# XP-SISR: Eingebaute Selbstdiagnose für Schaltungen mit Prüfpfad

Melanie Elm, Institut für Technische Informatik, Universität Stuttgart, D-70569 Stuttgart
Hans-Joachim Wunderlich, Institut für Technische Informatik, Universität Stuttgart, D-70569 Stuttgart

## Kurzfassung

Die Vorteile des Eingebauten Selbsttests (BIST — Built-In Self-Test) sind bekannt, für eingebettete Speicher ist BIST sogar die bevorzugte Teststrategie. Für freie Logik wird BIST deutlich seltener eingesetzt. Grund hierfür ist zum einen, dass deterministische Testmuster für eine hohe Fehlerabdeckung benötigt werden und diese im Selbsttest hohe Kosten verursachen. Zum anderen lassen sich aus den Testantworten, die zu einer einzigen Signatur kompaktiert werden, nur wenige diagnostische Informationen ziehen. In den vergangenen Jahren wurden kontinuierlich Fortschritte zur Lösung des ersten Problems erzielt. Dieser Beitrag befasst sich mit der Lösung des zweiten Problems.
Eine neue Methode für die Eingebaute Selbstdiagnose (BISD — Built-In Self-Diagnosis) wird vorgeschlagen. Kern der Methode ist eine kombinierte, extreme Raum- und Zeitkompaktierung, die es erstmals ermöglicht, erwartete Antworten und fehlerhafte Antworten mit vernachlässigbarem Aufwand auf dem zu testenden Chip zu speichern. Somit können in einer einzigen Selbsttestsitzung pro Chip alle zur Diagnose notwendigen Daten gesammelt werden.
Das BISD Schema umfasst neben der Kompaktierungshardware einen Diagnosealgorithmus und ein Verfahren zur Testmustererzeugung, die Aliasingeffekte und die durch die starke Kompaktierung verringerte diagnostische Auflösung kompensieren können. Experimente mit aktuellen, industriellen Schaltungen zeigen, dass die diagnostische Auflösung im Vergleich zum externen Test erhalten bleibt und der zusätzliche Hardware-Aufwand zu vernachlässigen ist.

## Abstract

The advantages of Built-In Self-Test (BIST) are well known, and for embedded memories BIST is already the preferred test method. However, for random logic BIST is less often employed. This is mainly due to the following two reasons: On the one hand, deterministic patterns might be necessary to achieve reasonable fault coverage, yet they are expensive in built-in tests. On the other hand, the diagnostic information provided by BIST-signatures is rather poor. During the last years the first issue has been tackled successfully. This paper deals with the second issue.
A new method for Built-In Self-Diagnosis (BISD) is presented. The method's backbone is a combination of extreme space and time compaction, which for the first time allows to store the expected test responses and the failing test responses with negligible overhead on chip. Consequently, all data relevant to diagnosis can be collected during a single self-test session.
The BISD method additionally comprises a diagnosis algorithm and a test pattern generation scheme, which overcome aliasing and the reduced diagnostic resolution introduced by the extreme compaction. Experiments with recent, industrial designs demonstrate, that diagnostic resolution is maintaned compared to external testing and the additional hardware needed to implement the BISD-scheme is negligibly small.

## Schlüsselwörter / Keywords

Logic BIST, Diagnosis

## 1 Einleitung

Die Vorteile des eingebauten Selbsttests (BIST) sind einerseits Einsparungen bei den teuren externen Testgeräten, ein hoher Durchsatz, eine hohe Defektabdeckung auch für Defekte, die nicht explizit von der Testmenge erfasst werden, und die Wiederverwendbarkeit im Feld (siehe [1]). Aus diesen Gründen ist BIST die bevorzugte Testmethode für eingebettete Speicher [2], die mit geringen Hardware Kosten verbunden ist und einfach um eine *stop-on-$n^{th}$-fail* Diagnose erweitert werden kann. Hierfür werden die ersten $n$ fehlerhaften Operationen gespeichert, diese sind in den meisten Fällen für eine Fehlerdiagnose ausreichend [3].

Für den Produktionstest freier Logik wird BIST jedoch seltener eingesetzt, da sich hier einige nicht triviale Probleme ergeben:

1) Der zusätzliche Platzbedarf zur Generierung deterministischer Testmuster, die eine gute Fehlererfassung ermöglichen, kann zu groß werden.
2) Die diagnostische Auflösung der kompaktierten Antworten kann zu niedrig oder der Platzbedarf für eine akzeptable Auflösung zu hoch sein.

Die nicht schraffierten Blöcke in Abbildung 1 stellen eine generische Struktur für den pseudo-zufälligen, eingebauten Selbsttest dar. Ein Mustergenerator erzeugt pseudozufällige Muster, die in Prüfpfade geschoben werden. Die Testantworten werden aus den Prüfpfaden in einen Kompaktor geschoben, der eine einzige Signatur für den gesamten Test erzeugt. Am Ende des Tests wird die Signatur des Kompaktors vom Tester ausgelesen und mit der Soll-Signatur verglichen.

Um eine hohe Fehlererfassung zu erreichen, ist es für

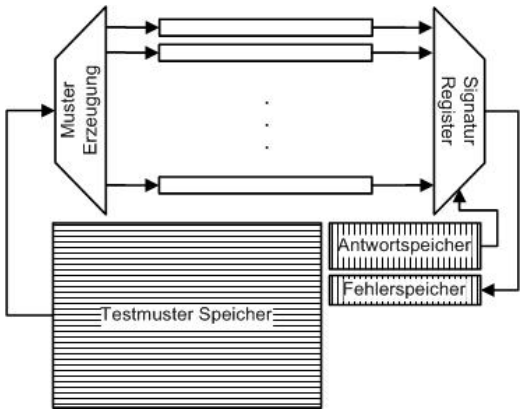


**Bild 1** Prinzipieller Aufbau einer Selbsttestarchitektur mit Erweiterung um deterministische Testmuster und BISD-Hardware.

die meisten, aktuellen Schaltungen notwendig, deterministisch erzeugte Testmuster anzuwenden. In diesem Fall wird ein Testmusterspeicher auf den zu testenden Chip synthetisiert. Er entspricht dem horizontal schraffierten Block in Abbildung 1. In den vergangenen Jahren sind kontinuierliche Fortschritte erzielt worden, die deterministische Mustermenge effizient zu kodieren [4], [5], [6], so dass nun Methoden zur Verfügung stehen, die das Problem auch für große, industrielle Schaltungen effizient lösen [7].

In diesem Beitrag wird die Selbsttestarchitektur modifiziert und um eine BISD Hardware ergänzt. Hierzu wird die Hardware um den Antwort- und den Fehlerspeicher ergänzt (vertikal schraffierte Blöcke in Abbildung 1). Der Antwortspeicher enthält die Soll-Signaturen für jedes Testmuster, während im Fehlerspeicher die Ist-Signaturen abgelegt werden, die von der Soll-Signatur abweichen. Kern der Architektur ist ein Kompaktor, der extrem kurze Signaturen für jedes Muster berechnet. Aufgrund der Kürze der Signaturen nehmen dann der Antwort- und der Fehlerspeicher nur einen Bruchteil der Fläche des Testmusterspeichers ein und können somit ebenfalls auf den zu testenden Chip synthetisiert werden.

