Reduktion der Verlustleistung beim Selbsttest durch Verwendung testmengenspezifischer Information

Michael E. Imhof, Hans-Joachim Wunderlich, Christian G. Zöllin
Institut für Technische Informatik
Universität Stuttgart
Pfaffenwaldring 47, D-70569 Stuttgart, Germany
email: {imhof, wu, zoellin}@iti.uni-stuttgart.de

Jens Leenstra, Nicolas Mäding
IBM Deutschland Entwicklung
Schönaicherstr. 220, D-71032 Böblingen, Germany
email: {leenstra, nmaeding}@de.ibm.com

***Zusammenfassung*— Der während des Selbsttests von Schaltungen mit deaktivierbaren Prüfpfaden verwendete Testplan entscheidet über die Verlustleistung während des Tests. Bestehende Verfahren zur Erzeugung des Testplans verwenden überwiegend topologische Information, zum Beispiel den Ausgangskegel eines Fehlers. Aufgrund der implizit gegebenen Verknüpfung zwischen Testplan und Mustermenge ergeben sich weitreichende Synergieeffekte durch die Ausschöpfung mustermengenabhängiger Informationen. Die Verwendung von testmengenspezifischer Information im vorgestellten Algorithmus zeigt bei gleichbleibender Fehlererfassungsrate und Testdauer deutliche Einsparungen in der benötigten Verlustleistung. Das Verfahren wird an industriellen und Benchmark-Schaltungen mit bestehenden, überwiegend topologisch arbeitenden Verfahren verglichen.**

I. Einführung

Während des Tests ist die Schaltaktivität und infolgedessen die durchschnittliche dynamische Verlustleistung von integrierten Systemen gegenüber der Verlustleistung im Systembetrieb um eine Größenordnung erhöht [1]. Zusätzlich zur wachsenden statischen Leistungsaufnahme muss dieser Effekt berücksichtigt werden, um eine Beeinträchtigung der Ausbeute und Zuverlässigkeit zu verhindern [2], [3]. Die häufig im Produktionstest verwendeten Lösungen umfassen die Reduktion der Schiebegeschwindigkeit, die Partitionierung der Schaltung und dedizierte Kühllösungen. Diese Methoden bringen hohe Kosten mit sich und können die Testqualität beeinträchtigen. Für Systems-on-a-Chip wurden Strategien zum Testscheduling und zur Testplanerzeugung vorgeschlagen, um bei Einhaltung eines maximalen Leistungsbudgets einen effizienten Test aller Module zu ermöglichen [1], [4], [5].

Für prüfpfad-basiertes Testen existiert eine Vielzahl an Methoden, die Schaltaktivität während des Schiebens der Muster und der Erfassung der Antworten zu reduzieren. Diese umfassen spezielle Flip Flop-Typen, die das Schalten des Ausgangs während dem Schieben unterdrücken, die Maskierung von Mustern, welche nicht zur Fehlererfassung beitragen, und das Abschalten des Schiebetakts während nutzloser Takte [6], [7], [8].

Normalerweise werden mehrere Prüfpfade benutzt, um einen eingebauten Selbsttest (BIST), eingebetteten deterministischen oder externen Test zu implementieren. Falls zu bestimmten Zeitpunkten nur eine Teilmenge der Prüfpfade aktiviert ist, können daraus signifikante Einsparungen in der Verlustleistung abgeleitet werden [9]. Eine ähnliche Methode wurde für den Test der Cell Broadband Engine™ (Cell Prozessor) vorgestellt [10] und in [11] verfeinert. Das zugrunde liegende Problem der Testplanerstellung ist komplexer als die Testplanerstellung für SoCs, da die Fehlerüberdeckung, die Testzeit und die Verlustleistung gleichzeitig betrachtet werden müssen.

