Automatische Synthese selbsttestbarer Moduln für hochkomplexe Schaltungen

Frank Kesel, Hans-Joachim Wunderlich

Universität Karlsruhe
Institut für Rechnerentwurf und Fehlertoleranz
(Prof. Dr. D. Schmid)
D-7500 Karlsruhe 1, Postfach 6980

Kurzfassung: Für den Test hochkomplexer digitaler Schaltungen bieten sich Selbsttestverfahren an, die auf multifunktionalen linear rückgekoppelten Schieberegistern beruhen. Diese erzeugen Pseudozufallsmuster und komprimieren die Testantworten zu einer Signatur. Durch einen automatischen Einbau der Selbsttestausstattung kann die Korrektheit des Entwurfs gewährleistet werden. Im vorliegenden Beitrag wird ein Verfahren vorgestellt, mit welchem sich multifunktionale Registerschaltungen automatisch synthetisieren lassen, welche gleich- und ungleichverteilte Pseudozufallsmuster erzeugen und die Testantworten durch Signaturanalyse komprimieren. Sie werden als Standardzellen erzeugt und können automatisch plaziert und verdrahtet werden.

Schlagworte: Selbsttest, automatische Synthese von Selbsttest-Hardware, ungleichverteilte Pseudozufallsmuster, linear rückgekoppelte Schieberegister

1. Selbsttest mit Pseudozufallsmustern

Technologische Fortschritte ermöglichen in zunehmendem Maße die Integration ganzer Systeme auf einem Chip, wobei die hohe Komplexität dieser Schaltungen nur durch einen weitgehend automatisierten Entwurfsablauf und gleichzeitige Berücksichtigung der Testbarkeit zu beherrschen ist. Beides wird in der englischsprachigen Literatur mit dem Begriff "Automatic Design for Testability" umschrieben (ADfT). Eine bekannte ADfT-Maßnahme ist beispielsweise der Einbau eines Prüfpfades nach dem LSSD-Prinzip [EiWi77], der die internen Speicherelemente einer Schaltung von außen zugänglich macht.

Da zahlreiche Defekte einer Schaltung nur durch einen Test mit hoher Geschwindigkeit erkannt werden können, steigen mit wachsender Leistungsfähigkeit der Schaltungen auch die Anforderungen an die Prüfautomaten. Für hochkomplexe Schaltungen bieten sich daher Selbsttestverfahren an, welche in normaler Betriebsgeschwindigkeit testen und bei denen der relative Anteil an Chipfläche für die Selbsttestausstattung hinreichend klein bleibt. Ein weit verbreitetes Selbsttestverfahren beruht auf multifunktionalen Registerschaltungen, beispielsweise "Built-In Logic Block Observer" (BILBO, [KMZ79]), mit denen im Testbetrieb Muster erzeugt und ausgewertet werden können. Bei zusätzlicher Integration eines Steuerwerks für den Selbsttest reduziert sich die Aufgabe des Testautomaten auf die Initialisierung der Register, die Erzeugung des Taktes und die Auswertung von Signaturen [HaWu89].

Der automatische Einbau der zusätzlichen Selbsttestausstattung soll die Korrektheit des Entwurfs gewährleisten ("correctness by construction") und den Entwerfer entlasten. Im vorliegenden Beitrag wird ein Verfahren vorgeschlagen, mit welchem multifunktionale Registerschaltungen für den Selbsttest mit Pseudozufallsmustern automatisch synthetisiert werden können. Zunächst werden einige grundlegende Sachverhalte des Zufallstests diskutiert und eine innovative multifunktionale Registerschaltung vorgestellt. Im zweiten Abschnitt wird die automatische Synthese dieser Registerschaltungen beschrieben und im letzten Abschnitt anhand eines Beispiels erläutert.