Um Aliasingeffekte und eine geringe diagnostische Auflösung der kurzen Signaturen zu umgehen, wird die Methode ergänzt durch ein auf die Architektur abgestimmtes Verfahren zur Testmustererzeugung (ATPG) und einen statistischen Diagnosealgorithmus, der auch bei verminderter diagnostischer Auflösung zuverlässig den zugrunde liegenden Fehler ermitteln kann.

Die Architektur zusammen mit der Testmustererzeugung und dem Diagnoseverfahren ermöglicht erstmalig, für Entwürfe mit Prüfpfad während einer einzigen Selbsttestsitzung alle für die Diagnose relevanten Daten zu sammeln. Experimentelle Ergebnisse zeigen, dass die Kombination des BISD Schemas, des Diagnoseverfahrens und der Testmustererzeugung nur einen Bruchteil des Hardware Aufwandes benötigen, den ein deterministischer Mustergenerator benötigt, und eine diagnostische Auflösung ergeben, welche die eines externen Tests sogar übersteigen kann. Das vorgeschlagene Schema berücksichtigt unbekannte Werte (X-Werte) in den Prüfpfaden nicht explizit, aber erlaubt die Anwendung von zusätzlichen X-Maskierungsschemata wie z.B. [8], [9], [10].

Der Rest des Beitrags ist wie folgt aufgebaut. Der nächste Abschnitt stellt die neue BISD Architektur vor, die auf einen *mixed mode BIST* für beliebige Defekte aufsetzt. Das Verfahren wird den bisherigen Ansätzen zum Sammeln diagnostischer Daten im Selbsttest gegenübergestellt. In Abschnitt 4 wird der Diagnosealgorithmus beschrieben, der aus den ersten $n$ fehlerhaften Signaturen die Defekte der Schaltung extrahiert. Abschnitt 4 stellt die modifizierte Testmustererzeugung (ATPG) vor, die das Aliasing für die vorgeschlagene Architektur minimiert. In Abschnitt 5 werden Experimente mit industriellen Schaltungen diskutiert, die zeigen, dass eine hohe diagnostische Auflösung mit geringem Hardware Aufwand erreicht werden kann.

# 2 Diagnostische Information im eingebauten Selbsttest

Wegweisende Arbeiten im Bereich der Testantwort Modifikation und Generierung sind von Agarwal, Aitken und Zorian [11], [12] vorgestellt worden. Das zugrunde liegende Prinzip ist in Abbildung 2 dargestellt. Die Testantworten der zu diagnostizierenden Schaltung werden modifiziert bzw. kompaktiert. Dies wird kombinatorisch [12] oder sequentiell durch Signaturanalyse [11] implementiert. Die kompaktierten Soll-Antworten der zu diagnostizierenden Schaltung werden auf einem Testchip erzeugt (dies kann zum Beispiel durch ein ROM oder ein PLA realisiert sein).

Die Soll- und Ist-Antworten werden verglichen und wenn der erste Fehler auftritt, wird dieser im Fehlerspeicher festgehalten. Das zugehörige Diagnoseverfahren versucht aus vorberechneten Erkennungswahrscheinlichkeiten und der ersten, falschen Antwort des Prüflings den Fehler diagnostizieren. Die größte Komponente in diesem Schema bildet die Antwort Generierung, sie wird daher auf einen Testchip ausgelagert. Somit eignet sich das Verfahren nicht für den Test im Feld. Das Schema ist außerdem nur begrenzt auf aktuelle Schaltungen anwendbar, da es weder den Prüfpfad basierten Entwurf noch eine deterministische Mustererzeugung unterstützt.

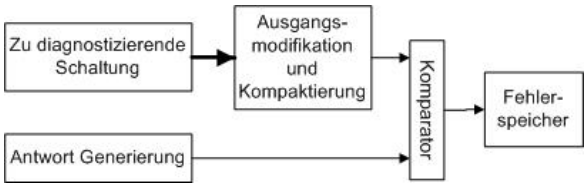


**Bild 2** Testantwort Generierung und Modifikation nach [11], [12].

In aktuellen, für den Selbsttest ausgerüsteten Schaltungen, wie sie in Abbildung 1 in den nicht schraffierten und den horizontal schraffierten Blöcken schematisch dargestellt sind, wird der Test durchgeführt, in dem alle Testmuster angewendet und die entsprechenden Antworten in eine einzige Signatur kompaktiert werden. Diese Signatur

wird vom externen Tester ausgelesen und mit der Soll-Signatur verglichen. Sie liefert ausreichend Information, um einen defekten Chip von einem fehlerfreien zu unterscheiden. Im Allgemeinen bietet die fehlerhafte Signatur am Ende des Tests jedoch nicht genügend diagnostische Auflösung und es ist notwendig, Teilsignaturen als Zwischenergebnisse zu analysieren [13].

Da das Speichern der Teilsignaturen für jedes Testmuster viel Fläche benötigen würde, beruhen die meisten BIST Schemata auf einem mehrphasigen Test. In einer ersten Testsitzung werden defektfreie von defekten Chips getrennt. Die defekten Chips werden dann mit mehreren folgenden Testsitzungen wiederholt getestet. Es können drei Methoden unterschieden werden, die Wiederholungen können auf verschiedenen Untermengen der Prüfelemente oder Prüfpfade ausgeführt werden, dies wird zum Beispiel in [14], [15], [16], [17] beschrieben. Sie können wie in [18] auf verschiedenen Untermengen der Testmustermenge ausgeführt werden, oder wie in [19], [20], [21] mit einer anderen Kompaktierung oder anderen LFSR Rückkopplungsfunktionen. Alle diese Ansätze haben gemeinsam, dass mehrere Signaturen außerhalb des Chips ausgewertet werden müssen und der Test mehrfach durchgeführt werden muss.

Die hier vorgeschlagene BISD Architektur ist in Abbildung 3 dargestellt. Auf der Eingangsseite kann ein beliebiges mixed mode BIST Schema angewendet werden (z.B. [7]). Die Testmenge, die aus pseudo-zufälligen und deterministisch erzeugten Mustern besteht, bezeichnen wir mit $T$.

