Hochoptimierter Ablauf zur Robustheitsprüfung

Stefan Frehse Finn Haedicke Melanie Diepenbeck Görschwin Fey Rolf Drechsler
Arbeitsgruppe Rechnerarchitektur, Fachbereich 3 – Mathematik und Informatik
Universität Bremen, 28359 Bremen, Germany
{sfrehse,finn,diepenbeck,fey,drechsle}@informatik.uni-bremen.de

ZUSAMMENFASSUNG

Zunehmend werden digitale Schaltkreise in sicherheitskritischen Bereichen eingesetzt, was ein korrekt funktionierendes System erfordert. Die stetige Verringerung der Strukturgrößen führt aufgrund von Umgebungsstrahlung oder Fertigungsungenauigkeiten immer häufiger zu transienten Fehlern in digitalen Schaltkreisen. Maßnahmen, um transiente Fehler zu tolerieren, sind seit Längerem verfügbar. Jedoch können Fehler bei der Implementierung solcher Techniken auftreten. Demnach muss die Implementierung hinsichtlich der Robustheit gegenüber transienten Fehlern bewertet und auf Korrektheit überprüft werden.
In der vorliegenden Arbeit wird ein hochoptimierter Ablauf zur Bestimmung der Robustheit digitaler Schaltkreise vorgestellt. Zufallssimulation sowie das Ausnutzen strukturellen Wissens in Verbindung mit formalen Beweistechniken werden für die Robustheitsprüfung angewendet. Die Experimente zeigen, dass die Effektivität des Verfahrens deutlich gesteigert wird, wenn verschiedene Techniken integriert werden, wobei die Qualität der Analyse gleich hoch bleibt.

ABSTRACT

Digital circuits are increasingly used in safety critical applications which demand for completely correct systems. Due to the continously shrinking feature sizes transient faults occur more often caused e.g., by radiation or process variation.
Techniques to tolerate transient faults are already available. However, during implementation bugs may be introduced. Consequently, the implemenation has to be verified with respect to transient faults.
This work presents a highly-optimized flow for computing the robustness of digital circuits. Random simulations as well as structural knowledge combined with formal methods are used. The experiments show that in particular combining those techniques the effectiveness is significantly increased while the quality is kept high.

I. EINFÜHRUNG

Digitale Schaltkreise werden in vielen Bereichen eingesetzt. Gerade in sicherheitskritischen Bereichen, wie z.B. in Flugzeugen, ist die vollständig korrekte Funktion der Schaltkreise erforderlich. Jedoch werden immer höhere Anforderungen hinsichtlich des Funktionsumfangs an die Schaltkreise gestellt. Einhergehend damit nimmt die Komplexität der Systeme zu, wobei stetig mehr Komponenten bei gleich bleibender Fläche auf einen Chip integriert werden. Ein damit verbundener Nachteil ist beispielsweise die immer stärker werdende Anfälligkeit gegenüber Umgebungsstrahlung. Das Auftreffen dieser Strahlung verursacht eine kurzzeitige Funktionsänderung des Systems durch die Invertierung eines Wertes einer bestimmten Signalleitung, d.h. sogenannte *transiente Fehler* treten auf. Die Funktionalität des Schaltkreises kann dadurch beeinflusst werden.

Um mit solchen Effekten umgehen zu können, wurden bereits Techniken vorgestellt, die die Schaltkreise auf Entwurfsebene bzgl. transienter Fehlern härten [1]. Weitere Beispiele dafür sind Dreifach-Redundanz (engl.: *Triple Modular Redundancy* TMR) oder fehlerkorrigierende und fehlererkennde Codes (engl.: *Error Correcting/Detecting Codes* ECC/EDC) wie z.B. der Hamming - Code.

Diese Arbeit wurde durch die Deutsche Forschungsgemeinschaft unter den Kennzeichen DR 287/19-1 und FE 797/5-1 unterstützt.

Die Implementierung solcher Systeme kann jedoch fehlerbehaftet sein und muss demnach verifiziert werden, um den Grad der Robustheit zu messen bzw. sicherzustellen.

