin, A. Sangiovanni-Vincentelli* and *D.O. Pedersen:* VICTOR: A fast VLSI testability analysis program. IEEE 1982 International Test Conference, November 15...18, 1982; Digest of Papers, p. 397...401.

[19] *E.I. Muehldorf* and *A.D. Savkar:* LSI logic testing–an overview. IEEE Trans. C 30(1981)1, p. 1...17.

[20] *T.W. Williams:* VLSI testing. Advances in CAD for VLSI. Vol. 5. Amsterdam a.o., North-Holland, 1986, p. 95...160.

[21] *T.W. Williams* and *K.P. Parker:* Design for testability–a survey. Proc. IEEE 71(1983)1, p. 98...112.

[22] *M.G. Buehler* and *M.W. Sievers:* Off-line, built-in test techniques for VLSI circuits. IEEE Computer 15(1982)6, p. 69...82.

[23] *J. Savir:* Syndrome-testable design of combinational circuits. IEEE Trans. C 29(1980)6, p. 442...451 + Nr. 11, p. 1012...1013.

[24] *A.K. Susskind:* Testing by verifying Walsh coefficients. IEEE Trans. C 32(1983)2, p. 198...201.

[25] *D.L. Ostapko* and *S.J. Hong:* Fault analysis and test generation for programmable logic arrays (PLAs). IEEE Trans. C 28(1979)9, p. 617...627.

[26] *H. Fujiwara* and *K. Kinoshita:* A design of programmable logic arrays with universal tests. IEEE Trans C.30(1981)11, p. 823...828.

[27] *J. Rajski* and *J. Tyszer:* Easily testable PLA design. In: Advances in microprocessing and microprogramming. Tenth Euromicro Symposium on Microprocessing and Microprogramming, Copenhagen, August 27...30, 1984, p. 139...146.

[28] *S. Bozorgui-Nesbat* and *E.J. McCluskey:* Lower-overhead design for testability of programmable logic arrays. IEEE Trans. C 35(1986)4, p. 379...383.

[29] *H. Fujiwara:* A new PLA design for universal testability. IEEE Trans. C 33(1984)8, p. 745...750.

[30] *B. Könemann, J. Mucha* and *G. Zwiehoff:* Built-in logic block observation techniques. Proceedings of the IEEE Test Conference, 1979, p. 37...41.

[31] *W. Daehn* and *J. Mucha:* A hardware approach to self-testing of large programmable logic arrays. IEEE Trans C 30(1981)11, p. 829...833.