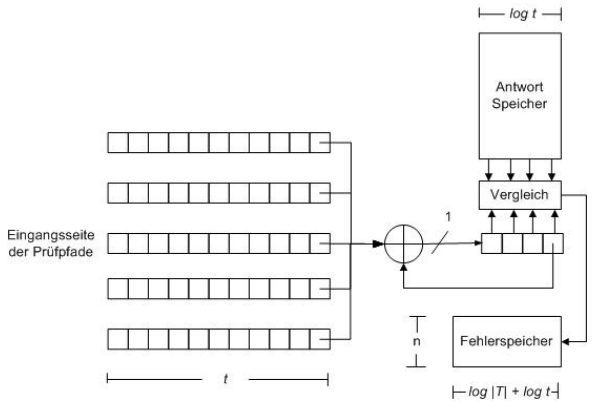


**Bild 3** BISD Schema.

Die Testantwortkompaktierung besteht aus drei Komponenten:

1) einer extremen Raumkompaktierung, wie sie in [22] vorgeschlagen wurde,
2) einer anschließenden Zeitkompaktierung mit einem Single Input Signature Register (SISR) und
3) einer *Vergleich* Komponente, die vom SISR befüllt wird und mit der Ausgabe des *Antwort Speichers* vergleicht.

Sei $k$ die Anzahl der Prüfpfade und $t$ die maximale Länge eines Prüfpfades. Die $k$ Bits, die im selben Takt aus den Prüfpfaden geschoben werden, bezeichnen wir als *Prüfvektor*. Der Raumkompaktor kompaktiert alle Bits eines Prüfvektors in ein einziges Paritätsbit. Folglich resultiert die Raumkompaktierung pro Muster in einem Bitstrom der Länge $t$. Dieser Bitstrom wird in ein SISR der Länge $\lceil \log t \rceil$ gegeben. Dies ermöglicht eine eineindeutige Signatur für jeden Einzelbitfehler im Paritäts-Bitstrom. Die Signatur eines Testmusters ist nun also $\lceil \log t \rceil$ Bits lang. Wir nennen dieses neue Kompaktierungsschema XP-SISR.

Nachdem eine ganze Testantwort aus den Prüfpfaden herausgeschoben wurde, wird die entstandene SISR Signatur mit der zugehörigen, vorberechneten Soll-Signatur verglichen. Die Soll-Signaturen können während der Fehlersimulation vorberechnet werden und werden dann im Antwortspeicher abgelegt. Der Antwortspeicher enthält dann sowohl die Soll-Signaturen für die deterministischen Testmuster, als auch die für die Zufallsmuster. Er hat somit eine Größe von $\lceil \log t \rceil \cdot |T|$ Bits.

Der Vergleich wird von der Vergleichskomponente durchgeführt, die außerdem auch nach jedem Testmuster das SISR zurücksetzt. Wird beim Vergleich der Soll- und der Ist-Signatur ein Fehler entdeckt, so wird die fehlerhafte Ist-Signatur im Fehlerspeicher abgelegt. Er bietet Platz für bis zu $n$ fehlerhafte Signaturen und den Index des zugehörenden Testmusters und hat somit eine Größe von $n \cdot (\lceil \log |T| \rceil + \lceil \log t \rceil)$ Bits. Der in Abschnitt 3 vorgestellte Diagnosealgorithmus kann aus den ersten $n$ fehlerhaften Ist-Signaturen zuverlässig den zugrunde liegenden Fehler ermitteln.

Nach der Testsitzung kann der Fehlerspeicher ausgelesen werden. Wenn dieser leer ist, wird der Chip als defektfrei angenommen, andernfalls kann auf dem Inhalt des Fehlerspeichers die Diagnose ausgeführt werden, ohne dass eine weitere Testsitzung nötig ist.

Kern des vorgestellten Ansatzes ist die extreme Kompaktierung, die das Antwortdatenvolumen so stark reduziert, dass die Soll- und fehlerhaften Ist-Signaturen auf dem zu testenden Chip gespeichert werden können. Die Architektur erfordert nur vernachlässigbaren, zusätzlichen Hardwareaufwand, verringert die Anzahl der Testsitzungen auf 1 für alle zu testenden Chips und ist im Feld einsetzbar.

# 3 Logikdiagnoseverfahren für den eingebauten Selbsttest

Diagnose auf fehlerhaften BIST-Signaturen lässt sich in direkte und indirekte Ansätze klassifizieren. Indirekte Verfahren rekonstruieren aus der Signatur die Information, welche Prüfelemente für welches Testmuster einen Fehler observiert haben [23], [19], [24], [25], [26], [27]. Diese Information kann dann als Eingabe für Standarddiagnoseverfahren für kombinatorische Logik verwendet werden [28], [13], [29], [30].

Direkte Diagnoseansätze arbeiten direkt auf der Signatur, ohne die fehlerobservierenden Prüfelemente zu berechnen. In [31] wurde beispielsweise eine direkte Diagnose auf MISR Signaturen vorgeschlagen und erfolgreich auf BIST

für zufallstestbare Fehler angewendet. Diese Methode basiert auch auf zwei BIST Sitzungen und ihre Ergebnisse demonstrieren eine hohe diagnostische Auflösung für zufallstestbare Fehler und eine geringe Test- / Diagnosezeit. Trotz der direkten Diagnose sind jedoch zwei Test Sitzungen nötig und während der zweiten Testsitzung muss ein hohes Datenvolumen zwischen externem Tester und Chip ausgetauscht werden.

Der hier verwendete Ansatz zur Logikdiagnose beruht auf einer direkten Diagnose mit den extrem kurzen Signaturen und ist in der Lage, trotz der sehr kompakten, oben beschriebenen Signaturen zuverlässig die zugrunde liegenden Fehler zu ermitteln.

Der Diagnosealgorithmus basiert auf dem SLAT Ansatz (Single Location At a Time) [32]. Jedoch wurde der Ansatz erweitert, um die Diagnoseinformation der gesamten Testmustermenge zu nutzen, auch die der korrekten Antworten [33]. Der Ansatz wird direkt auf die Signatur angewendet, ohne dass die fehlerobservierenden Prüfelemente ermittelt werden müssen.

Der Ansatz ist fehlermodellunabhängig, da er fehlerhafte Werte an internen Signalen als so genannte bedingte Haftfehler annimmt, die von einer Untermenge der Testmustermenge aktiviert werden. Ein bedingter Haftfehler ändert den logischen Wert eines Signals in Abhängigkeit von Bedingungen, die beispielsweise Werte von Nachbarleitungen, temporäre oder sogar nicht deterministische Eigenschaften sein können. Damit lässt sich eine Vielzahl von Fehlermodellen wiedergeben.

Für jeden Haftfehler $f$ und jedes Testmuster $p \in T$, beginnen wir damit, eine Referenzsignatur zu generieren $s_{ref}(f,p)$. Wir nennen $S_{ref}(f) = \bigcup_{p:T}\{s_{ref}(f,p)\}$ die Signatur von Fehler $f$. Die fehlerhaften Signaturen, die vom Prüfling heruntergeladen werden, können leicht in eine ähnliche Signatur $S_{observed}$ konvertiert werden, in dem man die korrekten XP-SISR Signaturen für die Muster, die keinen Fehler hervorgerufen haben, einfügt. Die Signaturen $S_{ref}(f)$ und $S_{observed}$ werden dann bezüglich der unten definierten Metriken verglichen.

