it 2/2009



# Effiziente Erfüllbarkeitsalgorithmen für die Generierung von Testmustern

## Efficient Satisfiability Solving Algorithms for Test Pattern Generation

Rolf Drechsler, Stephan Eggersglüß, Görschwin Fey, Universität Bremen,
Jürgen Schlöffel, Mentor Graphics Development (Deutschland) GmbH,
Daniel Tille, Universität Bremen

**Zusammenfassung** Während Schaltungen noch bis vor einigen Jahren vornehmlich in Rechenanlagen Verwendung fanden, werden sie zunehmend ein Teil unserer täglichen Umgebung und finden in vielen sicherheitskritischen Bereichen Anwendung. Exemplarisch seien hier medizinische Apparaturen und die Automobilindustrie genannt. Schon heute entstehen z. B. in dem zuletzt genannten Bereich bis zu 40% der Kosten durch die Elektronik. Hinzu kommt, dass hier immer mehr sicherheitsrelevante Komponenten, wie Bremsen und Lenkung, elektronisch unterstützt bzw. gesteuert werden. Daher werden immer höhere Anforderungen an die Verfahren gestellt, die die Korrektheit dieser Schaltungen und Systeme gewährleisten sollen. In den vergangenen Jahren wurden große Fortschritte erzielt, doch durch die immer höhere Komplexität der Systeme und den Einsatz neuer Technologien müssen auch die Werkzeuge weiterentwickelt werden, die die Qualität sicher stellen. Ein wesentlicher Aspekt bei der Erstellung von Schaltungen und Systemen ist der Test. Die durch den Test entstehenden Kosten betragen bis zu 30% der Gesamtkosten der Fertigung. Eine grundlegende Methode zum Testen ist die Generierung von Testmustern ausgehend von einem System und einem betrachteten Fehlermodell. In den frühen 90er Jahren wurden Verfahren zur automatischen Testmustergenerierung (ATPG = Automatic Test Pattern Generation) basierend auf Boolescher Erfüllbarkeit (SAT = Satisfiability) vorgeschlagen. Aufgrund der Mächtigkeit moderner SAT-Beweiser sind diese Verfahren wieder in den Fokus der Forschung gerückt. In dieser Arbeit werden die grundlegenden Mechanismen der Methode vorgestellt und aktuelle Erweiterungen diskutiert. Es zeigt sich, dass es möglich ist, große industrielle Schaltungen mit diesen Verfahren effizient zu bearbeiten. Abschließend werden aktuelle Forschungsaufgaben identifiziert. ►►► **Summary** Due to the significant advances in the development of integrated circuits, electronic devices are applied in many fields of everyday life – increasingly in safety critical applications. Exemplarily, they are widely used in medical equipment and in the automotive industry. Today, approximately 40% of the overall costs for a car are caused by electronics. Additionally, more and more safety critical components as brakes and steering are based on electronic devices. Therefore, the requirements on the procedures that guarantee the correctness of these circuits and systems are high. As technology advances, there is a need for new tools that can cope with the increased complexity. The post production test is a crucial part of the design flow. Up to 30% of the overall manufacturing costs are caused by the test. The basic method of testing is the generation of test patterns based on a circuit and a fault model. In the early 90s, Automatic Test Pattern Generation (ATPG) procedures based on Boolean Satisfiability (SAT) were proposed. Due to the increased efficiency of modern SAT solvers, the scientific interest on these procedures has grown recently. In this paper, the basics of this method are reviewed and current improvements are discussed. Experiments show that the SAT-based method can cope efficiently even with large industrial circuits. Finally topics for future work are presented.

**KEYWORDS** B.7.3 [Hardware: Integrated Circuits: Reliability and Testing] Test generation; ATPG, satisfiability, SAT, SAT solver / Testmustergenerierung, Erfüllbarkeitsproblem, SAT-Beweiser

## 1 Einleitung

Die Komplexität integrierter Schaltungen hat seit den 70er Jahren stetig zugenommen, da sich – gemäß Moore's Gesetz – die Anzahl der Komponenten ca. alle 18 Monate verdoppelte. Aktuelle technologische Fortschritte lassen erwarten, dass dieses Wachstum mindestens noch weitere zehn Jahre anhält. Daher ist es notwendig, auch die Entwurfswerkzeuge, die den Entwickler unterstützen, kontinuierlich weiter zu verbessern.

Erschwerend kommt hinzu, dass Schaltungen immer häufiger in sicherheitskritischen Systemen verwendet werden. Exemplarisch seien hier medizinische Apparaturen und die Automobilindustrie genannt. Schon jetzt entfallen ca. 40% der Kosten bei der Automobilproduktion auf die Elektronik, wobei der Bereich Schaltkreis- und Systementwurf etwa die Hälfte davon ausmacht. Diese Problematik ist auch von Seiten der Hersteller erkannt worden, wie durch Herrn Prof. Jürgen Hubbert (Vorstand, DaimlerChrysler AG) im Vorfeld der *Internationalen Automobilausstellung (IAA)* schon im September 2003 in Frankfurt festgestellt wurde: „Wir müssen alle intensiv daran arbeiten, die Elektronik insgesamt besser beherrschen zu lernen. Sie bietet uns in Zukunft große Chancen, und deshalb sehe ich keinen Weg zurück."

Die Probleme werden zunehmend „verschärft", da immer mehr Komponenten, die für das korrekte Verhalten des Fahrzeuges verantwortlich sind, wie z. B. die Steuerung und die Bremse, durch elektronische Komponenten ersetzt oder zumindest unterstützt werden. Im Automobilsektor bezeichnet man diesen Einsatz auch häufig als „X-by-wire", um zu unterstreichen, dass im Gegensatz zur mechanischen Realisierung hier „Drähte" zum Einsatz kommen. Diese Neuerungen sind vor dem Hintergrund steigender Komplexität, hoher Qualitätsstandards, sinkender Entwicklungszeiten und steigenden Kostendrucks zu sehen.

Daher kommt dem Bereich des Testens eine immer höhere Bedeutung zu. Es muss sicher gestellt werden, dass Schaltungen nach der Fertigung nicht durch Mängel im Produktionsprozess fehlerhaft sind und so an den Kunden weiter gereicht werden.

Statt nun bezüglich aller möglichen Fertigungsfehler zu testen, wird ein Fehlermodell auf Logikebene verwendet. Bezüglich dieses Fehlermodells werden dann durch *Automatische Testmustergenerierung* (engl.: *Automatic Test Pattern Generation*, ATPG) Eingabestimuli berechnet, die zur Fehlerentdeckung dienen. Zahlreiche klassische Algorithmen [16; 17; 22] stehen hierfür zur Verfügung, erreichen jedoch bei der Anwendung auf industrielle Schaltungen ihre Leistungsgrenzen.

Algorithmen zur Lösung des *Booleschen Erfüllbarkeitsproblems* (engl.: *Boolean Satisfiability*, SAT) wurden in der jüngeren Vergangenheit deutlich verbessert [20; 21]. Dadurch erfuhren frühe Versuche der SAT-basierten Testmustergenerierung [19; 24] eine Renaissance und können, wie dieser Beitrag zeigen wird, erfolgreich zur Ergänzung klassischer ATPG-Algorithmen verwendet werden.