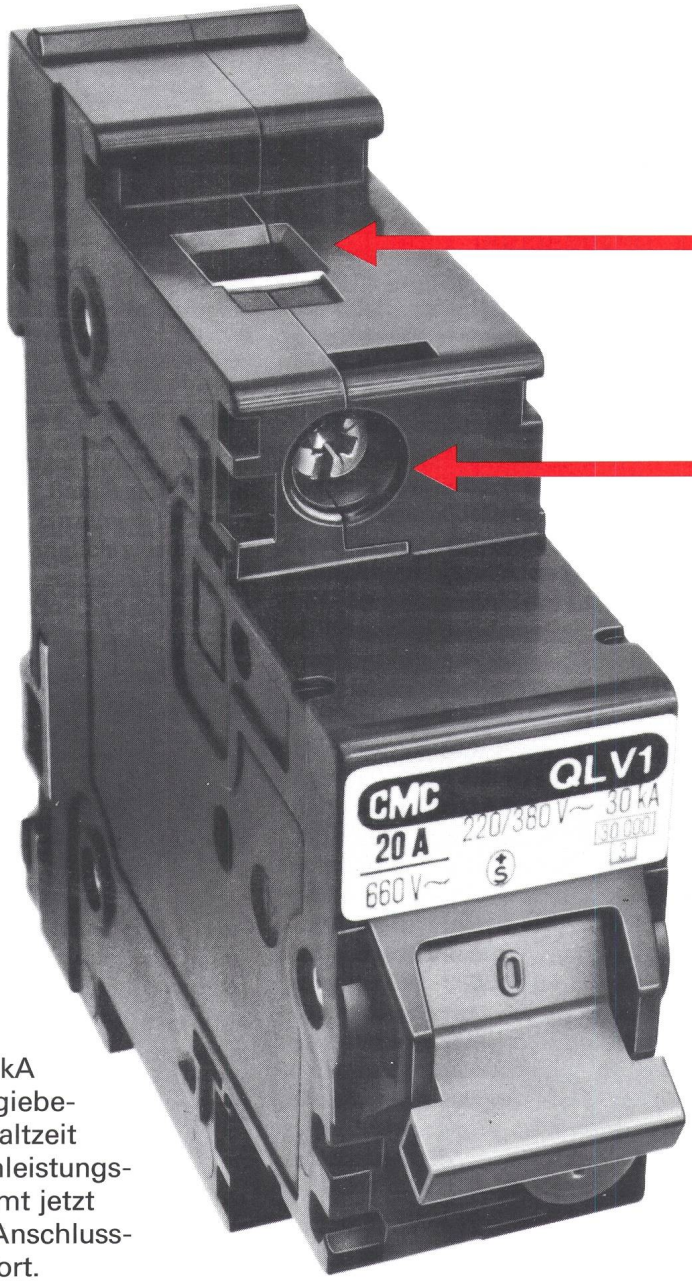
[32] *E.J. McCluskey* and *S. Bozorgui-Nesbat:* Design for autonomous test. IEEE Trans. C 30(1981)11, p. 866...875.

[33] *B.T. Murphy:* Cost-size optima of monolithic integrated circuits. Proc. IEEE 52(1964)12, p. 1537...1545.

[34] *R.B. Seeds:* Semiconductor handbook. Portola Valley/California, Technology Associates, 1967.

Dieser Artikel ist ein Auszug aus einer umfassenden Untersuchung, die zusätzlich in tabellarischer Form einen Vergleich der verschiedenen Methoden enthält. Dieser Bericht kann bei den Autoren angefordert werden.

Neu von CMC: Zusätzliche Vorteile für alle Hochleistungs-Automaten Q



- Käfigklemmen mit Drahteinführungshilfe
- Berührungsgeschützte, offene Klemmen
- Unverlierbare Plusminus-Schrauben
- Schraubenzieherführungen

Zusätzliche Vorteile, die besonders Praktiker schätzen: Zum hohen Nennschaltvermögen, 30 kA/50 kA (P2), und der energiebegrenzenden Abschaltzeit von 2 ms der Hochleistungs-Automaten Q kommt jetzt noch der erhöhte Anschluss- und Montagekomfort.
Hochleistungs-Automaten Q mit den neuen Vorteilen erhalten Sie wie bisher:

- für Leitungsschutz, QL, LV, LZ, G, Nennströme 6 – 63A
- für Leitungs- und FI-Schutz Q..FI.. Nennauslöseströme 10/15/30/300 mA
- für Gleichstrom, QDC, bis 750 V=
- mit Motorschutz-Charakteristik, QM, Nennströme 0,1 – 45 A
- mit anwendungsbezogener Auslösecharakteristik, QX

Interessiert? Dann schicken Sie einfach den ausgefüllten Coupon im frankierten Umschlag an:
CMC Carl Maier + Cie AG
Postfach
CH-8201 Schaffhausen

CMC

CMC Carl Maier+Cie AG, Schaffhausen
Apparate Systeme Anlagen

Coupon:
Informieren Sie mich umfassend über Ihre Hochleistungs-Automaten Q mit:

- ☐ Übersicht: Mit Leistungsdaten und Massen
- ☐ Sonderdruck: Aufbau und Anwendung des QLV-Hochleistungs-Automaten
- ☐ Sonderdruck: Q in Gleichstrombahnen
- ☐ Sonderdruck: Hochleistungs-Automaten Q im Maschinenbau

Name _____

Adresse _____

Firma _____

_____

_____