- $\sigma^f$: Anzahl der sowohl in $S_{observed}$ als auch in $S_{ref}(f)$ fehlerhaften Bit Positionen.
- $\iota^f$: Anzahl der Bit Positionen, die in $S_{ref}(f)$ fehlerhaft und in $S_{observed}$ fehlerfrei sind.
- $\tau^f$: Anzahl der Bit Positionen, die in $S_{ref}(f)$ fehlerfrei und in $S_{observed}$ fehlerhaft sind.

Die Fehler werden gemäß dieser Metriken so sortiert, dass der erste Fehler in der sortierten Liste am wahrscheinlichsten dem tatsächlichen Defekt in der Schaltung entspricht. Zur Sortierung werden die Metriken wie folgt interpretiert: Wenn der Defekt im Prüfling einem einfachen Haftfehler entspricht, gibt es ein $S_{ref}(f)$, für das gilt $\iota^f = 0$, $\tau^f = 0$ und $\sigma^f > 0$ ist maximal unter allen Fehlern $f$.

Wenn $\sigma^f$ maximal, $\iota^f > 0$ und $\tau^f = 0$, dann wird der Fehler nicht für alle Muster aktiv sondern nur unter bestimmten Bedingungen. Beispiele für Defekte, die einem solchen Verhalten zugrunde liegen, sind unter anderem ständig offene Transistoren oder Leitungen, deren Verhalten vom logischen Wert im vorangehenden Taktzyklus abhängt, oder Fehler durch Übersprechen, die durch Aktivität auf benachbarten Leitungen hervorgerufen werden.

Wenn $\tau^f > 0$ für alle Fehler $f$, gibt es keine einzelne Leitung, die das fehlerhafte Verhalten erklären kann. Ein Beispiel hierfür sind 4-Wege Brücken. Dennoch ist $\sigma^f$ für die Haftfehler, die den Defekt am besten erklären, maximal.

Die Sortierung wird nun definiert durch

$$r(f) > r(f') \quad \Leftrightarrow \quad \begin{cases} \sigma^f > \sigma^{f'} \text{ oder} \\ \sigma^f = \sigma^{f'} \wedge \iota^f > \iota^{f'} \text{ oder} \\ \sigma^f = \sigma^{f'} \wedge \iota^f = \iota^{f'} \wedge \tau^f < \tau^{f'} \end{cases}$$

Es wird also zunächst nach $\sigma$, dann nach $\iota$ und schließlich nach $\tau$ sortiert. Das Signal, das zu dem Fehler $f$ mit dem höchsten Wert $r(f)$ korrespondiert, ist mit der größten Wahrscheinlichkeit Ursache für das fehlerhafte Verhalten. Auf diese Weise ist es möglich, direkt aus der Signatur zu diagnostizieren, ohne die BISD Sitzung zu wiederholen.

Der nächste Abschnitt befasst sich mit der Testmustererzeugung für die vorgeschlagene BISD Architektur.

# 4 ATPG für BISD

Für die Automatische Testmuster Erzeugung (ATPG) ist das XP-SISR Schema in Abbildung 3 eine große Herausforderung:

1) Es gilt, Aliasingeffekte zu vermeiden, um eine gute Fehlerabdeckung zu erreichen.
2) Die Testmustermenge muss effizient kodierbar sein, um die Hardware Kosten auf der Eingangsseite der Schaltung zu minimieren.

ad1) Im XP-SISR werden sowohl Raum- als auch Zeitkompaktierung verwendet und daher können zwei Ausprägungen des Aliasings auftreten.

Fehlerauslöschung im XOR-Baum des Raumkompaktors kann zu Aliasing Effekten führen. Dies ist in Abbildung 4 dargestellt. Wenn in jedem Prüfvektor eine gerade An-

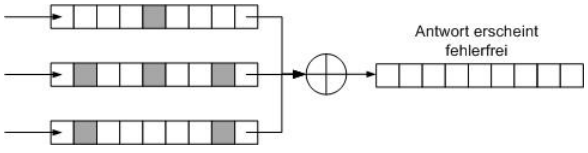


**Bild 4** Aliasing in extremen Raumkompaktoren.

zahl fehlerobservierender Prüfelemente liegt, löschen sich diese im XOR-Baum gegenseitig aus. Es muss also sichergestellt werden, dass es zu jedem Fehler $f$ mindestens ein Testmuster $p$ gibt, das in mindestens einem Prüfvektor zu einer ungeraden Anzahl von fehlerobservierenden Prüfelementen führt. Dies kann ein ATPG automatisch erreichen, wenn der Kompaktor vor dem ATPG an die Schaltung synthetisiert wird [22]. Die Schaltung, wie sie dem ATPG übergeben wird, ist in Abbildung 5 dargestellt. Die Prüfketten werden aufgelöst, und an die

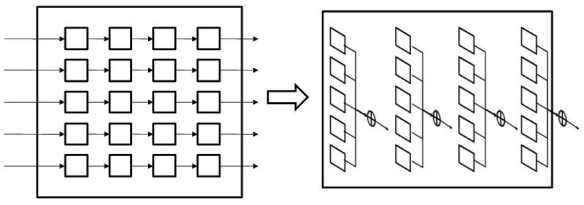

**Bild 5** Schaltungsmodell für das ATPG.

Schaltungsausgänge zu jeweils einem Prüfvektor wird ein XOR-Baum angehängt. Dies entspricht einer Schaltung mit Prüfpfaden, deren Vektoren in einen Paritätsbaum geschoben werden. Der ATPG Algorithmus vermeidet das Aliasing automatisch beim Versuch, eine hohe Fehlerabdeckung zu erreichen. Für das ATPG kann jedes Werkzeug genutzt werden, dass für den deterministischen BIST effektiv ist.

Die zweite Form des Aliasings kann im SISR auftreten. Das SISR verwendet ein primitives Polynom und hat die Länge $\lceil \log t \rceil$. Es kann daher jeden Einzelfehler im Paritätsstrom eines Testmusters eindeutig identifizieren und erkennt auch jeden Doppelfehler. Der oben beschriebene Diagnosealgorithmus kann dann aus der fehlerhaften Signatur ausreichend Information zur Fehlerlokalisierung ziehen. Sind mehr als zwei Bits im Paritätsstrom fehlerhaft, beträgt die Maskierungswahrscheinlichkeit höchstens $1/t$, jedoch haben solche Fehler in der Regel eine hohe Erkennungswahrscheinlichkeit und werden dann auch von anderen Mustern erkannt. Wie die experimentellen Ergebnisse im nächsten Abschnitt zeigen, sind weitere Maßnahmen zur Aliasingreduktion im SISR daher nicht notwendig.

ad2) Die meisten deterministischen BIST Schemata machen sich zunutze, dass deterministische Testmuster nur aus wenigen Care-Bits (spezifizierten Bits) bestehen. Ihr Hardware Overhead hängt meist linear von der Anzahl der Care Bits in der Testmustermenge ab [34], [35], [4], [6]. Eine Testmustermenge, die die Forderung 1 erfüllt, hat jedoch zum einen eine sehr hohe Anzahl an Care Bits und ist zusätzlich meistens größer als eine gewöhnliche Testmustermenge. Der Grund hierfür liegt darin, dass eine Mustermenge mit vielen Care Bits schlecht statisch oder dynamisch kompaktierbar ist. Dennoch ist es möglich, die Testmenge effizient zu kodieren. Ein entsprechendes Verfahren, das zweistufig arbeitet und auf Stripping Algorithmen beruht [36], [37], [38], ist in [39] beschrieben. Es kann analog für die hier vorgeschlagene BISD Architektur angewendet werden und ist für jeden Testmustergenerator adaptierbar. Durch das hier vorgeschlagene BISD Verfahren wird der Hardwareaufwand auf der Eingangsseite kaum berührt.