Die in [10] vorgestellte Technik zur Testplanerzeugung benutzt topologische Information, um einen Testplan zu erzeugen. In [11] wird die Testplanerzeugung auf ein Überdeckungsproblem abgebildet und mit einer "Teile & Herrsche"-Heuristik eine Lösung konstruiert. Aus Komplexitätsgründen wird dort nur für eine kleine Teilmenge der Fehler testmengenspezifische Information verwendet, die von den üblichen Werkzeugen für Test und Diagnose generiert wird. Dazu wird dort für besonders schwer zufallstestbare Fehler das erkennende Muster analysiert, und daraus werden die zu aktivierenden pseudoprimären Eingänge und der zu beobachtende pseudoprimäre Ausgang bestimmt, während für die restlichen Fehler auf topologische Information, den sogenannten "Support" [12] zurückgegriffen wird.

Der vorliegende Beitrag zeigt, daß ohne weiteren Rechenaufwand deutlich bessere Ergebnisse erzielt werden können, wenn noch weitere Diagnoseinformationen verwendet werden. Diagnose wird zumeist unter Verwendung eines Fehlerwörterbuchs durchgeführt [13], [14]. Für jeden Fehler kann daraus die Information entnommen werden, wann und wo er observierbar ist, und eine Menge von Mustern und zugehörigen Flip Flops extrahiert werden. Diese Information wird im hier vorgestellten Verfahren konsequent ausgenutzt.

Der Rest des Beitrags ist wie folgt aufgebaut: In Abschnitt 2 geben wir eine genaue Definition des Problems sowie dessen Abbildung auf ein allgemeines Überdeckungsproblem. Es wird ein Verfahren vorgestellt, mit dem eine kostengünstige Überdeckung effizient approximiert werden kann. In Abschnitt 3 bewerten wir die vorgestellte Methode an Hand der verbreiteten Benchmarkschaltungen sowie für verschiedene industrielle Schaltungen.

II. Berechnung eines optimierten Testplans

Das Ziel eines optimierten Testplans ist die Detektierung einer gegebenen Fehlermenge mit minimierter Leistungsaufnahme. Für jeden Startwert des linear rückgekoppelten Schieberegisters (LFSR) wird eine Menge von Prüfpfaden bestimmt, die abgeschaltet werden kann, ohne die Fehlerüberdeckung zu beeinträchtigen. Hierzu wird im Folgenden auf ein Überdeckungsproblem abgebildet, wobei der Rechenaufwand begrenzt wird, indem ein Teile-und-Herrsche Ansatz verwendet wird.

Für den Selbsttest mit mehreren Prüfpfaden wurde in [15] die STUMPS-Konfiguration (Abb. 1) vorgestellt, welche inzwischen die am weitesten verbreitete Struktur für den Logik-Selbsttest ist [16], [17]. Hierbei werden mehrere parallele Prüfpfade von einem Generator für Pseudo-Zufallsmuster (Pseudo-Random Pattern Generator, PRPG) mit Testmustern versorgt. Der PRPG besteht aus einem linear rückgekoppelten Schieberegister (Linear-Feedback Shift Register, LFSR), einem XOR-Netzwerk zur Phasenverschiebung und einer Logik zur Gewichtung der Häufigkeit von Einsen und Nullen im Testmuster [18].

Erweiterungen der STUMPS-Struktur gestatten es, den Schiebevorgang für einzelne Prüfpfade zu deaktivieren. Zum Beispiel erlaubt der Cell Prozessor, das Taktsignal für individuelle Prüfpfade gänzlich zu unterdrücken [19] und dadurch nicht nur die durch den Schiebevorgang verursachte Schaltaktivität zu blockieren, sondern auch die Verlustleistung im Taktbaum zu verringern. Der Cell Prozessor benutzt mehrere Testregister (Abb. 1) über die ein Tester Zugriff auf die Werte für den LFSR-Startwert ($seed$), die Gewichtung ($weight$) sowie die Prüfpfadaktivierung ($scanena$) hat. Eine ausführliche Beschreibung der Selbsttestausstattung des Cell Prozessors wurde in [10] vorgestellt, und ähnliche Eigenschaften sind auch in anderen großen industriellen Schaltungen implementiert.