Die wesentlichen Probleme bei den bisher vorgeschlagenen, auf SAT basierenden ATPG-Algorithmen lassen sich wie folgt zusammenfassen:

- Eine praxisnahe *mehrwertige Modellierung* wurde nicht eingesetzt. Um die Instanz einfach zu halten wurden bisher nur die Booleschen Werte 0 und 1 unterstützt. Dies ist aber für den praktischen Einsatz nicht ausreichend, da heutige Schaltungen auch über Tri-State Elemente verfügen, z. B. zur Beschreibung von Bussen oder von Restriktionen durch die Umgebung.
- Für die realistische mehrwertige Modellierung wurden bisher keine *Optimierungen bezüglich einzelner Fehlermodelle* untersucht, z. B. für sequentielle Probleme, wie sie bei Pfadverzögerungsfehlern auftreten.
- Die in den vergangenen Jahren gewonnenen Erkenntnisse, die zur Entwicklung *moderner SAT-Beweiser* führten, wurden nicht auf den Testbereich übertragen.
- Für moderne SAT-Beweiser fehlten bisher für das Testen *problemspezifische Heuristiken*.
- Aufgrund der Einschränkung auf Boolesche Werte wurde SAT-ATPG bisher nicht auf *große industrielle Schaltungen* mit mehreren Millionen Gattern angewandt.

Im Folgenden wird gezeigt, wie SAT-Techniken im Bereich der Testmustergenerierung eingesetzt werden können. Es werden sowohl SAT-Verfahren als auch testspezifische Methoden diskutiert und im praktischen Umfeld evaluiert. Zu den einzelnen Anwendungen werden experimentelle Untersuchungen – zum Teil für industrielle Schaltungen – präsentiert. Abschließend werden die bisherigen Resultate zusammengefasst und weitere mögliche Schritte diskutiert.

Abschnitt 2 stellt die Grundlagen zu ATPG und SAT im Überblick dar. In Abschnitt 3 wird die Transformation der Testmustergenerierung auf SAT erläutert. Dann wird das Werkzeug PASSAT in Abschnitt 4 vorgestellt. Wichtige Optimierungen werden in Abschnitt 5 erläutert. Bis hierher werden nur statische Fehler betrachtet, eine Erweiterung auf dynamische Fehler bietet Abschnitt 6. Die Integration des SAT-Ansatzes in eine existierende Umgebung diskutiert Abschnitt 7. Schließlich werden neben einer Zusammenfassung weiterführende Fragen in Abschnitt 8 aufgezeigt.

## 2 Grundlagen

### 2.1 Fehlermodelle

Es gibt viele verschiedene Fehler, die während der Schaltkreisproduktion auftreten können. Als Beispiel seien Verunreinigungen oder Fehler im

lithographischen Fertigungsprozess genannt. Da es nicht praktikabel ist, all diese Fehler zu testen, abstrahiert man von den möglichen physikalischen Fehlern auf ein mathematisches Fehlermodell. Hierbei unterscheidet man zwischen statischen und dynamischen Fehlermodellen.

Statische Fehlermodelle modellieren eine fehlerhafte Funktion der Schaltung. Beispielsweise besitzt beim *Haftfehlermodell* eine Leitung fehlerhafterweise dauerhaft den logischen Wert 0 oder 1, und ist somit unabhängig von den primären Eingängen.

Bei dynamischen Fehlermodellen besitzen die Schaltungen hingegen eine korrekte Funktionalität; das Zeitverhalten wird als fehlerhaft modelliert. Bei einer Transition, d. h. dem Schalten von logisch 0 auf logisch 1, oder umgekehrt, ist eine Verzögerung (engl.: *delay*) innerhalb eines Gatters (*Transitionsfehlermodell*) oder entlang eines Pfades (*Pfadverzögerungsfehlermodell*) ungewöhnlich hoch, sodass nachfolgende Schaltungsteile mit „veralteten" Werten rechnen.

Im Folgenden beschränken wir uns in unseren Ausführungen zunächst auf das Haftfehlermodell, um die Darstellung einfach zu halten. Eine Erweiterung auf dynamische Fehlermodelle wird in Abschnitt 6 gegeben.

### 2.2 Testmustergenerierung

Die Testmustergenerierung hat das Ziel, eine Belegung der primären Eingänge zu finden, die es ermöglicht, einen bestimmten Fehler bezüglich einer Schaltung und eines Fehlermodells zu erkennen. Das Entscheidungsproblem, ob ein solches Testmuster existiert, ist NP-vollständig.

Um es zu lösen, existieren verschiedene Verfahren. Der überwiegende Teil basiert auf dem D-Algorithmus [22]. Dieser nutzt die 4-wertige Logik $\{0, 1, D, \overline{D}\}$ und arbeitet nach folgendem Prinzip:

An der Fehlerstelle wird eine Differenz zwischen der korrekten und der fehlerhaften Schaltung injiziert. Dies wird beim Haftfehler 0 durch den Wert $D$ und beim Haftfehler 1 durch den Wert $\overline{D}$ modelliert. Um den Fehler zu testen, muss der Wert 1 (Wert 0) für den Haftfehler 0 (Haftfehler 1) an der Fehlerstelle eingestellt werden. Zusätzlich muss die Differenz zu einem der primären Ausgänge propagiert werden, d. h. es muss einen Pfad von der Fehlerstelle zu einem Ausgang geben, auf dem jedes Signal den Wert $D$ oder $\overline{D}$ besitzt. Diesen Pfad nennt man *D-Pfad*.

Es ist notwendig, alle Seiteneingänge, d. h. alle Eingänge eines Gatters, die nicht auf einem D-Pfad liegen, mit dem *nicht kontrollierenden* Wert zu belegen. Beim OR und beim NOR ist das der Boolesche Wert 0, beim AND und NAND der Wert 1.

Gelingt es nicht, eine Differenz zu einem der Ausgänge zu propagieren, so ist der Fehler untestbar.

### 2.3 Boolesche Erfüllbarkeit

Das Boolesche Erfüllbarkeitsproblem ist wie folgt definiert:

**Definition 1.** *Sei $f : \{0,1\}^n \to \{0,1\}$ eine Boolesche Funktion. Das Erfüllbarkeitsproblem ist dann die Frage, ob eine Variablenbelegung $a$ existiert, so dass $f(a) = 1$ gilt. Falls solch ein $a$ existiert, so nennt man $f$ erfüllbar (engl.: satisfiable), anderenfalls unerfüllbar (engl.: unsatisfiable).*

Die Boolesche Funktion liegt üblicherweise in *konjunktiver Normalform* (KNF) vor. Genau wie die Testmustergenerierung ist das Erfüllbarkeitsproblem NP-vollständig [4].

Algorithmen, die das SAT-Problem lösen (sogenannte SAT-Beweiser), basieren meist auf dem DPLL-Algorithmus [6]. Dabei wird der Suchraum mit einem Backtracking Algorithmus durchquert. Die Effizienz heutiger SAT-Beweiser ist verschiedenen Erweiterungen zu verdanken. Auf die drei wichtigsten wird im Folgenden kurz eingegangen.

Während der *Konfliktanalyse* wird die Ursache, warum die aktuelle Belegung zu logisch 0 auswertet, genauer untersucht. Dabei findet man eine bestimmte Kombination von Variablenbelegungen, die für diesen Konflikt verantwortlich ist. Ein Ausschließen dieser Belegung erlaubt es, große Teile des Suchraumes, in denen sich keine Lösungen befinden können, frühzeitig zu erkennen [20].