Bild 1 zeigt die zwei grundlegenden Typen linear rückgekoppelter Schieberegister (LRSR). Bei beiden Typen wird durch Antivalenzglieder eine Rückkopplungsfunktion realisiert. Der Zustand des LRSR zum Zeitpunkt τ wird durch einen Vektor $[s_0^\tau, ..., s_n^\tau]$ beschrieben. Ein LRSR vom Standardtyp berechnet über einen Baum von XOR-Gliedern den neuen Wert $s_0^{\tau+1} = f(s_0^\tau, ... , s_n^\tau)$ und erzeugt $s_{i+1}^{\tau+1} := s_i^\tau$ aus den anderen Stufen. Bei einem modularen LRSR werden die XOR-Glieder den einzelnen Stufen zugeordnet, so daß bei jedem Zustandswechsel ein Gatter durchlaufen wird und somit Geschwindigkeitsvorteile erreicht werden. Falls die eingehende Folge E Testantworten darstellt, genügt es, den Zustand

$[s_0^\tau, \ldots, s_n^\tau]$ des Registers, der Signatur genannt wird, nach τ Takten zu prüfen. Entspricht sie dem erwarteten Zustand, kann mit großer Wahrscheinlichkeit angenommen werden, daß die Folge E fehlerfrei war [(HeLe83), (Will87)]. Bei paralleler Signaturanalyse können auch gleichzeitig mehrere Datenströme E_i eingespeist werden.

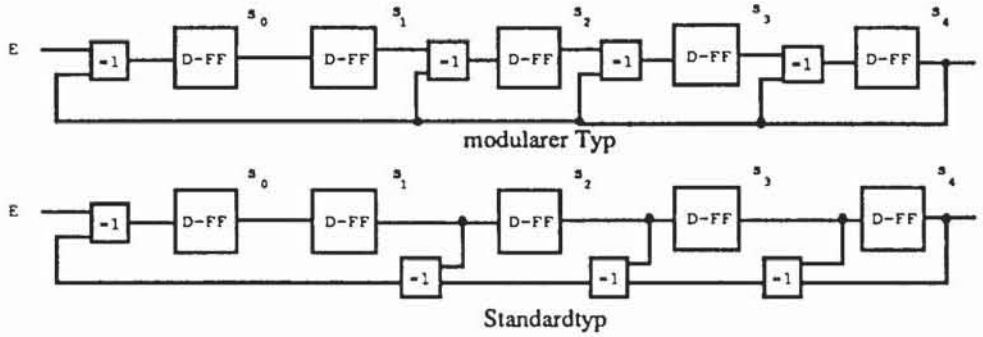

Bild 1: LRSR, modularer Typ und Standardtyp

Autonom betriebene (E ≡ 0) linear rückgekoppelte Schieberegister können, abhängig vom Anfangszustand $[s_0^0, \ldots, s_n^0]$ und der Rückkopplungsfunktion, maximal 2^n-1 Zustände durchlaufen. Es kann gezeigt werden, daß in diesem Fall an jeder Stelle s_i des Registers eine Musterfolge mit bestimmten Eigenschaften von gleichverteilten Zufallsfolgen entsteht [Golo82]. Sie wird daher als Pseudozufallsfolge bezeichnet.

Der Vektor $[s_0^\tau, \ldots, s_n^\tau]$ kann als Testmuster an ein Schaltnetz mit den primären Eingängen I angelegt werden. Die notwendige Testlänge ist dann die Zahl von Testmustern, auf die die Schaltung korrekt antworten muß, damit mit einer bestimmten vorgegebenen Wahrscheinlichkeit Fehlerfreiheit angenommen werden kann. Die Testlänge kann durch eine Testbarkeitsanalyse, z. B. mit PROTEST [Wund85], geschätzt werden. Sie hängt im wesentlichen von den Fehlern mit der geringsten Entdeckungswahrscheinlichkeit ab ([BaSa84], [Wund88]). Beim Test mit gleichverteilten Pseudozufallsmustern können sich hohe Testlängen ergeben, setzt man jedoch durch ungleichverteilte Pseudozufallsmuster [Wund87a] jeden Eingang $i \in I$ des Schaltnetzes mit einer für ihn spezifischen, optimalen Wahrscheinlichkeit $P_{i,opt} \in]0, 1[$ auf logisch 1, so läßt sich die erforderliche Testlänge reduzieren. Tabelle 1 zeigt einige Benchmarkschaltungen nach [Brgl85], für die mit PROTEST die Testlängen für gleichverteilte und ungleichverteilte Zufallsmuster bestimmt wurden. Hierbei wurde gefordert, daß alle Fehler mit einer Wahrscheinlichkeit größer als 98% entdeckt werden.