Die *automatische Testmuster-Generierung* (engl.: *Automatic Testpattern Generation* - ATPG) für Schaltkreise wird in der Industrie seit Längerem in einem hochoptimierten Ablauf verschiedener Algorithmen auf sehr großen Schaltkreisen angewandt [2]. Insbesondere durch die starke Integration dieser Algorithmen ist ATPG sehr effektiv. Das Problem wird dazu mittels sogenannter Scan-Ketten auf den kombinatorischen Fall reduziert.

Da der Ablauf aus dem ATPG sehr effektiv ist, ist eine Adaptation auf die Robustheitsprüfung sinnvoll, wobei die Problemstellung verschieden ist. Die Analyse der Auswirkungen von Fehlern in sequentiellen Schaltkreisen ist ein inhärent schwereres Problem als kombinatorisches ATPG. Zusätzlich sind die Probleminstanzen meist schwierig, da durch die implementierten Schutzmaßnahmen fast alle Fehler nicht testbar und somit redundant sind.

Zur Analyse über die starke Fehlersicherheit von Schaltkreisen wurde ein Verfahren auf Basis von ATPG vorgestellt, jedoch nur für den kombinatorischen Fall [3]. Die Verfahren aus [4], [5] betrachten ebenfalls sequentielle Schaltkreise zur Analyse über die Auswirkung von verschiedenen Fehlern, wobei diese sehr aufwändig sind.

Tabelle I
FEHLERÄQUIVALENZEN DES UND- UND ODER-GATTERS [6]

Gatter	Äquivalenzen
$o = \text{UND}(i_1, \ldots, i_k)$	$\{(i_1$, s-a-0), …, $(i_k$, s-a-0), $(o$, s-a-0)$\}$
$o = \text{ODER}(i_1, \ldots, i_k)$	$\{(i_1$, s-a-1), …, $(i_k$, s-a-1), $(o$, s-a-0)$\}$

Die vorliegende Arbeit stellt einen hochoptimierten Ablauf zur Berechnung der Robustheit eines Schaltkreises vor. Dieser Ablauf integriert dazu:

1) Approximation mittels formaler Methoden
2) Zufallssimulation
3) Fehlerimplikation bzgl. transienter Fehler
4) Strukurelles Wissen

Diese Techniken werden zu einem Ablauf zusammengefasst, der die Verwendung und die Laufzeit der Robustheitsprüfung verbessert, wobei die hohe Qualität der Analyse gleich bleibt.

Die Arbeit ist wie folgt gegliedert: Die Grundlagen werden in Abschnitt II vorgestellt. Abschnitt III beschreibt den Ablauf zur Robustheitsprüfung, wobei die integrierten Techniken in Abschnitt IV die Fehlerimplikation, V die Zufallssimulation und schließlich in Abschnitt VI die strukturelle Analyse beschrieben werden. Im Abschnitt VII wird der neue Ablauf in die Berechnung der Robustheit eingebettet. Die experimentellen Ergebnisse werden in Abschnitt VIII vorgestellt und die Arbeit in Abschnitt IX zusammengefasst.

II. GRUNDLAGEN

A. Schaltkreismodell

In dieser Arbeit werden sequentielle Schaltkreise, dargestellt durch $\mathbb{C}$ betrachtet. Eine Komponente $g \in \mathbb{C}$ kann dabei ein Gatter oder ein komplexeres Modul sein. Die primären Eingänge werden durch $\text{PI}(\mathbb{C})$, die primären Ausgänge durch $\text{PO}(\mathbb{C})$ und die Zustandselemente durch $\text{FF}(\mathbb{C})$ dargestellt.

B. Fehleräquivalenzen

Im ATPG wird häufig das Haftfehlermodell (engl.: *Single Stuck-At Fehlermodell* - SAFM) verwendet. Dabei kann der Wert einer Signalleitung entweder konstant 0 (engl.: stuck-at-0, s-a-0) oder konstant 1 (engl.: stuck-at-1, s-a-1) sein. Um Testmuster für einen Schaltkreis zu erzeugen wird eine *Fehlerliste* angelegt, die dann durch ATPG abgearbeitet wird, um für jeden Fehler einen Test bereitzustellen.