Eine effektive *Boolean Constraint Propagation* (BCP) trägt ebenfalls zum Erfolg neuer SAT-Beweiser bei. Bei BCP werden Implikationen ausgeführt, d. h. Variablenbelegungen durchgeführt, die aus der aktuellen (partiellen) Belegung gefolgert werden können. Da dieser Prozess einen Großteil der Laufzeit in Anspruch nimmt, sind effektive Datenstrukturen hier essenziell [21].

Falls keine Variablenbelegung impliziert werden kann, so wird eine „beliebige" Variable zunächst mit einem Booleschen Wert belegt. Findet man unter dieser Belegung keine Lösung, wird der Wert komplementiert. Die Auswahl, welche Variable belegt wird, ist Aufgabe einer *Entscheidungsheuristik*. Erste Heuristiken basierten auf statistischen Eigenschaften, wie Häufigkeit des Auftretens einer Variablen innerhalb der KNF. Bei aktuellen Strategien werden Variablen ausgewählt, die häufig in Konflikten vorkommen [18].

## 3 SAT-basierte Testmustergenerierung

Es ist sehr einfach, einen Schaltkreis in eine KNF zu überführen [30]. Zunächst wird jedem Gatter, d. h. dem Ausgang jedes Gatters, eine Boolesche Variable zugeordnet. Anschließend erstellt man die charakteristischen Funktion aller Gatter. Die Konjunktion sämtlicher charakteristischen Funktionen bildet die KNF des Schaltkreises.

Eine naheliegende Transformation der Testmustergenerierung auf das Erfüllbarkeitsproblem ist die Boolesche Differenz. Hier werden sowohl der korrekte als auch der

fehlerhafte Schaltkreis in KNF transformiert. Sucht man nun nach einer Variablenbelegung, bei der sich die primären Ausgänge beider Schaltungen unterscheiden, so hat man das ATPG-Problem auf das SAT-Problem reduziert. Diesen Schaltkreis nennt man *Miter* [2].

Schon zu Beginn der 80er Jahre wurden erste Techniken, basierend auf Boolescher Erfüllbarkeit, vorgeschlagen und seitdem weiter entwickelt (siehe z. B. [20; 24; 25]). In [24] wurde ein Vergleich mit mehr als 40 „klassischen" Verfahren, die auf dem D-Algorithmus [22] oder dessen Erweiterungen PODEM [17] und FAN [16] beruhen, angegeben. Es zeigte sich, dass SAT-basierte Verfahren deutlich robuster sind und auch in vielen Fällen schneller zu Lösungen führen. Im Gegensatz zu den ersten SAT-basierten Verfahren der frühen 80er Jahre, in denen SAT auf PROLOG oder logische Programmierung abgebildet wurde, unterstützen die modernen Methoden auch verschiedene aus dem Bereich des Testens bekannte Optimierungen, wie z. B. Lernen und globale Implikationen (siehe z. B. [19; 24; 25]). Dadurch wurde es möglich, die Leistungsfähigkeit von SAT mit Algorithmen aus dem Testbereich zu kombinieren.

In den vergangenen zehn Jahren gab es zur algorithmischen Lösung von SAT signifikante Verbesserungen. Es wurden neue Implementierungskonzepte vorgeschlagen, die es ermöglichen, sehr große Instanzen zu bearbeiten. Die wichtigsten wurden im letzten Abschnitt erwähnt.

Diese neuen Methoden haben zu bedeutenden Verbesserungen in zahlreichen Anwendungsgebieten geführt, wie z. B. in der formalen Hardware-Verifikation [7]. Im Bereich des ATPG wurden diese neuen Methoden allerdings bisher kaum eingesetzt.

Bevor die neuesten Konzepte genauer diskutiert werden, soll die prinzipielle Methodik, wie sie im ATPG-Algorithmus TEGUS [24] eingeführt wurde, nochmals kurz

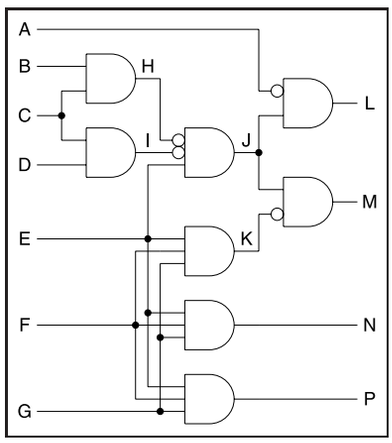


**Bild 1** Beispiel aus [24].

skizziert werden: Das Verfahren basiert auf dem D-Algorithmus [22], dessen Grundidee im vorherigen Abschnitt zusammengefasst wurde. Im Wesentlichen müssen die *D*-Werte – vom Fehlerort ausgehend – zu den Ausgängen propagiert und die seitlichen Signale entlang des Pfades sowie der Fehlerort entsprechend eingestellt werden.

Bei TEGUS wird eine gegebene Schaltkreisbeschreibung in eine KNF überführt, die dann dem SAT-Beweiser übergeben wird. Bei der Erzeugung der KNF werden die Beobachtungen vom D-Algorithmus berücksichtigt. Hierbei werden für ein Gatter $G$ drei Boolesche Variablen verwendet:

- $G_g$ beschreibt den Wert im korrekten Schaltkreis,
- $G_f$ beschreibt den Wert im fehlerhaften Schaltkreis,
- $G_d$ gibt an, ob $G$ auf einem D-Pfad liegt.

Um nun einen Fehler zu modellieren, wird die Variable $G_f$ auf den entsprechenden festen Wert gesetzt. Hierbei ist $G$ das Vorgängergatter der fehlerhaften Leitung.

Zusätzliche Implikationen erlauben eine schnellere Bearbeitung der KNF: Ist ein Gatter $G$ auf einem *D*-Pfad, so sind die Werte im fehlerhaften und korrekten Schaltkreis unterschiedlich. Es gilt

$$G_d \rightarrow (G_f \neq G_g)\,.$$

Ist ein Gatter $G$ auf einem *D*-Pfad, so muss auch mindestens ein Nachfolger von $G$ auf dem Pfad sein. Es gilt also:

$$G_d \rightarrow \bigvee_{i=1}^{n} H_d^i$$

wobei $H^i, 1 \leq i \leq n$, die Nachfolger von $G$ sind.

Ist ein Gatter $G$ nicht im Fanout-Kegel des Fehlers, so sind die Werte im fehlerhaften und korrekten Schaltkreis identisch, d. h.

$$G_f = G_g\,.$$

Die erzeugte SAT-Instanz, d. h. die KNF für den korrekten und den fehlerhaften Schaltkreis, zusammen mit den eben genannten Implikationen, wird an den SAT-Beweiser übergeben. Die SAT-Instanz ist genau dann erfüllbar, wenn es einen *D*-Pfad gibt, also wenn der Fehler testbar ist. In diesem Fall liefert die Variablenbelegung der primären Eingänge unmittelbar ein Testmuster.