Schaltung	C880	C2670	C7552
Testlänge gleichverteilt	$3.7*10^4$	$1.1*10^7$	$4.9*10^{11}$
Testlänge ungleichverteilt	$6.6*10^2$	$6.9*10^4$	$1.2*10^4$

Tabelle 1: Testlängen für gleichverteilte und ungleichverteilte Zufallsmuster

Ungleichverteilte Pseudozufallsmuster lassen sich auch im Selbsttest durch den in [Wund87b] vorgeschlagenen GURT (Generator of Unequiprobable Random Tests) erzeugen. Hierbei werden von einem modularen LRSR, im folgenden mit LR[j] bezeichnet, an bestimmten Stellen gleichverteilte Pseudozufallsfolgen abgegriffen und einer booleschen Funktion F_j zugeführt, die so eine ungleichverteilte Pseudozufallsfolge erzeugt und diese wiederum einem Schieberegister SR[j] zuführt (Bild 2).

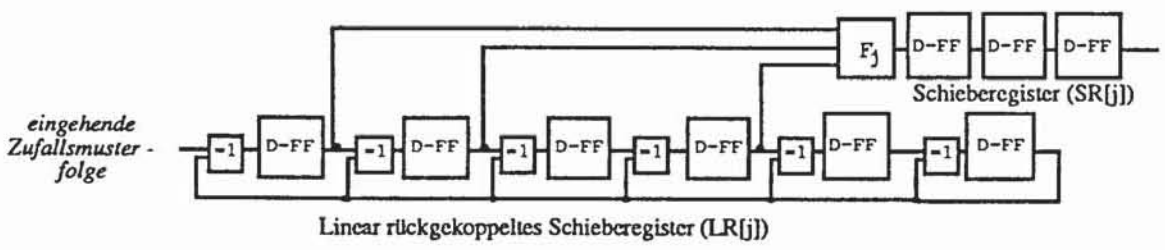

Bild 2: Erzeugung ungleichverteilter Pseudozufallsmuster durch Modul GR[j]

Der Index j zeigt an, daß die Funktion F_j eine Pseudozufallsfolge ausgibt, deren Bits mit Wahrscheinlichkeit p = j/8 zu "1" werden. Es hat sich gezeigt, daß die Erzeugung von Zufallsmustern im Wahrscheinlichkeitsraster von 1/8 ausreicht [Wund87a]. Man benötigt dazu drei Moduln GR[j] mit unterschiedlichen Funktionen F_j, da in SR[j] auch p = 1 - j/8 durch Invertierung erzeugt werden kann.

Die entstehende ungleichverteilte Bitfolge muß ebenfalls die Zufallseigenschaften aufweisen. Dies läßt sich erreichen, wenn in dem modularen Schieberegister LR[j] sowohl die Rückkopplungsfunktion als auch die Abgriffpunkte für die erzeugende F_j geeignet gewählt werden und LR[j] durch eine eingehende Pseudozufallsfolge stimuliert wird. Der erste Modul GR[j] wird daher durch ein maximales LRSR stimuliert, dessen Größe durch die benötigte Testlänge bestimmt wird. Es ergibt sich die GURT-Anordnung nach Bild 3.