Ein Testblock ist ein Tupel (s, SC_b) bestehend aus einem Startwert für die Mustererzeugung und einer Menge aktivierter Prüfpfade $SC_b \subset SC$, auch Konfiguration genannt. Jeder Block $b = (s, SC_b)$ bestimmt eine Menge von Fehlern F_b, die bei Abarbeitung dieses Blocks erkannt werden. Ein Startwert entspricht einer zugeordneten Testmenge von fester Größe N, die Menge der Startwerte S sei gegeben. Für jeden Testblock b kann eine Fehlermenge F_b bestimmt werden, die durch diesen Block detektiert werden können. Gesucht ist eine Menge von Blöcken B, die eine Fehlermenge F mit minimierter Leistungsaufnahme überdeckt. Ein Fehler f kann durch unterschiedliche Konfigurationen überdeckt werden und es können mehrere Blöcke für einen Startwert existieren. Diese unterscheiden sich nur in ihrer Konfiguration und können verschiedene Fehlermengen überdecken.

Die Kosten einer Übderdeckung sind eine Schätzung ihres Energieverbrauchs, der durch Aufsummieren der Anzahl aktiver Prüfpfade für alle Startwerte bestimmt wird. Eine genauere aber deutlich aufwendigere Schätzung kann durch die Berechnung einer gewichteten Schaltaktivität pro Block während der Schiebe- und Beobachtungszyklen bestimmt werden. Sei S_B eine Menge von Startwerten für B. Dann ist B_s für jeden Startwert $s \in S_B$ die zugehörige Blockmenge. Die Kosten einer Blockmenge können nun abgeschätzt werden zu

$$Kosten(B) = \sum_{s \in S_B} \left| \bigcup_{(s,sc) \in B_s} sc \right|.$$

A. Generierung des Überdeckungsproblems

Sei $F_i \subset F$ die betrachtete Teilmenge der zu detektierenden Fehler in einem Schritt der Teile-und-Herrsche Heuristik. Für jeden der Fehler $f \in F_i$ wird mit Hilfe einer Funktion $c(f, s_f)$ zu jedem Startwert $s_f \in S_f$, $S_f = \{s \in S \mid s \text{ detektiert } f\}$ eine Menge von minimalen Blöcken

$$B_{f,s_f} = \bigcup_{c \in c(f,s_f)} \{(s_f, c)\}$$

ermittelt.

Da ein einziges Ausgangs-Flip Flop für die Erkennung eines Fehlers ausreichend ist, enthält jede Konfiguration in $c(f, s_f)$