Die eingefügten Bedingungen erfordern die Weiterleitung eines *D*-Wertes an mindestens einen Nachfolger eines Gatters. Dadurch ist es unnötig, eine Miter-Schaltung zu erzeugen. Außerdem kann die

| Implikationen | KNF-Beschreibung |
|---|---|
| $J_g \equiv (\overline{H}_g \cdot \overline{I}_g \cdot E_g)$ | $(J_g + H_g + I_g + \overline{E}_g) \cdot (\overline{H}_g + \overline{J}_g) \cdot (\overline{I}_g + \overline{J}_g) \cdot (\overline{E}_g + \overline{J}_g)$ |
| $J_f \equiv (\overline{H}_f \cdot \overline{I}_f \cdot E_f)$ | $(J_f + H_f + I_f + \overline{E}_f) \cdot (\overline{H}_f + \overline{J}_f) \cdot (\overline{I}_f + \overline{J}_f) \cdot (\overline{E}_f + \overline{J}_f)$ |
| $J_d \rightarrow (J_g \neq J_f)$ | $(\overline{J}_d + J_g + J_f) \cdot (\overline{J}_d + \overline{J}_g + \overline{J}_f)$ |
| $J_d \rightarrow (L_d + M_d)$ | $(\overline{J}_d + L_d + M_d)$ |

**Bild 2** Repräsentation eines Gatters in KNF.

SAT-Instanz durch die zusätzlichen Implikationen des D-Algorithmus oft sehr schnell klassifiziert werden. Insbesondere Konflikte bei untestbaren Fehlern führen oft unmittelbar zur Unerfüllbarkeit.

Die Methode soll an einem kleinen Beispiel aus [24] illustriert werden.

**Beispiel 1.** *Man betrachte die Teilschaltung in Bild 1. Wendet man das obige Verfahren auf das Gatter J an, so erhält man die Booleschen Implikationen in Bild 2. Diese können jeweils direkt in eine KNF übersetzt werden, die in der zweiten Spalte angegeben ist.*

## 4 PASSAT

In einem ersten Schritt wurde das prototypische ATPG-Werkzeug PASSAT in Zusammenarbeit mit NXP Semiconductors Germany GmbH (Hamburg) und mit Mentor Graphics Development GmbH (Hamburg) implementiert, um die Erfolgsaussichten der Techniken zu bewerten. PASSAT basiert auf dem Werkzeug TEGUS aus [24]. Die wesentlichen Erweiterungen sind:

- *Moderne SAT-Beweiser*
  Es wurden moderne SAT-Beweiser für den Beweisprozess verwendet. In einer ersten Version wurde der SAT-Beweiser Zchaff [21] eingebaut. Dieser wurde in der Zwischenzeit durch den neueren Beweiser MiniSat [9] ersetzt.
- *Mehrwertige Modellierung*
  Während TEGUS nur die Modellierung mit Booleschen Werten unterstützt, kann PASSAT auch mehrwertige Signale verarbeiten. Neben den Booleschen Werten 0 und 1 sind auch *U* und *Z* zulässig. Hierbei steht *U* für einen unbekannten, nicht beeinflussbaren Wert, wie z. B. Restriktionen an primären Eingängen. Der Wert *Z* symbolisiert hingegen einen hochohmigen Zustand, der benötigt wird, um Busse korrekt zu modellieren. Die Wahl der Kodierung ist dabei von großer Bedeutung für die Effizienz des Verfahrens. Details zu dieser Problematik finden sich in [14].

Eine detaillierte Beschreibung der Arbeitsweise von PASSAT findet man in [8].

## 5 SAT-Optimierung

Im Folgenden werden verschiedene Optimierungstechniken kurz vorgestellt, die über den einfachen Einsatz moderner SAT Beweiser hinausgehen. Die Effizienz der einzelnen Techniken wurde sowohl anhand von frei verfügbaren Benchmarks als auch anhand industrieller Schaltungen nachgewiesen.

In Tabelle 1 sind einige Eigenschaften der Schaltkreise, auf denen die Techniken erprobt wurden, zusammengefasst. In der ersten Spalte ist der Schaltkreisname angegeben. Die Schaltungen, deren Namen mit ‚b' beginnen, sind Teil des ITC'99 Benchmark Pakets [5]. Die übrigen Schaltkreise sind industrielle Schaltungen, zur Verfügung gestellt von NXP Semiconductors Germany GmbH, Hamburg. Hier gibt der Name Aufschluss über die ungefähre Anzahl an Gattern. Die Anzahl der primären Ein- und Ausgänge und der Flip-Flops sind in den Spalten zwei bis vier gegeben. Spalte fünf gibt den prozentualen Anteil des Schaltkreises an, der vierwertig behandelt werden muss (vgl. *Hybride Logik*).

### 5.1 Konfliktbasiertes Lernen

ATPG-Methoden durch Lernen zu unterstützen ist eine gebräuchliche Methode. Gelernte Informationen werden vom SAT-Beweiser in Form von Konfliktklauseln berücksichtigt. Kommen nun ähnliche SAT-Instanzen $\Phi_1$ und $\Phi_2$ vor, so ist es möglich, Konfliktklauseln, die beim Lösen von $\Phi_1$ aus der Schnittmenge $\Phi_1 \cap \Phi_2$ gelernt wurden, bei der Lösung von $\Phi_2$ wiederzuverwenden [31]. Dies kann eine substanzielle Reduktion des Suchraumes bewirken. In der formalen Verifikation führt dies zu erheblicher Leistungssteigerung [23].

Auch bei der Testmustergenerierung liegt eine ähnliche Situation vor, da auf dem immer gleichen Schaltkreis eine große Anzahl verschiedener Fehler modelliert wird. Große Bereiche der Probleminstanz bleiben dabei unverändert. Weiterhin ist die Reihenfolge, in der einzelne Fehler betrachtet werden, nicht vorgegeben.

Hierbei gilt es, einen guten Kompromiss zwischen der Menge der zu lernenden Klauseln und dem Wiederverwendungsgrad zu finden. Lernt man zu intensiv, benötigt der Lernprozess viel Zeit, wohingegen die Informationen nur selten eingesetzt werden können. Wird nicht gelernt, profitiert man nicht und muss sehr ähnliche Berechnungen gegebenenfalls mehrfach durchführen.

**Tabelle 1** Schaltkreisinformationen.

| Schaltkreis | Eingänge | Ausgänge | FF | %4V |
|---|---|---|---|---|
| b14 | 32 | 55 | 245 | 0 |
| b15 | 36 | 71 | 449 | 0 |
| b17 | 37 | 98 | 1415 | 0 |
| b18 | 37 | 23 | 3320 | 0 |
| b20 | 32 | 23 | 490 | 0 |
| b21 | 32 | 23 | 490 | 0 |
| b22 | 32 | 23 | 735 | 0 |
| p44k | 739 | 56 | 2175 | 0 |
| p49k | 303 | 71 | 334 | 0 |
| p80k | 152 | 75 | 3878 | 0 |
| p88k | 331 | 183 | 4309 | 8,65 |
| p99k | 167 | 82 | 5747 | 1,73 |
| p177k | 768 | 1 | 10 507 | 33,81 |
| p462k | 1815 | 1193 | 29 205 | 17,49 |
| p565k | 964 | 169 | 32 409 | 21,22 |
| p1330k | 584 | 90 | 104 630 | 11,54 |

Studien in [15] zeigen, dass bei geschickter Wahl der Lernparameter eine Verbesserung um einen Faktor 2 erreicht werden kann.