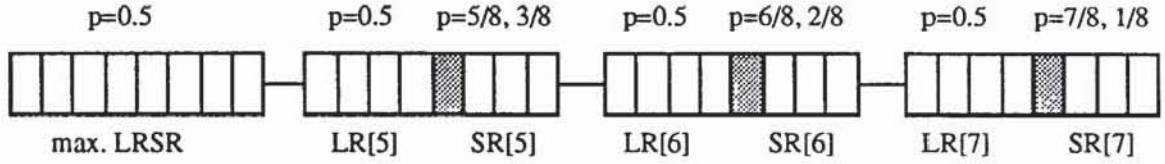

Bild 3: GURT-Anordnung (Flipflops der Selbsttestregister)

In [KeWu89] wurde ein Zellgenerator für entsprechende GURT-Standardzellen vorgestellt. Der Zellgenerator verlangt als Eingabe eine Beschreibung der GURT-Anordnung, nach der dann gewisse Grundzellen zu einer Standardzelle zusammengefügt werden. Zu einem gegebenen Schaltnetz mit optimierten Eingangswahrscheinlichkeiten lassen sich unterschiedliche GURT-Anordnungen finden, die sich bezüglich des Hardware-Aufwands und der notwendigen Testlänge unterscheiden. Falls die während der Testbarkeitsanalyse bestimmten, optimalen Wahrscheinlichkeiten stärker gerundet bzw. in der GURT-Anordnung die Muster in einem gröberen Raster als 1/8 erzeugt werden können, spart man durch die geringere Zahl der Moduln GR[j] Hardware-Aufwand, benötigt aber eine größere Testlänge. Bei der Entscheidung zwischen diesen beiden sich widersprechenden Zielvorgaben soll der Entwerfer durch ein Programm unterstützt werden. Zur Gewährleistung der Zufallseigenschaften ist es notwendig, daß jedes Modul LR[j] aus mindestens sechs Stufen besteht, die Musterfolgen mit Wahrscheinlichkeit p = 0.5 produzieren. Falls die Testbarkeitsanalyse nicht so vielen Eingängen als optimale Wahrscheinlichkeit p = 0.5 zugewiesen hat, muß die Anzahl der Moduln GR[j] reduziert oder eine entsprechende Anzahl an freilaufenden Registerzellen ohne Systemfunktion hinzugefügt werden. Für die automatische Synthese der GURT-Register benötigt man folglich ein Programm, welches aus einer Verteilung der optimalen Wahrscheinlichkeiten und entsprechenden Benutzervorgaben bezüglich des Hardware-Aufwands und der Testlänge verschiedene GURT-Anordnungen erzeugt. Im nächsten Abschnitt wird dieses Programm erläutert und ein Gesamtsystem zur GURT-Synthese vorgestellt.

2. Automatische Synthese von GURT-Registern

Ausgangspunkt der Synthese ist eine synchrone Schaltung, die in einzelne Moduln ohne speichernde Elemente partitioniert wurde. Für jeden primären Eingang der Schaltung wird die optimale Wahrscheinlichkeit $P_{i,opt}$ und für die Schaltung die zu erwartende Testlänge $T_{schätz}$ beispielsweise mit PROTEST ([Wund87a], [Wund85]) berechnet. Nun muß diejenige GURT-Anordnung bestimmt werden, welche unter gewissen, vom Benutzer vorgebbaren Randbedingungen zur kleinsten Testlänge führt.

Nachfolgend wird hierfür ein Algorithmus beschrieben, der die in Tabelle 2 zusammengestellten Parameter verwendet und mit zwei unterschiedlichen Strategien die erwähnten Konflikte bei der Zuordnung der optimalen Wahrscheinlichkeiten $P_{i,opt}$ zu realisierbaren Registern SR[j] auflöst. Die Expansionsstrategie erweitert die Schaltung um zusätzliche Flipflops, so daß alle Wahrscheinlichkeiten nur im 1/8-Raster gerundet werden müssen. Die Reduktionsstrategie führt eine stärkere Rundung durch und verzichtet auf die Realisierung einiger SR[j]. Die Rundung verursacht einen Fehler $\Delta(i) = | P_{i,opt} - P_{i,rast} |$, der die Abweichung von der optimalen Eingangswahrscheinlichkeit darstellt. Die Expansionsstrategie führt zu mehr Hardware-Aufwand, die Reduktionsstrategie dagegen durch einen größeren Rundungsfehler zu höheren Testlängen.