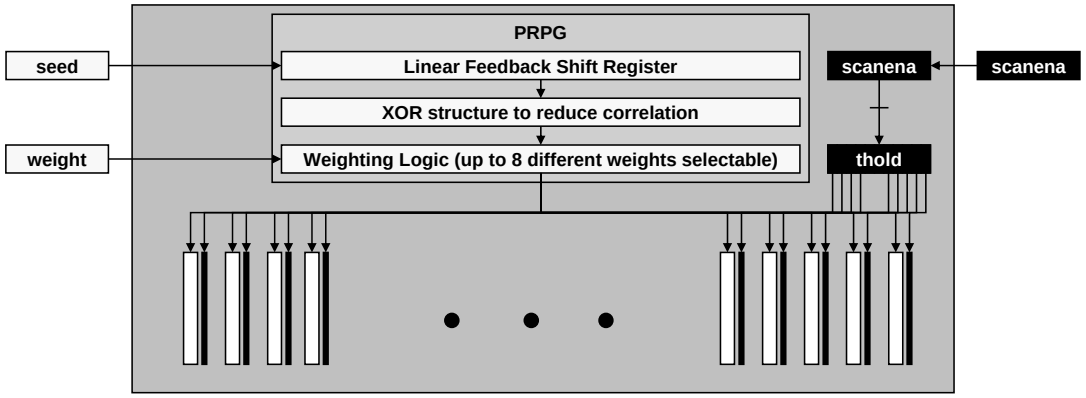

Fig. 1. STUMPS Selbsttest-Konfiguration implementiert im Cell-Prozessor

nur einen Ausgangsprüfpfad und eine minimale Menge von Eingangsprüfpfaden. Für alle durch den Startwert s_f generierten Muster findet sich die Information über die Ausgänge im Fehlerwörterbuch. Anschließend wird für jedes Flip Flop die Menge der Flip Flops in den zugehörigen Eingangskegeln gesammelt und die Menge der benötigten Prüfpfade bestimmt.

Die Menge

$$\bigcup_{f \in F_i} \bigcup_{s_f \in S_f} B_{f,s_f}$$

bildet nun das Überdeckungsproblem bzgl. F_i. Solange die Zahl der Blöcke gering ist, kann mit einem Branch-and-Bound Verfahren eine Teilmenge B_i gefunden werden [20], so daß $\bigcup_{b \in B_i} F_b$ alle Fehler aus F_i überdeckt und $Kosten(\bigcup_{j=1..i} B_j)$ minimal ist.

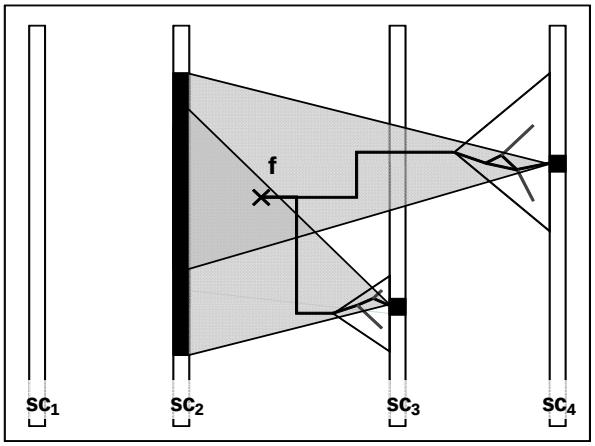

Fig. 2. Beispiel: f wird vom Startwert s in zwei Flip Flops detektiert, welche sich in den Prüfpfaden sc_3 und sc_4 befinden. Zusammen mit den jeweiligen Eingangskegeln ergeben sich dadurch die Konfigurationen $c(f,s) = \{\{sc_2, sc_3\}, \{sc_2, sc_4\}\}$ und damit die Blöcke $\{s, \{sc_2, sc_3\}\}$ und $\{s, \{sc_2, sc_4\}\}$.

B. Komplexitätsreduktion

Für große, industrielle Schaltungen ist das Finden eines globalen Optimums nicht durchführbar, da die exakte Lösung des Überdeckungsproblems zu rechenaufwendig sind. Im Folgenden präsentieren wir die Schritte der Teile-und-Herrsche Heuristik, welche das Überdeckungsproblem effizient löst. Zu diesem Zweck wird die Menge der Fehler in drei verschiedene Klassen unterteilt:

1) Harte Fehler sind solche, die nur durch einen Startwert aus S erkannt werden können. Es ist dadurch möglich jeweils ein Überdeckungsproblem pro Startwert zu generieren und jedes dieser Probleme getrennt zu lösen.

2) Schwer erkennbare Fehler können nur durch eine Zahl von Startwerten erkannt werden, die unter einer vom Anwender vorgegebenen Konstante lim liegt. Die Zahl zu bearbeitender schwer erkennbarer Fehler kann stark reduziert werden, indem zuvor eine Fehlersimulation der Zwischenlösung nach Schritt 1) durchgeführt wird. Hierbei wird angenommen, dass der Zustand deaktivierter Prüfpfade unbekannt (X) ist und Flip Flops in deaktivierten Prüfpfaden nicht observierbar sind. Dieser Pessimismus verhindert, dass zusätzlich aktivierte Prüfpfade in einem späteren Schritt das Ergebnis dieser Fehlersimulation ungültig machen.