### 5.2 Generierung der Instanz

Wie bereits erwähnt, wird für jeden Fehler eine SAT-Instanz aufgebaut, die im Anschluss vom SAT-Beweiser gelöst wird. Da eine kleine KNF in der Regel leichter zu lösen ist als eine große, liegt bereits bei der Instanzgenerierung Optimierungspotenzial.

#### *5.2.1 Multi-input Gatter*

Eine erste Optimierung ist die verbesserte Gatter-Repräsentation als KNF. Hat ein Gatter mehr als zwei Eingänge, so ist dessen Repräsentation nicht eindeutig: Neben der Darstellung als einzelnes Gatter (*multi-input Gatter*) ist auch eine Kaskade von 2-input Gattern möglich.

Bei der Verwendung von Boolescher Logik ist die Darstellung als einzelnes Gatter vorzuziehen. Bei 4-wertiger Logik hingegen steigt bei dieser Repräsentation die KNF-Größe – d. h. die Anzahl der benötigten Klauseln – exponentiell an. Eine Kaskade von 2-input Gattern ist somit hinsichtlich der Klauselanzahl hier zwar vorzuziehen, jedoch werden damit viele unnötige KNF-Variablen erzeugt.

Eine „Mischung" aus beiden Ansätzen zur Darstellung ist folglich am vielversprechendsten: Zwar werden multi-input Gatter verwendet, die Anzahl der Eingänge ist jedoch auf fünf beschränkt. Bei Gattern mit mehr als fünf Eingängen geschieht eine Aufteilung in mehrere Gatter. Diese Größe wurde durch Experimente bestimmt. Sie bietet den besten Kompromiss zwischen der Variablenanzahl und Klauselanzahl. Details hierzu findet man in [28].

#### *5.2.2 Hybride Logik*

Wie in Abschnitt 4 angegeben, wird die Modellierung vierwertiger Logik unterstützt. Hierbei wird jedes Signal durch zwei Boolesche Variablen kodiert.

Da in den meisten industriellen Schaltkreisen die Anzahl der Signale, die vierwertig behandelt werden müssen, jedoch gering ist (vgl. Tabelle 1), bedeutet diese Art der Modellierung einen enormen Mehraufwand. Durch einen Ansatz, bei dem nur die notwendigen Teile im Schaltkreis vierwertig behandelt werden, während der Rest rein Boolesch behandelt wird, kann die Größe der SAT-Instanz signifikant reduziert werden.

Ein Überblick der Effizienz der Methode bezüglich der Laufzeit ist in Tabelle 3 gegeben. Spalte *SAT_4V* beinhaltet die Anzahl der Abbrüche (*Ab.*) und die Laufzeit (*Zeit*) für die rein vierwertige Kodierung. Spalte *SAT* gibt hingegen die Werte für die neu vorgestellte Methode an. Wie man sieht, werden sowohl die benötigte Laufzeit als auch die Anzahl der Abbrüche zum Teil drastisch reduziert (vgl. Schaltkreis p177k).

Die Methode ist in [8] detailliert beschrieben.

#### *5.2.3 BDD-basierter Instanzaufbau*

Eine weitere Möglichkeit, SAT-Instanzen beim Aufbau zu optimieren, wurde in [29] vorgeschlagen. Dabei wird anfangs der zugrunde liegende Schaltkreis hinsichtlich verzweigungsfreier Regionen (*Fanout-Free Region*, FFR) partitioniert. Für jede FFR wird anschließend ein binäres Entscheidungsdiagramm (*Binary Decision Diagram*, BDD) [3] aufgebaut.

Ein BDD stellt eine kanonische Datenstruktur einer Booleschen Funktion dar. Daher sind die zum Teil zahlreichen Redundanzen innerhalb einer FFR, die in industriellen Schaltkreisen vorkommen, wenn sich mehrere FFR-Eingänge ein Vorgängergatter teilen, nicht im BDD enthalten.

Die Generierung der SAT-Instanz erfolgt weiterhin über die Schaltungsstruktur – jedoch ist das Vorgehen nicht mehr Gatter-weise sondern FFR-weise. Zunächst wird die FFR, die die fehlerhafte Leitung enthält, traditionell aufgebaut. Die restlichen FFRs werden basierend auf den BDDs erzeugt. Somit sind die KNFs ebenfalls frei von den im Schaltkreis enthaltenen Redundanzen, wodurch die Instanzen deutlich kompakter werden.

Die Methode ist in [29] ausführlich beschrieben.

#### *5.2.4 Inkrementeller Instanzaufbau*

Vergleicht man die Komplexitäten von Instanzgenerierung (linear in der Anzahl der Gatter im Schaltkreis) mit dem Lösen der Instanz (NP-vollständig), so sollte die für den Aufbau der SAT-Instanz benötigte Zeit gegenüber der Zeit für das Lösen vernachlässigbar sein.

Experimente haben jedoch gezeigt, dass oft das Gegenteil der Fall ist [26]. Während das Lösen nahezu keine Zeit benötigt, ist der Aufbau sehr langwierig. Um den gesamten ATPG-Prozess zu beschleunigen, ist es daher ungenügend, lediglich Verbesserungen bezüglich des Lösungsprozesses zu betrachten; auch die Instanzgenerierung muss optimiert werden.

Hierzu wurde ein inkrementeller Aufbau der SAT-Instanz vorgeschlagen. Indem man anfangs nur einen Teil des Schaltkreises in KNF überführt, werden sowohl Aufbau- als auch Lösungszeit verkürzt. Im herkömmlichen Ansatz werden alle Gatter im transitiven Fan-in aller vom Fehlerort strukturell beeinflussten primären Ausgänge zur KNF Erzeugung genutzt. Beim inkrementellen Ansatz besteht die initiale SAT-Instanz hingegen lediglich aus Gattern, die im Fan-in Kegel eines Ausgangs liegen.

Diese partielle SAT-Instanz wird dem SAT-Beweiser übergeben. Ist sie nicht ausreichend, um den Fehler zu klassifizieren, werden Gatter von Fan-in Kegeln weiterer Ausgänge hinzugefügt. Anschließend wird die erweiterte KNF erneut vom SAT-Beweiser gelöst. Diese Schritte werden fortgeführt, bis eine Klassifikation möglich ist – die Instanz also vollständig aufgebaut oder ein Testmuster gefunden wurde.

Ausführliche Informationen zu dieser Technik sind in [26] dargestellt.

### 5.3 Kompakte Testmuster

Ein Nachteil aktueller SAT-Beweiser ist, dass immer eine vollständige Belegung der Variablen berechnet wird. Da aus dieser Variablenbelegung ein Testmuster abgeleitet wird, hat das zur Folge, dass die resultierenden Testmuster überspezifiziert sind, d. h. es gibt keine *Don't Cares*.

Um die Zeit für das Testen jedes einzelnen Chips so kurz wie möglich zu halten, muss die Anzahl der Testmuster jedoch so gering wie möglich sein. Daher werden die berechneten Testmuster mittels statischer Kompaktierung zusammengefasst. Hierfür ist es essenziell, so viele Don't Cares wie möglich zu haben. Durch die oben genannte Problematik können die durch SAT-ATPG generierten Testmuster jedoch nur ungenügend kompaktiert werden.

In [10] wird eine Technik vorgestellt, die diesen Mangel weitestgehend behebt. Mittels eines Postprozesses werden notwendige und hinreichende Variablenbelegungen berechnet. Unnötige Belegungen werden rückgängig gemacht und durch Don't Cares ersetzt.