Programm-Parameter	
A. Aus der Schaltungsstruktur :	
SR[j]	:= { i ∈ I | $P_{i,rast}$ = j/8 oder $P_{i,rast}$ = 1 - j/8 }, j∈ {5,6,7}
A	:= { i ∈ I | $P_{i,rast}$ = 0.5 }
λ	: notwendige Länge des LRSR
B. Benutzervorgaben :	
L	: Länge der LR[j] ≥ 6
G	: Anzahl der Moduln GR[j]
STRATEGIE	: e = expandieren, r = reduzieren

Tabelle 2: Parameter des GURT-Synthese-Programms

Die obenstehenden Parameter aus Tabelle A werden aus der Schaltungsstruktur direkt oder mithilfe einer Testbarkeitsanalyse wie folgt erzeugt:

1. Die optimierten Eingangswahrscheinlichkeiten $P_{i,opt}$ werden eingelesen, zu $P_{i,rast}$ im vorgegebenen Raster gerundet und den entsprechenden Mengen SR[j] aus Tabelle 2 zugeordnet.

2. Die notwendige Länge λ des maximalen LRSR wird aus der Schätzung $T_{schätz}$ der Testlänge berechnet, so daß $2^{\lambda} > T_{schätz}$ gilt. Hierbei wird $T_{schätz}$ nicht auf der Basis der optimalen Wahrscheinlichkeiten $P_{i,opt}$ sondern auf Grundlage der tatsächlichen im Raster gerundeten Wahrscheinlichkeiten $P_{i,rast}$ bestimmt.

Damit und mit den vom Benutzer vorgegebenen Parametern nach Tabelle B läuft die Synthese folgendermaßen ab:

3. Eine den Benutzervorgaben entsprechende GURT-Anordnung wird bestimmt. Die Menge J := { SR[j] \| SR[j] ≠ {}, j ∈ { 5, 6, 7}} besteht aus den nichtleeren Mengen SR[j]. Es sind drei Fälle zu unterscheiden:

   I) Ist |A| ≥ λ + G * L, so sind genügend Eingänge mit $P_{i,rast}$ = 0.5 vorhanden und es wird die Prozedur "Zuteilen" aufgerufen, die jedem Primäreingang ein Flipflop zuordnet und im folgenden noch erläutert wird.

   II) Ist |A| < λ + G * L, so liegt ein Konflikt vor:

   IIa) Lösung des Konflikts durch die Reduktionsstrategie:
   Es werden iterativ fehlende Eingänge i ∈ I \ A aus den Mengen SR[j] entfernt und der Menge A zugewiesen. Hierbei kann der Fall auftreten, daß Mengen SR[j] leer werden und dann die Anzahl \|J\| der nichtleeren Mengen SR[j] kleiner als der Parameter G ist. In diesem Fall wird G dekrementiert und die Anzahl n der noch der Menge A hinzuzufügenden Eingänge neu berechnet, wobei dann weniger Moduln GR[j] realisiert werden als vom Benutzer vorgegeben wurde:

```
n := λ + G * L - |A|;
solange n > 0 :
        A := A ∪ {i}, i ∈ I \ A mit |P_i,opt - 0.5| minimal ;
        Für i ∈ SR[j] setze SR[j] := SR[j]\{i};
        Ist |J| ≥ G :
                n := n - 1;
        Ist |J| < G :
                G := G - 1;
                n := λ + G * L - |A|;
Zuteilen (A, G, J);
```

   IIb) Lösung des Konflikts durch die Expansionsstrategie:
   Zunächst werden die Eingänge i ∈ I \ A mit der Prozedur "Zuteilen" den entsprechenden Mengen zugeordnet. Da dabei auch Eingänge der Menge A zugeordnet werden können, werden erst anschließend die jetzt

noch fehlenden n := λ + G * L - |A| Elemente zur Menge A als freilaufende Registerzellen ohne Systemfunktion hinzugefügt.