# 5 Experimentelle Ergebnisse

Um die BISD Architektur zu validieren, wurden Experimente mit industriellen Schaltungen durchgeführt, die von NXP zur Verfügung gestellt wurden. Ihre Schaltungscharakteristiken sind in Tabelle 1 aufgeführt. Die Spalte *design* enthält den Namen der Schaltung, die Spalte *# gates* die Anzahl der Gatter, *# PPO* gibt die Anzahl der pseudoprimären Ausgänge an, *k* die Anzahl der Prüfpfade und *t* ihre maximale Länge.

| design | # gates | # PPO | k | t |
|---|---|---|---|---|
| p100k | 84356 | 5829 | 270 | 53 |
| p141k | 152808 | 10502 | 264 | 45 |
| p239k | 224597 | 18495 | 360 | 61 |
| p259k | 298796 | 18495 | 360 | 61 |
| p267k | 238697 | 16621 | 360 | 62 |
| p269k | 239771 | 16621 | 360 | 62 |
| p279k | 257736 | 17827 | 385 | 59 |
| p286k | 332726 | 17835 | 385 | 60 |
| p295k | 249747 | 18521 | 330 | 62 |
| p378k g 341315 | 17420 | 325 | 64 | |

**Tabelle 1** Schaltungscharakteristiken.

## 5.1 Testmustererzeugung

In dem beschriebenen mixed mode BIST Szenario wurden zunächst 4096 Zufallsmuster angelegt, die von einem LFSR generiert werden können. Für die ursprüngliche Schaltung und die Schaltung mit $t$ angehängten Paritätsbäumen sind dann die nicht zufallstestbaren Fehler ermittelt und dafür deterministische Testmuster generiert worden. Um die Testlänge zu minimieren, wird das LFSR angehalten, sobald der letzte zufallstestbare Fehler erkannt wurde.

Die Ergebnisse der Testmustererzeugung sind in Tabelle 2 dargestellt. In der Spalte *Original* stehen die Ergebnisse für die ursprüngliche Schaltung, in der Spalte *XP-SISR* für die Schaltung mit angehängtem XP-SISR. In der Unterspalte *# rp* ist die Anzahl der Zufallsmuster für jede Schaltung angegeben. Die resultierende Fehlerabdeckung für beide Schaltungsvarianten ist in Unterspalte *r fc* angegeben. Die Ergebnisse zeigen, dass die Sättigung noch nicht erreicht ist und für die meisten Schaltungen noch zufallstestbare Fehler unerkannt bleiben, wenn man nur 4096 Zufallsmuster anlegt. Dennoch werden keine weiteren Zufallsmuster generiert, da die verbleibenden Fehler auch von den deterministischen Mustern erkannt werden, die für die nicht zufallstestbaren Fehler angewendet werden müssen. Auf diesem Wege spart man zusätzliche Testzeit ein, während das Testdatenvolumen gleich bleibt. Nur eine Schaltung in der Menge ist vollständig zufallstestbar.

In den Unterspalten *# p* und *fc* sind die endgültige Größe der Testmustermenge und die endgültige Fehlerabdeckung angegeben. Für einige Schaltungen sieht man einen leichten Zuwachs der Testmustermenge für den hier vorgeschlagenen BISD Ansatz. Insgesamt wird die Musteranzahl weniger als zweimal so groß wie im Original. Wie in Abschnitt 4 beschrieben lässt sie sich dennoch effizient kodieren, so dass die Größe des Testmusterspeichers nicht steigt. Für die meisten Schaltungen kann außerdem die Fehlerabdeckung erhalten bleiben, in wenigen Fällen gibt es einen vernachlässigbar geringen Verlust.

## 5.2 Diagnose

Das XP-SISR wird mit zwei weiteren, häufig verwendeten Schemata verglichen. Das erste beruht auf einer reinen

| | | Original | | | XP-SISR | | |
|---|---|---|---|---|---|---|---|
| design | # rp | fc r | # p | fc | fc r | # p | fc |
| p100k | 4088 | 90.46 | 5397 | 99.56% | 90.45 | 5726 | 99.56 |
| p141k | 4091 | 89.80 | 5642 | 98.86% | 89.79 | 6712 | 98.86 |
| p239k | 4096 | 94.22 | 4778 | 98.84% | 94.21 | 7170 | 98.84 |
| p259k | 4095 | 94.72 | 4919 | 99.10% | 94.72 | 7702 | 99.10 |
| p267k | 4096 | 88.52 | 5191 | 99.60% | 88.48 | 8505 | 99.56 |
| p269k | 4095 | 88.45 | 5164 | 99.60% | 88.40 | 8550 | 99.56 |
| p279k | 4096 | 87.63 | 5360 | 97.89% | 87.51 | 9212 | 97.78 |
| p286k | 4094 | 87.72 | 6224 | 98.34% | 87.71 | 10524 | 98.34 |
| p295k | 4096 | 73.32 | 7916 | 99.15% | 73.31 | 12317 | 99.15 |
| p378k | 664 | 100.00 | 664 | 100.00% | 100.00 | 664 | 100.00 |

**Tabelle 2** Testmustermengen und Fehlerabdeckung.

Raumkompaktierung und speichert für jeden Prüfvektor ein kurzes Codewort ab. Es wurde X-Compact mit einer Länge von 10 Bit gewählt, wobei die X-Toleranz als 0 angenommen wurde [25]. Für dieses Schema bestehen die abzuspeichernden, fehlerhaften Antworten aus der Adresse des fehlerhaften Prüfvektors und dem fehlerhaften Codewort.

Das zweite Schema kombiniert Raum- und Zeitkompaktierung. Es besteht aus einem MISR mit einem vorgeschalteten BCH-Raumkompaktor mit Korrekturfaktor 2 [40]. Für dieses Schema und für das XP-SISR bestehen die Fehlerdaten aus den fehlerhaften Signaturen und der zugehörigen Musteradresse.

Um die Diagnosefähigkeit des Ansatzes zu bewerten, wurden je 200 Haftfehler, 200 Verzögerungsfehler und 200 Fehler durch Übersprechen zufällig und gleich verteilt in die Schaltungen injiziert. Die Diagnose wurde dann auf den ersten 50 fehlerhaften Antworten ausgeführt. Die Ergebnisse sind in Tabelle 3 dargestellt.