## 6 Dynamische Fehlermodelle

Je nach Fehlermodell ergeben sich unterschiedliche Nebenbedingungen, die in Form von Klauseln darzustellen sind. In bisherigen Arbeiten zur SAT-basierten Testmustergenerierung wurde in erster Linie das statische Haftfehlermodell berücksichtigt. Für weitere statische Fehlermodelle ist zu erwarten, dass die Transformation auf SAT sehr ähnlich zur bisherigen für Haftfehler ist. In direkter Weise ist auch eine Erweiterung auf Mehrfachfehler umsetzbar. Während hierbei klassische ATPG-Methoden signifikant verändert werden müssen, entspricht dies bei SAT-basierten Verfahren nur dem Konstantsetzen mehrerer Werte in der KNF.

Aber auch die Modellierung von dynamischen Fehlern ist einfach möglich. Dynamische Fehlermodelle werden in der Praxis durch geringere Strukturgrößen immer wichtiger. Für die Testmustergenerierung muss hier allerdings ein sequentielles Problem betrachtet werden. Um einen zeitabhängigen Fehler zu entdecken, müssen zwei Testvektoren in aufeinanderfolgenden Takten auf den Schaltkreis angewendet werden. Deshalb müssen die zeitlichen Abhängigkeiten im Schaltkreis bei der Testmustergenerierung z. B. beim Pfadverzögerungs- oder Transitionsfehlermodell berücksichtigt werden. Das gleiche Problem stellt sich in der formalen Verifikation beim Eigenschaftsbeweis. Dort wird die Schaltung „abgerollt", d. h. die Schaltung wird innerhalb einer Probleminstanz zu verschiedenen Zeitpunkten modelliert [1]. Dazu werden in der Instanz mehrere Kopien der Schaltung dargestellt und die Zustandsübergangslogik zwischen den Kopien wird entsprechend des zeitlichen Ablaufs verbunden. Eine ähnliche Modellierung wurde in einem ersten Schritt für dynamische Fehlermodelle verwendet.

In [12] wird gezeigt, dass eine korrekte Modellierung von Pfadverzögerungsfehlern möglich ist. Auch wird dies mit dem oben beschriebenen SAT-Verfahren für das Lernen von Informationen kombiniert [11]. Obgleich das Pfadverzögerungsfehlermodell akkurater ist, wird es vorwiegend zu diagnostischen Zwecken eingesetzt, da die Anzahl der Pfade in modernen Schaltkreisen zu hoch ist. Das Transitionsfehlermodell ist weniger akkurat; es lässt sich aber eine höhere Fehlerüberdeckung realisieren. Daher wird dieses Modell vorwiegend in der Testmustergenerierung verwendet.

Für Transitionsfehler konnten, basierend auf SAT-Verfahren, Schaltungen mit über einer Million Gattern bearbeitet werden. Details zur Modellierung sind in [13] gegeben. Tabelle 2 gibt einen Laufzeitüberblick. Die Anzahl der zu betrachtenden Fehler (d. h. die Fehlermenge nach Fehlerkollabierung) bzw. die Anzahl der nicht testbaren Fehler sind in den Spalten zwei und drei gegeben. Die Gesamtlaufzeit wurde auf 24 Stunden begrenzt. Es zeigt sich, dass durch das neue Verfahren oft mit moderaten Laufzeiten sehr gute Ergebnisse erzielt werden können.

**Tabelle 2** Experimentelle Resultate für das Transitionsfehlermodell.

| Schaltkreis | Targets | Untest. | Abbrüche | Zeit |
|---|---|---|---|---|
| b14 | 40 086 | 322 | 0 | 3:00 m |
| b15 | 38 094 | 1632 | 0 | 4:31 m |
| b17 | 133 804 | 4095 | 0 | 20:53 m |
| b18 | 459 360 | 35 846 | 2 | 2:10 h |
| b20 | 80 606 | 514 | 0 | 8:02 m |
| b21 | 82 060 | 567 | 0 | 8:37 m |
| b22 | 119 810 | 624 | 0 | 12:08 m |
| p44k | 109 806 | 7456 | 23 | 8:31 h |
| p49k | 255 326 | | *Timeout* | |
| p80k | 311 416 | 1790 | 4 | 9:26 m |
| p88k | 256 050 | 3002 | 0 | 31:05 m |
| p99k | 274 376 | 17 975 | 1 | 12:29 m |
| p177k | 410 240 | | *Timeout* | |
| p462k | 1 134 924 | 390 706 | 775 | 11:03 h |
| p565k | 1 524 044 | 45 257 | 331 | 2:54 h |
| p1330k | 2 464 440 | 97 811 | 32 | 12:25 h |

## 7 Kombination von SAT mit FAN

Bisherige Untersuchungen haben insbesondere gezeigt, dass SAT-

**Tabelle 3** Experimentelle Resultate für das Haftfehlermodell.

| | | | SAT_4V | | SAT | | FAN | | FAN(long) | | FAN+SAT | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Schaltkreis | Targets | Untest. | Ab. | Zeit | Ab. | Zeit | Ab. | Zeit | Ab. | Zeit | Ab. | Zeit |
| b14 | 22 700 | 156 | 0 | 1:00 m | 0 | 0:19 m | 107 | 0:11 m | 7 | 1:42 m | 0 | 0:12 m |
| b15 | 21 850 | 727 | 0 | 1:16 m | 0 | 0:24 m | 619 | 0:11 m | 318 | 26:25 m | 0 | 0:18 m |
| b17 | 76 493 | 1958 | 0 | 4:36 m | 0 | 2:22 m | 1382 | 1:41 m | 622 | 56:54 m | 0 | 1:58 m |
| b18 | 264 043 | 2844 | 0 | 27:33 m | 0 | 22:30 m | 740 | 19:16 m | 270 | 41:40 m | 0 | 20:34 m |
| b20 | 45 461 | 319 | 0 | 2:30 m | 0 | 0:56 m | 225 | 0:35 m | 42 | 7:46 m | 0 | 0:44 m |
| b21 | 46 156 | 378 | 0 | 2:41 m | 0 | 0:59 m | 198 | 0:39 m | 43 | 6:48 m | 0 | 0:43 m |
| b22 | 67 540 | 344 | 0 | 3:49 m | 0 | 1:35 m | 284 | 1:07 m | 52 | 9:34 m | 0 | 1:14 m |
| p44k | 64 105 | 2385 | 0 | 2:18 h | 0 | 26:01 m | 12 | 4:58 m | 0 | 4:59 m | 0 | 5:55 m |
| p49k | 142 461 | 1089 | *Timeout* | | 77 | 1:43 h | 3770 | 2:06 h | 162 | 2:38 h | 74 | 1:55 h |
| p80k | 197 834 | 124 | 1 | 42:58 m | 0 | 9:43 m | 218 | 34:55 m | 21 | 39:13 m | 0 | 39:38 m |
| p88k | 147 048 | 2640 | 0 | 11:41 m | 0 | 9:33 m | 195 | 9:13 m | 38 | 12:40 m | 0 | 10:27 m |
| p99k | 162 019 | 2141 | 0 | 8:41 m | 0 | 6:50 m | 1398 | 6:02 m | 512 | 1:16 h | 0 | 7:25 m |
| p177k | 268 176 | 13 840 | 941 | 10:28 h | 0 | 1:19 h | 270 | 16:06 m | 47 | 20:03 m | 0 | 19:03 m |
| p462k | 673 465 | 132 249 | 129 | 3:31 h | 6 | 2:16 h | 1383 | 1:34 h | 423 | 2:07 h | 0 | 1:51 h |
| p565k | 1 025 273 | 28 287 | 0 | 2:23 h | 0 | 2:23 h | 1391 | 2:21 h | 85 | 2:47 h | 0 | 2:47 h |
| p1330k | 1 510 574 | 44 299 | 1 | 4:58 h | 1 | 5:05 h | 889 | 4:15 h | 144 | 4:28 h | 0 | 5:00 h |