Prozedur "Zuteilen(A,G,J)": Die Prozedur setzt voraus, daß die Primäreingänge aus A mit Wahrscheinlichkeit 0.5 belegt werden, und sucht eine Teilmenge $M \subseteq J$, so daß $|M| = G$ gilt, jeder Eingang $i \in I\backslash A$ einem $SR[j] \in M$ zugeordnet wird und dabei die Testlänge minimal ist. Da $|J| \leq 3$ ist, gibt es höchstens 3 solcher Mengen M_n, n = 1, ..., 3. Die durch M_n beschriebene Testkonfiguration kann bewertet werden, indem man jedem Eingang $i \in I\backslash A$ diejenige Wahrscheinlichkeit $P_{i,n} := j/8$ oder $P_{i,n} := 1-j/8$ zuordnet, für die $SR[j] \in M_n$ und zugleich der Rundungsfehler $\Delta_n(i) := |P_{i,opt} - P_{i,n}|$ minimal ist. Die Kosten der gesamten Konfiguration M_n kann beispielsweise durch

$$K(M_n) = \sum_{i \in I\backslash A} \frac{\Delta_n(i)}{P_{i,opt}(1 - P_{i,opt})}$$

geschätzt werden. Es wird das M_n mit der kleinsten Kostenfunktion als Lösung übernommen, $J := M_n$ gesetzt und die Eingänge werden entsprechend zugewiesen.

4. Im letzten Schritt wird aus den |J| nichtleeren Mengen SR[j] und der Menge A, welche nun eine gültige GURT-Anordnung angeben, in Verbindung mit den Tabellen für das Rückkopplungspolynom und die Abgriffpunkte Eingabedateien für den Zellgenerator und die Validierung erzeugt.

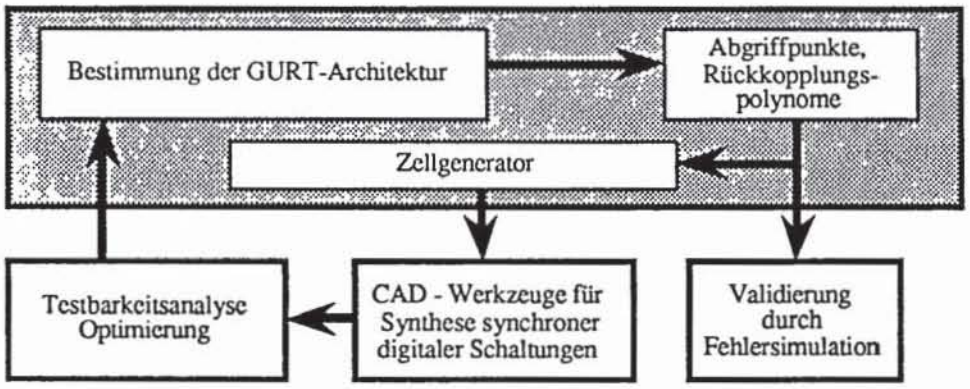

Bild 4: GURT-Synthese

Die GURT-Anordnung wird validiert, indem der Testsatz über eine Simulation der GURT-Register erzeugt und dessen Fehlererfassung über eine Fehlersimulation bestimmt wird. Hieraus erhält man auch die tatsächlich benötigte Testlänge T_{eff} und den zugehörigen Anfangszustand. T_{eff} wird vom Testautomaten oder von einem Teststeuerwerk auf dem Chip benötigt [HaWu89]. Die validierte GURT-Anordnung dient dann als Eingabe für den in [KeWu89] beschriebenen Zellgenerator.

3. Beispiel

Abschließend sollen die Ergebnisse verschiedener GURT-Anordnungen für die Beispielschaltung C880 [Brgl85] angegeben werden. C880 weist 60 primäre Eingänge auf, für die von PROTEST die optimalen Wahrscheinlichkeiten berechnet wurden. Mit dem vorgestellten Verfahren wurden dann drei verschiedene Anordnungen erzeugt, wobei die erste ausschließlich gleichverteilte Pseudozufallsmuster erzeugt. Während bei der zweiten GURT-Variante nur das Modul GR[6] verwendet wird, benutzt die dritte Variante die Moduln GR[6] und GR[7]. Die Testlängen T_{eff} in Tabelle 3 beziehen sich auf eine Fehlererfassung von 100%, d. h. alle möglichen Einfach-Haftfehler werden erkannt. Die Flächenangaben stellen den benötigten Hardware-Aufwand bezogen auf die Fläche eines Registers aus statischen D-Flipflops der Breite 60 bit dar, wobei es sich hier um das mit einem Zellgenerator erzeugte Layout handelt.