Für jedes der drei Kompaktierungsschemata ist in den letzten beiden Spalten die Kompaktierungsrate (CR) und die Größe des Antwort Speichers in Kilo Byte aufgeführt. Das Datenvolumen des Antwortspeichers für X-Compact ist nicht vertretbar, da nur Raumkompaktierung eingesetzt wird. Die Kompaktierungsraten sind für das XP-SISR im Schnitt dreimal höher als für das BCH-MISR und die Größe des Antwortspeichers, der zur Implementierung des BISD-Ansatzes bereitgestellt werden muss, ist nur halb so groß wie für das BCH-MISR. Das XP-SISR ist folglich der platzeffizienteste Kompaktierungsansatz.

Als Gütekriterium für die Diagnose ist die Prozentzahl der perfekten Übereinstimmung angegeben. Dies bedeutet, dass der erste Kandidat in der sortierten Liste der möglichen Kandidaten den Defekt tatsächlich erklärt. Für die drei Kompaktorarchitekturen ist außerdem der Unterschied in der diagnostischen Auflösung zur ursprünglichen Schaltung angegeben. In der letzten Zeile ist das gewichtete Mittel der diagnostischen Auflösung über alle Schaltungen aufgeführt, wobei mit der Anzahl der Gatter gewichtet wird.

Es ist zu beobachten, dass für das X-Compact die Auflösung im Vergleich zu der Originalschaltung erhalten bleibt. Beim BCH-MISR ist ein Verlust an diagnostischer Auflösung zu verzeichnen. Das XP-SISR übertrifft alle anderen Architekturen in seiner diagnostischen Auflösung. Die Gründe hierfür sind zum einen, dass die Testmustermenge größer ist, und zum anderen, dass sie eine bessere Auflösung hat. Der Grund hierfür liegt wiederum darin, dass während der Testmustererzeugung für die Schaltung mit Paritätsbaum das ATPG Werkzeug versucht, den Fehler zu möglichst vielen Paritätsbits zu propagieren, um Aliasing zu vermeiden. Folglich sensibilisieren diese Muster mehr observierende Ausgänge als die Muster, die für die ursprüngliche Schaltung generiert wurden und so können mehr diagnostische Informationen gesammelt werden.

Als Nebeneffekt der Diagnoseexperimente gewinnt man eine Abschätzung der Defektabdeckung für Defekte, die nicht durch Haftfehler repräsentiert werden. Aus den Ergebnissen der Diagnose kann man den Anteil der detektierbaren Verzögerungsfehler und Fehler durch Übersprechen ableiten. Diese sind in Tabelle 4 aufgelistet. Es sind die Defektabdeckung für die Originalschaltung (Spalte *Original*), die Schaltung mit X-Compactor (Spalte *X-Compact*), das MISR mit BCH Kompaktor (Spalte *BCH-MISR*) und die Defektabdeckung für das hier vorgeschlagene XP-SISR (Spalte *XP-SISR*) gezeigt.

| design | Original | X-Compact | CR | [KB] | BCH-MISR | CR | [KB] | XP-SISR | CR | [KB] |
|---|---|---|---|---|---|---|---|---|---|---|
| p100k | 72.7% | 72.7% 0.0 | 10 X | 357.6 | 71.7% –1.0 | 306 X | 12.8 | 71.8% –0.8 | 971 X | 4.3 |
| p141k | 73.5% | 73.5% 0.0 | 23 X | 317.4 | 73.5% 0.0 | 552 X | 13.4 | 74.0% +0.5 | 1750 X | 5.0 |
| p239k | 81.5% | 81.5% 0.0 | 30 X | 364.3 | 81.5% 0.0 | 973 X | 11.3 | 82.3% +0.8 | 3082 X | 5.4 |
| p259k | 75.7% | 75.7% 0.0 | 30 X | 375.1 | 75.5% –0.2 | 973 X | 11.7 | 77.2% +1.5 | 3082 X | 5.8 |
| p267k | 67.0% | 67.0% 0.0 | 26 X | 402.3 | 67.3% +0.3 | 874 X | 12.3 | 74.8% +7.8 | 2770 X | 6.4 |
| p269k | 66.3% | 66.3% 0.0 | 26 X | 400.2 | 64.5% –1.8 | 874 X | 12.3 | 72.8% +6.5 | 2770 X | 6.4 |
| p279k | 68.7% | 68.7% 0.0 | 30 X | 395.3 | 68.7% 0.0 | 938 X | 12.7 | 70.3% +1.7 | 2971 X | 6.9 |
| p286k | 65.3% | 65.3% 0.0 | 29 X | 466.8 | 65.3% 0.0 | 938 X | 14.8 | 71.0% +5.7 | 2972 X | 7.9 |
| p295k | 56.2% | 56.0% –0.2 | 29 X | 613.5 | 54.0% –2.2 | 974 X | 18.8 | 55.3% –0.8 | 3086 X | 9.2 |
| p378k | 85.8% | 85.8% 0.0 | 27 X | 53.1 | 85.8% 0.0 | 916 X | 1.6 | 85.2% –0.7 | 2903 X | 0.5 |
| avg. | 71.4% | 71.4% –0.0 | | | 71.0% –0.4 | | | 73.9% +2.5 | | |

**Tabelle 3** Diagnostische Auflösung nach 600 Fehlerinjektionen.

Man sieht, dass das XP-SISR eine höhere Defekterfassung erzielt als die anderen Ansätze. Dies kann wiederum mit der höheren Auflösung der Testmustermenge für das XP-SISR erklärt werden.

| design | Original | X-Compact | BCH-MISR | XP-SISR |
|---|---|---|---|---|
| p100k | 93.2 | 93.2 | 93.2 | 95.0 |
| p141k | 91.5 | 91.5 | 91.5 | 93.2 |
| p239k | 93.0 | 93.0 | 93.0 | 95.2 |
| p259k | 92.2 | 92.2 | 92.2 | 96.8 |
| p267k | 91.2 | 91.2 | 91.2 | 94.2 |
| p269k | 88.5 | 88.5 | 88.5 | 93.8 |
| p279k | 88.0 | 88.0 | 88.0 | 91.2 |
| p286k | 86.2 | 86.2 | 86.2 | 92.5 |
| p295k | 78.0 | 78.0 | 78.0 | 82.8 |
| p378k | 98.8 | 98.8 | 98.8 | 98.8 |

**Tabelle 4** Abschätzung der Defektabdeckung für Verzögerungsfehler und Fehler durch Übersprechen.

Der vorgeschlagene Ansatz zur eingebauten Selbstdiagnose verändert die Hardware Kosten zur Testmustererzeugung nicht und bietet eine gleich bleibende Fehlerabdeckung für Haftfehler im Vergleich zu der ursprünglichen Schaltung ohne Kompaktierung. Außerdem bietet er im Vergleich zu X-Compact und BCH-MISR eine höhere Defekterfassung für Defekte, die nicht durch das Haftfehlermodell beschrieben werden, und eine höhere diagnostische Auflösung auch im Vergleich zu einem externen Test.