basierte Verfahren oft besonders gut bei „harten“ Fehlern geeignet sind. Dies sind Fehler, bei denen klassische Verfahren lange Laufzeiten benötigen bzw. zu keinem Ergebnis kommen. Bei „leichten“ Fehlern hingegen sind klassische Verfahren klar im Vorteil. In diesem Sinne sind SAT-basierte Methoden komplementär und ergänzen die existierenden Verfahren. Eine naheliegende Idee ist somit die Kombination der Ansätze.

In Tabelle 3 sind Laufzeiten für die Testmustergenerierung mittels eines klassischen Algorithmus sowohl für industrielle als auch für frei verfügbare Schaltkreise angegeben. Erneut gibt Spalte *Targets* die Anzahl der zu betrachtenden und Spalte *Untest.* die Anzahl der untestbaren Fehlern an. Die Anzahl der Abbrüche und die benötigte Zeit sind in Spalte *Ab.* bzw. in Spalte *Zeit* angegeben.

Spalte *FAN* präsentiert die Ergebnisse eines klassischen Verfahrens. Zur Anwendung kam ein ATPG-Werkzeug der Firma NXP Semiconductors, das im Kern aus einem hochoptimierten Algorithmus, ähnlich zu FAN, besteht. Verglichen mit den Ergebnissen des SAT-basierten Verfahrens benötigt der klassische Ansatz oft weniger Laufzeit, ist aber oft nicht in der Lage, den Fehler zu klassifizieren.

Erhöht man die zur Verfügung stehenden Ressourcen, so erhält man die Resultate, die in Spalte *FAN(long)* angegeben sind. Wie erwartet reduziert sich auf Kosten höherer Laufzeit die Anzahl nicht klassifizierter Fehler. Im Vergleich zum SAT-basierten ATPG sind diese Verbesserungen aber nur marginal.

Übergibt man stattdessen die nicht klassifizierten Fehler des einfachen FAN-Verfahrens an den SAT-Beweiser, erhält man die Ergebnisse in der letzten Spalte. Es zeigt sich, dass die Qualität – gemessen in der Anzahl der klassifizierten Fehler – signifikant erhöht werden kann. Bemerkenswert ist weiterhin, dass außerdem in vielen Fällen die Laufzeit reduziert wird.

Genauere Beschreibungen befinden sich in [27].

## 8 Zusammenfassung

Die bisher durchgeführten Untersuchungen haben gezeigt, dass SAT-basierte ATPG-Verfahren eine erfolgversprechende Ergänzung zu klassischen Algorithmen darstellen. Erste Ansätze einer Kombination lieferten sehr gute Ergebnisse, die auch bei der Anwendung auf größere, industrielle Probleminstanzen überzeugen konnten.

Motiviert durch die bisherigen Resultate sollen im Folgenden untersucht werden:

- *Testspezifische SAT-Heuristiken* Erste Studien zeigten, dass sich die „standard“ SAT-Heuristiken für den ATPG-Prozess durch Schaltkreisinformationen verbessern lassen. Beispielsweise kann es sinnvoll sein, die Variablenauswahl auf primäre Eingänge zu beschränken. Testbarkeitsmaße wie Kontrollierbarkeit oder Beobachtbarkeit können hier ebenfalls Anwendung finden. Basierend auf diesen Ideen sollen neue Heuristiken entwickelt werden.
- *Integration der SAT-Beweiser* Der oben diskutierte Ansatz beschreibt bisher nur eine Hintereinanderausführung der Methoden. Eine tiefere Integration mit Informationsaustausch lässt bessere Ergebnisse erwarten.
- *Parallelität* Durch die steigende Verbreitung von Mehrkern-Prozessoren wird bei sequentiellen ATPG-Verfahren zur Verfügung stehende Rechenleis-

tung nicht genutzt. Daher sollen parallele Verfahren entwickelt werden, welche die vorhandene Anzahl der Prozessorkerne möglichst effizient auslasten.

## Danksagung

Die hier beschriebenen Arbeiten entstanden zum Teil mit Unterstützung von NXP Semiconductors Germany GmbH (Hamburg), DFG DR 287/16-1 und BMBF im Projekt MAYA, Vertragsnummer 01M3172B. Weiterhin möchten wir uns für die Hilfe bei der Umsetzung der Methoden und die zahlreichen fruchtbaren Diskussionen bei René Krenz-Bååth, Andreas Glowatz, Friedrich Hapke, Junhao Shi und Tim Warode bedanken.

## Literatur

[1] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In: *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 1579 of *LNCS*, pp. 193–207, 1999.

[2] D. Brand. Verification of large synthesized designs. In: *Int'l Conf. on CAD*, pp. 534–537, 1993.

[3] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. In: *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[4] S. A. Cook. The complexity of theorem proving procedures. In: *3. ACM Symp. on Theory of Computing*, pp. 151–158, 1971.

[5] F. Corno, M. Reorda, and G. Squillero. RT-level ITC 99 benchmarks and first ATPG results. In: *IEEE Design & Test of Comp.*, pp. 44–53, 2000.

[6] M. Davis, G. Logeman, and D. Loveland. A machine program for theorem proving. In: *Comm. of the ACM*, 5:394–397, 1962.

[7] R. Drechsler. *Advanced Formal Verification*. Kluwer Academic Publishers, 2004.

[8] R. Drechsler, S. Eggersglüß, G. Fey, A. Glowatz, F. Hapke, J. Schloeffel, and D. Tille. On acceleration of SAT-based ATPG for industrial designs. In: *IEEE Trans. on CAD*, 27:1329–1333, 2008.

[9] N. Eén and N. Sörensson. An extensible SAT solver. In: *SAT 2003*, vol. 2919 of *LNCS*, pp. 502–518, 2004.

[10] S. Eggersglüß and R. Drechsler. Improving test pattern compactness in SAT-based ATPG. In: *Asian Test Symp.*, pp. 445–452, 2007.

[11] S. Eggersglüß, G. Fey, and R. Drechsler. SAT-based ATPG for path delay faults in sequential circuits. In: *IEEE Int'l Symp. on Circuits and Systems*, 2007.

[12] S. Eggersglüß, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schloeffel. Combining multi-valued logics in SAT-based ATPG for path delay faults. In: *ACM & IEEE Int'l Conf. on Formal Methods and Models for Codesign (MEMOCODE)*, pp. 181–187, 2007.

[13] S. Eggersglüß, D. Tille, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schlöffel. Experimental studies on SAT-based ATPG for gate delay faults. In: *Int'l Symp. on Multi-Valued Logic*, 2007.