Man erkennt an diesem Beispiel, daß die Testlänge und der Hardware-Aufwand nicht linear voneinander abhängen, sondern daß sich durch Einsatz von nur einem Modul GR[j] die Testlänge schon drastisch reduzieren läßt. Eine weitere Reduktion der Testlänge ist aber durch einen überproportional starken Anstieg des Hardware-Aufwandes gekennzeichnet.

Dies zeigt nochmals, daß es für die zu testende Schaltung sinnvoll ist, verschiedene GURT-Varianten zu untersuchen und zu bewerten. Hierfür ist das vorgestellte Verfahren eine Hilfe, da es die automatische Erzeugung dieser Varianten und des zugehörigen Layouts gestattet, ohne daß der Benutzer vertiefte Kenntnisse über den Aufbau eines GURT benötigt.

	Testlänge T_{eff}	Fläche Testlogik
GURT (gleichverteilt)	28958	129%
GURT (GR[6])	635	153%
GURT (GR[6], GR[7])	293	231%

Tabelle 3: GURT-Varianten für C880 mit vollständiger Fehlererfassung

Literatur

[BaSa84] Bardell, P.H.; Savir, J.; On Random Pattern Test Length;
IEEE Trans. On Comp., Vol. C-33, No. 6, Juni 1984

[Brgl85] Brglez, F.; Pownall, P.; Hum, R.; Accelerated ATPG and Fault Grading via Testability Analysis;
Proc. IEEE, Intern. Symposium on Circuits and Systems, Kyoto, 1985

[EiWi77] Eichelberger, E.W.; Williams, T.W.; A Logic Design Structure for LSI Testability;
Proc. 14th Design Automation Conference, 1977

[Golo82] Golomb, S.W.; Shift Register Sequences;
Aegan Park Press, Laguna Hills, Cal., 1982

[HaWu89] Haberl, O.F.; Wunderlich, H.-J.; The Synthesis of Self-Test Control Logic;
Proc. CompEuro 89, Hamburg, 1989

[HeLe83] Heckmaier, J.H.; Leisengang, D.; Fehlererkennung mit Signaturanalyse;
Elektronische Rechenanlagen, Heft 3, 1983

[KeWu89] Kesel, F.; Wunderlich, H.-J..; Parametrisierte Speicherzellen zur Unterstützung des Selbsttests mit optimierten und konventionellen Zufallsmustern;
4. E.I.S.-Workshop, Bonn, 1989

[KMZ79] Koenemann, B.; Mucha, J.; Zwiehoff, G.; Built-In Logic Block Observation Techniques;
IEEE Test Conference, Cherry Hill, New Jersey, 1979

[Will87] Williams, T.W.; Daehn, W.; Gruetzner, M.; Starke, C.W.; Aliasing Errors with Primitive and Non-Primitive Polynomials;
IEEE Intern. Test Conference, 1987

[Wund85] Wunderlich, H.-J.; PROTEST: A Tool for Probabilistic Testability Analysis;
22nd Design Automation Conference, Las Vegas, 1985

[Wund87a] Wunderlich, H.-J.; Probabilistische Verfahren für den Test hochintegrierter Schaltungen;
Dissertation, Informatik Fachberichte 140, Springer, 1987

[Wund87b] Wunderlich, H.-J.; Self Test Using Unequiprobable Random Patterns;
International Symposium on Fault-Tolerant Computing, FTCS-17, Pittsburgh, 1987

[Wund88] Wunderlich, H.-J.; Multiple Distributions for Biased Random Patterns;
International Test Conference, Washington, 1988