## 5.3 Datenvolumen und Hardwareaufwand

Das Datenvolumen stellt den Hauptaspekt der vorliegenden Arbeit dar. Im Vergleich zu den bisher im BIST verwendeten Kompaktorschemata bringt das XP-SISR erstmalig die Voraussetzungen zusammen, die notwendig sind, um die Soll-Signaturen und fehlerhaften Ist-Signaturen auf dem zu testenden Chip zu speichern.

In Tabelle 5 ist die Größe des Fehlerspeichers und des Antwortspeichers mit der Größe des Speichers verglichen, der zum Erzeugen der deterministischen Testmuster notwendig ist. Als Vergleich wird die Speichergrößen in [7] herangezogen. In der Spalte *Muster* ist die Größe des Musterspeichers in Kilo Byte angegeben. In der Spalte *Antwort* wird die Größe des Antwort- und Fehlerspeichers angegeben. Die letzte Spalte gibt den Mehraufwand in Prozent an. Für die Schaltung p100k ist die Mustermenge außerordentlich gut komprimierbar, weshalb der Mehraufwand zur Speicherung der Selbstdiagnose-Daten hier mit 60% deutlich höher ist, als für alle anderen Schaltungen. In den meisten Fällen ist der Aufwand für die hier untersuchten Schaltungen kleiner als 15%.

Es ist außerdem zu beobachten, dass der Anteil der zu speichernden Bits für die Antworten in Bezug zu den zu speichernden Bits für die Muster mit wachsender Schaltungsgröße sinkt. Dies liegt daran, dass das Volumen der Antworten logarithmisch mit der Schaltungsgröße wächst (das SISR ist $\lceil \log t \rceil$ Bits lang), während die Anzahl der Care bits in den Testmustern linear wächst. Mit wachsender Schaltungsgröße verringert sich der Mehraufwand für die eingebaute Selbstdiagnose also weiter.

| design | Muster ([7]) [KB] | Antwort [KB] | Mehraufwand (%) |
|---|---|---|---|
| p100k | 7.25 | 4.41 | 60.83% |
| p141k | 36.18 | 5.15 | 14.24% |
| p239k | 17.97 | 5.50 | 30.58% |
| p259k | 23.54 | 5.89 | 25.04% |
| p267k | 47.95 | 6.50 | 13.55% |
| p269k | 47.44 | 6.53 | 13.77% |
| p279k | 48.37 | 7.03 | 14.53% |
| p286k | 63.69 | 8.01 | 12.58% |
| p295k | nicht verfügbar | 9.36 | nicht verfügbar |
| p378k | nicht verfügbar | 0.59 | nicht verfügbar |

**Tabelle 5** Mehraufwand für Fehler- und Antwortspeicher.

In Tabelle 6 sind die Diagnoseergebnisse für die drei Kompaktorarchitekturen gezeigt, wenn der Antwortspeicher in seiner Größe auf die des XP-SISRs begrenzt ist. Das bedeutet, es ist für jeden Kompaktor $s$ nur das Anwenden von $x$ Testmustern erlaubt, wobei

$$x \cdot SignaturBreite(s) = |T| \cdot SignaturBreite(\text{XP-SISR}).$$

| design | X-Compact | BCH-MISR | XP-SISR |
|---|---|---|---|
| p100k | 32.7% | 56.5% | 71.8% |
| p141k | 39.3% | 64.2% | 74.0% |
| p239k | 55.0% | 75.2% | 82.3% |
| p259k | 48.0% | 70.2% | 77.2% |
| p267k | 25.5% | 52.7% | 74.8% |
| p269k | 30.2% | 50.3% | 72.8% |
| p279k | 31.7% | 57.5% | 70.3% |
| p286k | 32.3% | 55.5% | 71.0% |
| p295k | 20.8% | 37.2% | 55.3% |
| p378k | 57.5% | 80.3% | 85.2% |

**Tabelle 6** Diagnostische Auflösung mit eingeschränkter Antwort Speicher Größe.

Der Verlust in diagnostischer Auflösung für das X-Compact und BCH-MISR ist nicht vertretbar und das benötigte Antwortvolumen für diese Ansätze ist folglich nicht weiter reduzierbar.

Das XP-SISR-Schema ermöglicht somit erstmalig eine BIST basierte Diagnose für freie Logik, die vernachlässigbaren Hardwareaufwand benötigt und zusätzliche Testsitzungen vermeidet.

# 6 Fazit

Es wurde eine neue Architektur für die eingebaute Selbstdiagnose (BISD) vorgestellt, die erstmalig eine dem Speicherselbsttest ähnliche Diagnose für freie Logik ermöglicht. Der Hauptbestandteil des Schemas ist eine Testantwortkompaktierung, die eine sehr hohe Kompaktierungsrate erlaubt und so das Speichern der Soll-Antworten und der fehlerhaften Antworten auf dem Chip ermöglicht. Die fehlerhaften Antworten, die später als Eingabe für den zugehörigen Diagnosealgorithmus dienen, können in einer einzigen BIST Sitzung gesammelt werden. Somit ist Volumen Diagnose auch in BIST Umgebungen möglich, ohne Testsitzungen für fehlerhafte Prüflinge wiederholen zu müssen. Experimente mit großen, industriellen Schaltungen zeigen, dass eine hohe Defekterfassung möglich ist, eine hohe diagnostische Auflösung unabhängig vom Defektmechanismus erreicht wird und nur geringer Mehraufwand für das Speichern der Antworten notwendig ist.

# 7 Literatur


[1] H.-J. Wunderlich, "BIST for Systems-on-a-Chip," *INTEGRATION, the VLSI journal*, vol. 26, no. 1-2, pp. 55–78, 1998.

[2] ITRS, "International Technology Roadmap for Semiconductors 2007 Edition - Test and Test Equipment," 2007.

[3] S. Pateras, "IP for Embedded Diagnosis," *IEEE Design & Test of Computers.*, vol. 19, pp. 44–53, May/Jun 2002.

[4] N. A. Touba and E. J. McCluskey, "Altering a Pseudo-Random Bit Sequence for Scan-Based BIST," in *Proceedings of the IEEE International Test Conference.*, pp. 167–175, 1996.

[5] H.-J. Wunderlich and G. Kiefer, "Bit-Flipping BIST," in *Proceeding of the International Conference on Computer-Aided Design (ICCAD)*, pp. 337–343, 1996.

[6] G. Kiefer, H. Vranken, E. J. Marinissen, and H.-J. Wunderlich, "Application of Deterministic Logic BIST on Industrial Circuits," in *Proceedings of the 31st IEEE International Test Conference.*, 2000.

[7] A.-W. Hakmi, S. Holst, H.-J. Wunderlich, J. Schloeffel, F. Hapke, and A. Glowatz, "Restrict Encoding for Mixed–Mode BIST," in *Proceedings of the IEEE VLSI Test Symposium.*, 2009.

[8] M. Naruse, I. Porneranz, S. Reddy, and S. Kundu, "On-Chip Compression of Output Responses with Unknown Values Using LFSR Reseeding," in *Proceedings of the International Test Conference.*, vol. 1, pp. 1060–1068, 30-Oct. 2, 2003.