3) Übrige Fehler sind leicht erkennbar, diese Restmenge kann jedoch sehr groß sein, insbesondere wenn besonders viele Pfade deaktiviert sind. Wiederum wird Fehlersimulation verwendet um die Zwischenlösung zu bewerten und bereits detektierte Fehler zu ermitteln. Die Zahl der Fehler kann hier aber zusätzlich reduziert werden, wenn der eigentliche Zustand eines deaktivierten Prüfpfads mit berücksichtigt wird. Für gewöhnlich wird das jeweils erste Muster eines Startwerts in alle Prüfpfade geschoben, um die Schaltung in einen definierten Zustand zu bringen, und die Fehlersimulation kann dies entsprechend berücksichtigen.

Übrige Fehler weisen eine hohe Zufallstestbarkeit auf, durch die entstehenden Freiheitsgrade ergibt sich auch eine große Zahl von Blöcken im zugrundeliegenden Überdeckungsproblem. Andererseits wird die erhöhte Zufallstestbarkeit in der Regel dazu führen, daß zur Fehlererkennung nur eine relativ kleine Zahl von Bits in der Testmenge zu spezifizieren ist und somit auch die Zahl der zu aktivierenden Prüfpfade relativ klein wird. Mit Fehlersimulation wird in der Testmenge ein Muster gesucht, welches eine möglichst kleine Zahl an zusätzlichen Prüfpfaden aktiviert. Pro Fehler erhält man somit den zugehörigen Block, insgesamt wird damit ein relativ kleines Übderdeckungsproblem konstruiert.

Hier unterscheidet sich die vorgestellte Methode erheblich von der in [11] vorgestellten. Durch die Verwendung eines möglichst kleinen Kegels im Gegensatz zum ganzen Supports eines Fehler können erhebliche Einsparungen erzielt werden. Dies wird im folgenden Ergebnisabschnitt gezeigt.

III. Ergebnisse

Das beschriebene Verfahren wurde in Java als Teil einer hauseigenen Entwurfsautomatisierungslösung implementiert und auf eine Reihe von Schaltungen angewendet. Die Schätzung der Leistungsaufnahme ergibt sich aus den Kosten der erzeugten Blockmenge.

A. Benchmarks und industrielle Schaltungen

Zur Evaluierung des vorgestellten Verfahrens wurden verschiedene Schaltungsmodelle verwendet (Tabelle I). Die Schaltungen aus ISCAS89 ($s*$) und ITC99 ($b*$) besitzen keine Selbsttestausstattung und wurden für diesen Beitrag um die benötigte Testarchitektur erweitert. Die von NXP zur Verfügung gestellten Schaltungen ($p*$) enthalten bereits einen prüfgerechten Entwurf mit parallelen Prüfpfaden. Sie repräsentieren die typischen Eigenschaften industrieller Schaltungen, nämlich kürzere Pfade sowie kleinere Ausgangskegel, bedingt durch

die stärkere Optimierung auf hohe Taktfrequenzen bei geringer Fläche.

Als Beispiel für die Anwendung des vorgestellten Verfahrens mit partiellen Prüfpfaden wurden die Synergistic Processing Elemente (SPE) des Cell-Prozessors verwendet, dessen übergeordnete Selbsttestarchitektur aus 15 Selbsttestdomänen mit jeweils eigener STUMPS Instanz [21] besteht.