[14] G. Fey, J. Shi, and R. Drechsler. Efficiency of multiple-valued encoding in SAT-based ATPG. In: *Int'l Symp. on Multi-Valued Logic*, page 25 (6 pages), 2006.

[15] G. Fey, T. Warode, and R. Drechsler. Using structural learning techniques in SAT-based ATPG. In: *VLSI Design Conf.*, pp. 69–74, 2007.

[16] H. Fujiwara and T. Shimono. On the acceleration of test generation algorithms. In: *IEEE Trans. on Comp.*, 32:1137–1144, 1983.

[17] P. Goel. An implicit enumeration algorithm to generate tests for combinational logic. In: *IEEE Trans. on Comp.*, 30:215–222, 1981.

[18] E. Goldberg and Y. Novikov. BerkMin: a fast and robust SAT-solver. In: *Design, Automation and Test in Europe*, pp. 142–149, 2002.

[19] T. Larrabee. Test pattern generation using Boolean satisfiability. In: *IEEE Trans. on CAD*, 11:4–15, 1992.

[20] J. P. Marques-Silva and K. A. Sakallah. GRASP: A search algorithm for propositional satisfiability. In: *IEEE Trans. on Comp.*, 48:506–521, 1999.

[21] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In: *Design Automation Conf.*, pp. 530–535, 2001.

[22] J. P. Roth. Diagnosis of automata failures: A calculus and a method. In: *IBM J. Res. Dev.*, 10:278–281, 1966.

[23] O. Shtrichman. Pruning techniques for the SAT-based bounded model checking problem. In: *CHARME*, vol. 2144 of *LNCS*, pp. 58–70, 2001.

[24] P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Combinational test generation using satisfiability. In: *IEEE Trans. on CAD*, 15:1167–1176, 1996.

[25] P. Tafertshofer, A. Ganz, and K. Antreich. Igraine – an implication graph based engine for fast implication, justification, and propagation. In: *IEEE Trans. on CAD*, 19:907–927, 2000.

[26] D. Tille and R. Drechsler. Incremental SAT instance generation for SAT-based ATPG. In: *IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, pp. 68–73, 2008.

[27] D. Tille, S. Eggersglüß, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schlöffel. Studies on integrating SAT-based ATPG in an industrial environment. In: *GI/ITG Workshop „Testmethoden und Zuverlässigkeit von Schaltungen und Systemen"*, 2007.

[28] D. Tille, G. Fey, and R. Drechsler. Instance generation for SAT-based ATPG. In: *IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, pp. 153–156, 2007.

[29] D. Tille, R. Krenz-Bååth, J. Schloeffel, and R. Drechsler. Improved circuit-to-CNF transformation for SAT-based ATPG. In: *GI/ITG Workshop „Testmethoden und Zuverlässigkeit von Schaltungen und Systemen"*, pp. 25–29, 2008.

[30] G. Tseitin. On the complexity of derivation in propositional calculus. In: *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pp. 115–125, 1968. (Reprinted in: J. Siekmann, G. Wrightson (Ed.), Automation of Reasoning, Vol. 2, Springer, Berlin, 1983, pp. 466–483).

[31] J. Whittemore, J. Kim, and K. Sakallah. SATIRE: A new incremental satisfiability engine. In: *Design Automation Conf.*, pp. 542–545, 2001.


1


2


3


4


5

**1 Prof. Dr. Rolf Drechsler** hat an der Johann-Wolfgang-Goethe Universität in Frankfurt am Main Mathematik und Informatik studiert. Im Jahr 1995 schloss er sein Studium mit der Promotion ab. Von 1995 bis 2000 war er als Wissenschaftlicher Assistent an der Albert-Ludwigs-Universität in Freiburg beschäftigt, wo er 1999 habilitierte. Im Anschluss arbeitete er als Senior Engineer in der Abteilung Corporate Technologies der Siemens AG, München, bis er im Oktober 2001 die Professur für Rechnerarchitektur am Fachbereich 3, Mathematik und Informatik, der Universität Bremen annahm. Seine wissenschaftlichen Interessen umfassen den Schaltkreis- und Systementwurf mit dem Schwerpunkt Formale Verifikation.
Adresse: Universität Bremen, Bibliothekstraße 1, 28359 Bremen, Germany, Tel.: +49-421-21863932, Fax: +49-421-2187385, E-Mail: drechsle@informatik.uni-bremen.de

**2 Dipl.-Inf. Stephan Eggersglüß** studierte Informatik an der Universität Bremen, wo er 2006 sein Diplom erhielt. Seit 2006 ist er dort als wissenschaftlicher Mitarbeiter in der AG Rechnerarchitektur tätig. Seine Forschungsinteressen liegen im Test von digitalen Schaltkreisen sowie in formalen Beweistechniken.
Adresse: Universität Bremen, Bibliothekstraße 1, 28359 Bremen, Germany, Tel.: +49-421-21863936, Fax: +49-421-2187385, E-Mail: segg@informatik.uni-bremen.de

**3 Dr. Görschwin Fey** erhielt sein Diplom in Informatik 2001 von der Martin-Luther-Universität Halle-Wittenberg. Seit 2002 ist er als wissenschaftlicher Mitarbeiter in der AG Rechnerarchitektur an der Universität Bremen tätig. Seine Promotion hat er im Jahr 2006 unter der Betreuung von Prof. Dr. Rolf Drechsler abgeschlossen. Von November 2007 bis Juni 2008 hatte Görschwin Fey eine Gastprofessur an der Universität Tokyo, Japan inne. Seine Forschungsinteressen liegen in den Bereichen Verifikation und Test von digitalen Schaltkreisen.
Adresse: Universität Bremen, Bibliothekstraße 1, 28359 Bremen, Germany, Tel.: +49-421-21863944, Fax: +49-421-2187385, E-Mail: fey@informatik.uni-bremen.de

**4 Dipl.-Phys. Jürgen Schlöffel** studierte Physik an der Universität in Göttingen, wo er 1986 sein Diplom erhielt. Nach verschiedenen Tätigkeiten in den Bereichen Charakterisierung mikroelektronischer Bauelemente, Library Entwicklung und CAD ist er heute Project Manager und Principal im Bereich EDA und DFT. Seine Hauptinteressen liegen im Bereich der Entwicklung neuer Tools und Methoden für den automatisierten Test und die Diagnose. Er ist Mitglied bei IEEE, VDE, Mitglied der ITG Arbeitsgruppe Test und Zuverlässigkeit und Mitglied der Steuerungsgruppe des deutschen edacentrums.
Adresse: Mentor Graphics Development (Deutschland) GmbH, Tempowerkring 1B, 21079 Hamburg, Germany, Tel.: +49-40-79012808, Fax: +49-40-79012829, E-Mail: juergen_schloeffel@mentor.com

**5 Dipl.-Inform. Daniel Tille** studierte Informatik an der Martin-Luther-Universität Halle-Wittenberg, wo er 2006 sein Diplom erhielt. Seit 2006 ist er als wissenschaftlicher Mitarbeiter in der AG Rechnerarchitektur an der Universität Bremen tätig. Seine Forschungsinteressen liegen im Test von digitalen Schaltkreisen sowie in formalen Beweistechniken.
Adresse: Universität Bremen, Bibliothekstraße 1, 28359 Bremen, Germany, Tel.: +49-421-21863938, Fax: +49-421-2187385, E-Mail: tille@informatik.uni-bremen.de