[9] Y. Tang, H.-J. Wunderlich, H. P. E. Vranken, F. Hapke, M. Wittke, P. Engelke, I. Polian, and B. Becker, "X-Masking During Logic BIST and Its Impact on Defect Coverage," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 442–451, February 2006.

[10] P. Wohl, J. Waicukauski, and S. Patel, "Scalable Selector Architecture for X-Tolerant Deterministic BIST," in *41st Design Automation Conference.*, pp. 934–939, 2004.

[11] R. Aitken and V. Agarwal, "A Diagnosis Method Using Pseudo-Random Vectors without Intermediate Signatures," in *Digest of Technical Papers of the IEEE International Conference on Computer-Aided Design.*, pp. 574–577, Nov 1989.

[12] Y. Zorian and V. Agarwal, "A General Scheme To Optimize Error Masking in Built-in Self-Testing," *Digest of Papers*, p. 410, 1986.

[13] J. Waicukauski and E. Lindbloom, "Failure Diagnosis of Structured VLSI," *Design & Test of Computers, IEEE*, vol. 6, pp. 49–60, Aug 1989.

[14] I. Bayraktaroglu and A. Orailoglu, "Cost-Effective Deterministic Partitioning for Rapid Diagnosis in Scan-Based BIST," *IEEE Design & Test of Computers.*, vol. 19, pp. 42–53, Jan/Feb 2002.

[15] J. Ghosh-Dastidar and N. Touba, "A Rapid and Scalable Diagnosis Scheme for BIST Environments with a Large Number of Scan Chains," in *Proceedings of the IEEE VLSI Test Symposium.*, pp. 79–85, 2000.

[16] Y. Nakamura, T. Clouqueur, K. Saluja, and H. Fujiwara, "Diagnosing At-Speed Scan BIST Circuits Using a Low Speed and Low Memory Tester," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems.*, vol. 15, pp. 790–800, July 2007.

[17] J. Rajski and J. Tyszer, "Diagnosis of Scan Cells in BIST Environment," *IEEE Transactions on Computers.*, vol. 48, pp. 724–731, Jul 1999.

[18] P. Wohl, J. Waicukauski, S. Patel, and G. Maston, "Effective Diagnostics through Interval Unloads in a BIST Environment," in *Proceedings of the Design Automation Conference.*, pp. 249–254, 2002.

[19] A. Leininger, M. Goessel, and P. Muhmenthaler, "Diagnosis of Scan-Chains by Use of a Configurable Signature Register and Error-Correcting Codes," in *Proceedings of the Design, Automation and Test in Europe Conference.*, vol. 2, pp. 1302–1307, 2004.

[20] C. Liu, K. Chakrabarty, and M. Goessel, "An Interval-Based Diagnosis Scheme for Identifying Failing Vectors in a Scan-BIST Environment," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition.*, pp. 382–386, 2002.

[21] Y. Wu and S. Adham, "Scan-Based BIST Fault Diagnosis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*, vol. 18, pp. 203–211, Feb 1999.

[22] S. Holst and H.-J. Wunderlich, "A Diagnosis Algorithm for Extreme Space Compaction," in *Proceedings of the European Design Automation and Test Conference and Exhibition*, 2009.

[23] J. Rajski, J. Tyszer, C. Wang, and S. Reddy, "Finite Memory Test Response Compactors for Embedded Test Applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*, vol. 24, pp. 622–634, April 2005.

[24] N. Touba, "X-Canceling MISR - An X-Tolerant Methodology for Compacting Output Responses with Unknowns Using a MISR," in *IEEE International Test Conference.*, pp. 1–10, Oct. 2007.

[25] S. Mitra and K. S. Kim, "X-Compact: an Efficient Response Compaction Technique," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,*, vol. 23, pp. 421–432, March 2004.

[26] K. K. Saluja and M. Karpovsky, "Testing Computer Hardware through Data Compression in Space and Time," in *Proceedings of the IEEE International Test Conference.*, pp. 83–89, 1983.

[27] J. Patel, S. Lumetta, and S. Reddy, "Application of Saluja-Karpovsky Compactors to Test Responses with Many Unknowns," in *Proceedings of the 21st VLSI Test Symposium.*, pp. 107–112, 2003.

[28] M. Abramovici and M. Breuer, "Fault Diagnosis Based on Effect-Cause Analysis: An Introduction," in *17th Conference on Design Automation.*, pp. 69–76, June 1980.

[29] R. Desineni, O. Poku, and R. D. Blanton, "A Logic Diagnosis Methodology for Improved Localization and Extraction of Accurate Defect Behavior," in *IEEE International Test Conference.*, pp. 1–10, Oct. 2006.

[30] M. E. Amyeen, D. Nayak, and S. Venkataraman, "Improving Precision Using Mixed-level Fault Diagnosis," in *IEEE International Test Conference.*, pp. 1–10, Oct. 2006.

[31] W.-T. Cheng, M. Sharma, T. Rinderknecht, L. Lai, and C. Hill, "Signature Based Diagnosis for Logic BIST," in *Proceedings of the IEEE International Test Conference (ITC '06).*, pp. 1–9, Oct. 2006.

[32] T. Bartenstein, D. Heaberlin, L. M. Huisman, and D. Sliwinski, "Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm.," in *Proceedings IEEE International Test Conference 2001, Baltimore, MD, USA, 30 October - 1 November 2001*, pp. 287–296, 2001.

[33] S. Holst and H.-J. Wunderlich, "Adaptive Debug and Diagnosis Without Fault Dictionaries," in *Proceedings European Test Symposium.*, 2007.

[34] B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs," in *Proceedings of the European Test Conference.*, pp. 237–242, 1991.

[35] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Transactions on Computers*, vol. 44, no. 2, pp. 223–233, 1995.

[36] K. Miyase and S. Kajihara, "XID: Don't Care Identification of Test Patterns for Combinational Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*, vol. 23, pp. 321–326, Feb. 2004.

[37] A. El-Maleh and A. Al-Suwaiyan, "An Efficient Test Relaxation Technique for Combinational & Full-Scan Sequential Circuits," in *Proceedings of the 20th IEEE VLSI Test Symposium.*, pp. 53–59, 2002.

[38] M. Kochte, C. Zoellin, M. Imhof, and H.-J. Wunderlich, "Test Set Stripping Limiting the Maximum Number of Specified Bits," in *4th IEEE International Symposium on Electronic Design, Test and Applications.*, pp. 581–586, Jan. 2008.

[39] M. A. Kochte, S. Holst, M. Elm, and H.-J. Wunderlich, "Test encoding for extreme response compaction," in *Proceedings of the 14th IEEE European Test Symposium*, pp. 155–160, 2009.

[40] T. Reungpeerakul, X. Qian, and S. Mourad, "BCH-based Compactors of Test Responses with Controllable Masks.," in *15th Asian Test Symposium, 2006. ATS '06, Nov. 2006, Fukuoka*, pp. 395–401, 2006.