Schaltung	**# Fehler**		**Leistungsaufnahme relativ zu** [10]	
	Alle	**Detektiert**	[11]	**Hier**
s38417	32320	31589	89%	73%
s38584	38358	36385	86%	19%
b17	81330	70300	73%	72%
b18	277976	240385	80%	80%
b19	560696	479834	87%	85%
p286k	648044	609609	93%	93%
p330k	547808	491079	91%	78%
p388k	856678	839075	86%	70%
p418k	688808	639787	89%	78%
p951k	1590490	1545320	95%	83%
SPE	1065190	904364	74%	51%

TABLE I
200 STARTWERTE, 1024 MUSTER PRO STARTWERT

B. Experimente

Tabelle I zeigt Ergebnisse für das vorgestellte Verfahren. Die dargestellten Werte sind auf die Werte aus [10] normiert um eine bessere Vergleichbarkeit der erreichten Reduktion zu ermöglichen. Die Anzahl der zufällig gewählten Startwerte liegt bei 200, die pro Startwert generierte Anzahl von Mustern beträgt 1024. Für die Klassifikation schwer erkennbarer Fehler wurden nur solche herangezogen, die durch maximal 3 Startwerte erkannt werden können.

Der erste Spaltenblock enthält den Schaltungsnamen, die Anzahl der Haftfehler sowie die Anzahl der durch die Startwerte erkennbaren Fehler. Der zweite Block zeigt die Verlustleistung für das Verfahren aus [11] sowie das hier vorgestellte Verfahren jeweils relativ zum topologischen Verfahren aus [10].

Während das in [11] vorgestellte Verfahren die Leistungsaufnahme auf bis zu 73% gegenüber [10] reduziert, ergeben sich mit dem hier vorgestellten Verfahren Reduktionen auf bis zu 19%. Hierbei werden die besten Ergebnisse fuer die Schaltungen erreicht, die besonders viele leicht selbst-testbare Fehler enthalten. Dies trifft speziell auf die großen, industriellen Schaltungen zu, wobei die Verlustleistung für die SPE nochmals um 31% reduziert wird. Es zeigt sich also, dass durch die konsequente Ausnutzung der testmengenspezifischen Information erhebliche Einsparungen erzielt werden können. Die Schaltungen b18 und p286k sind schwer zufallstestbar und entsprechend ist auch die erreichte Einsparung gegenüber [11] und [10] gering.

IV. SCHLUSSBEMERKUNG

In vielen aktuellen Chip-Entwürfen können Prüfpfade während des Tests individuell deaktiviert werden. Das vorgestellte Verfahren erzeugt hierfür einen Testplan, der die Verlustleistung während des Tests optimiert. Durch die konsequente Nutzung testmengenspezifischer Information wird besonders bei großen, industriellen Schaltungen ein deutlich besseres Ergebnis als mit den bisher vorgestellten Verfahren erreicht.

V. DANKSAGUNG

Die vorliegende Arbeit wurde durch das IBM CAS Projekt „Improved Testing of VLSI Chips with Power Constraints“, sowie durch die Deutsche Forschungsgemeinschaft unter dem DFG Projekt „Realtest“ Wu245/5-1 unterstützt. Die Schaltungen von NXP wurden im Rahmen des Projekts „Realtest“ zur Verfügung gestellt.

Cell Broadband Engine ist ein eingetragenes Warenzeichen von Sony Computer Entertainment Inc.

REFERENCES

[1] Y. Zorian, “A distributed BIST control scheme for complex VLSI devices,” in *Proceedings of the 11th IEEE VLSI Test Symposium (VTS)*, 1993, pp. 4–9.

[2] C. F. Hawkins and J. Segura, “Test and reliability: Partners in IC manufacturing, part 1,” *IEEE Design & Test of Computers*, vol. 16, no. 3, pp. 64–71, 1999.

[3] C. F. Hawkins, J. Segura, J. M. Soden, and T. Dellin, “Test and reliability: Partners in IC manufacturing, part 2,” *IEEE Design & Test of Computers*, vol. 16, no. 4, pp. 66–73, 1999.

[4] Y. Huang, S. M. Reddy, W.-T. Cheng, P. Reuter, N. Mukherjee, C.-C. Tsai, O. Samman, and Y. Zaidan, “Optimal core wrapper width selection and SOC test scheduling based on 3-D bin packing algorithm,” in *Proceedings IEEE International Test Conference (ITC), Baltimore, MD, USA, October 7-10*, 2002, pp. 74–82.

[5] N. Nicolici and B. M. Al-Hashimi, “Power-conscious test synthesis and scheduling,” *IEEE Design & Test of Computers*, vol. 20, no. 4, pp. 48–55, 2003.

[6] P. Girard, “Survey of low-power testing of VLSI circuits,” *Design & Test of Computers, IEEE*, vol. 19, no. 3, pp. 80–90, 2002.

[7] S. Gerstendoerfer and H.-J. Wunderlich, “Minimized power consumption for scan-based BIST,” in *Proceedings IEEE International Test Conference (ITC), Atlantic City, NJ, USA, 27-30 September*, 1999, pp. 77–84.

[8] S. Manich, A. Gabarro, M. Lopez, and J. Figueras, “Low power BIST by filtering non-detecting vectors,” *Journal of Electronic Testing Theory and Applications (JETTA)*, vol. 16, no. 3, pp. 193–202, 2000.

[9] R. Sankaralingam, N. A. Touba, and B. Pouya, “Reducing power dissipation during test using scan chain disable,” in *19th IEEE VLSI Test Symposium (VTS 2001), 29 April - 3 May 2001, Marina Del Rey, CA, USA*, 2001, pp. 319–325.

[10] C. Zoellin, H.-J. Wunderlich, N. Maeding, and J. Leenstra, “BIST power reduction using scan-chain disable in the Cell processor,” in *IEEE International Test Conference (ITC), Santa Clara, CA, USA, October 24 - 26*, 2006.

[11] M. E. Imhof, C. G. Zoellin, H.-J. Wunderlich, N. Maeding, and J. Leenstra, “Scan test planning for power reduction,” in *DAC ’07: Proceedings of the 44th annual conference on Design automation*. New York, NY, USA: ACM Press, 2007, pp. 521–526.

[12] I. Hamzaoglu and J. H. Patel, “New techniques for deterministic test pattern generation,” in *16th IEEE VLSI Test Symposium (VTS), 28 April - 1 May, Princeton, NJ, USA*, 1998, pp. 446–452.

[13] B. Chess and T. Larrabee, “Creating small fault dictionaries,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 18, no. 3, pp. 346–356, 1999.

[14] C. Liu and K. Chakrabarty, "Design and analysis of compact dictionaries for diagnosis in scan-BIST," *IEEE Trans. VLSI Systems*, vol. 13, no. 8, pp. 979–984, 2005.

[15] P. H. Bardell and W. H. McAnney, "Self-testing of multichip logic modules," in *Proceedings International Test Conference (ITC), Philadelphia, PA, USA, November*, 1982, pp. 200–204.

[16] B. L. Keller and T. J. Snethen, "Built-in self-test support in the IBM engineering design system," *IBM Journal of Research and Development*, vol. 34, no. 2/3, pp. 406–415, 1990.

[17] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. S. M. Hassan, and J. Rajski, "Logic BIST for large industrial designs: real issues and case studies," in *Proceedings IEEE International Test Conference (ITC), Atlantic City, NJ, USA, 27-30 September*, 1999, pp. 358–367.

[18] H.-J. Wunderlich, "Multiple distributions for biased random test patterns," in *Proceedings International Test Conference, Washington, D.C., USA, September*, 1988, pp. 236–244.

[19] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa, "The design and implementation of a first-generation Cell processor," in *International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers, 6-10 Feb., San Francisco CA*, 2005, pp. 184–185.

[20] O. Coudert, "On solving covering problems," in *Proceedings of the 33rd Conference on Design Automation, Las Vegas, Nevada, USA, June 3-7*, 1996, pp. 197–202.

[21] M. Riley, L. Bushard, N. Chelstrom, N. Kiryu, and S. Ferguson, "Testability features of the first-generation Cell processor," in *Proceedings of the IEEE International Test Conference (ITC), 8-10 Nov., Austin TX*, 2005, p. 6.1.