Mithilfe von Äquivalenzrelationen zwischen den Fehlern wird die Fehlerliste reduziert. Das Verfahren, welches diese Beziehungen ausnutzt ist in [6, Kapitel 3, Seite 75ff] ausführlicher beschrieben. Dabei gilt: Zwei Fehler f_1 und f_2 sind genau dann äquivalent, wenn jedes Testmuster, das f_1 erkennt auch f_2 erkennt und umgekehrt. Als Beispiel sind die Äquivalenzrelation für das UND- und ODER-Gatter in Tabelle I angegeben. Alle äquivalenten Fehler bis auf einen können aus der Fehlerliste gelöscht werden, wodurch die Fehlerliste reduziert wird.

Algorithmus 1: boundsRob($\mathbb{C}, t^d$)

```
Input : C, t^d ∈ ℕ
Output : Bounds of Robustness
1 begin
2     for i = 1; i ≤ t^d; i++ do
3         (Ŝ^i, T̂^i, D̂^i) = Rob(C, i, S↓, ∅, ∅);
4     end
5     for i = 1; i ≤ t^d; i+ = 1 do
6         (Ŝ^i, T̂^i, D̂^i) = Rob(C, i, S↑, ∅, ∅);
7     end
8     return (R_lb^{t^d}, R_ub^{t^d})
9 end
```

C. Verfahren zur Robustheitsprüfung

Dieser Abschnitt beschreibt das Verfahren zur Berechnung der Robustheit aus [7]. Die darin beschriebene Analyse klassifiziert Komponenten unter einer Fehlerannahme hinsichtlich der Beobachtbarkeit an Zustandselementen bzw. Ausgängen. Als Fehlermodell werden transiente Fehler betrachtet. Dazu wird eine nicht-deterministische Änderung einer Signalleitung zu einem Zeittakt modelliert. Die Komponenten werden dabei als einfache Gatter oder komplexere Module verstanden.

Implementierte Schutzmaßnahmen signalisieren unregelmäßiges Verhalten des Schaltkreises durch ein Fehlersignal, wodurch Gegenmaßnahmen, beispielsweise zum Neustart, eingeleitet werden können. Dies ist in dem Verfahren in die Modellierung mit einbezogen.

Die Robustheitsprüfung berechnet wie sich der Schaltkreis unter einem nicht-deterministischen Fehler an einer Komponente über die Zeit verhält. Dabei werden die Komponenten des Schaltkreises verschiedenen Klassen zugeordnet, d.h. die Komponenten werden klassifiziert, was schließlich ein Robustheitsmaß definiert. Eine Komponente g aus Schaltkreis $\mathbb{C}$ sei dabei wie folgt eingeteilt:

- *nicht robust*: Mindestens ein Fehler an der Komponente g ist am Ausgang des Schaltkreises beobachtbar und das Fehlersignal meldet keinen Fehler.
- *gefährlich*: Mindestens ein Fehler an Komponente g verändert den Zustand des Systems, wobei kein Fehler an den Ausgängen beobachtbar ist bzw. durch das Fehlersignal signalisiert wird. Dieses Verhalten wird auch als engl. *Silient Data Corruption* (SDC) bezeichnet.
- *robust:* Andernfalls, d.h. es gibt keinen Fehler, der sich an den Ausgängen bemerkbar macht und nicht von der Fehlererkennungslogik erkannt oder maskiert wird.

Basierend auf dieser Klassifikation werden die Komponenten des Schaltkreises partitioniert: $\mathbb{S}$ beinhaltet nicht robuste Komponenten, $\mathbb{D}$ gefährliche Komponenten und $\mathbb{T}$ robuste Komponenten. Berechnungsmodelle, die diese Klassifikation durchführen, wurden auf Basis Boolescher Erfüllbarkeit vorgestellt. Dazu wird der

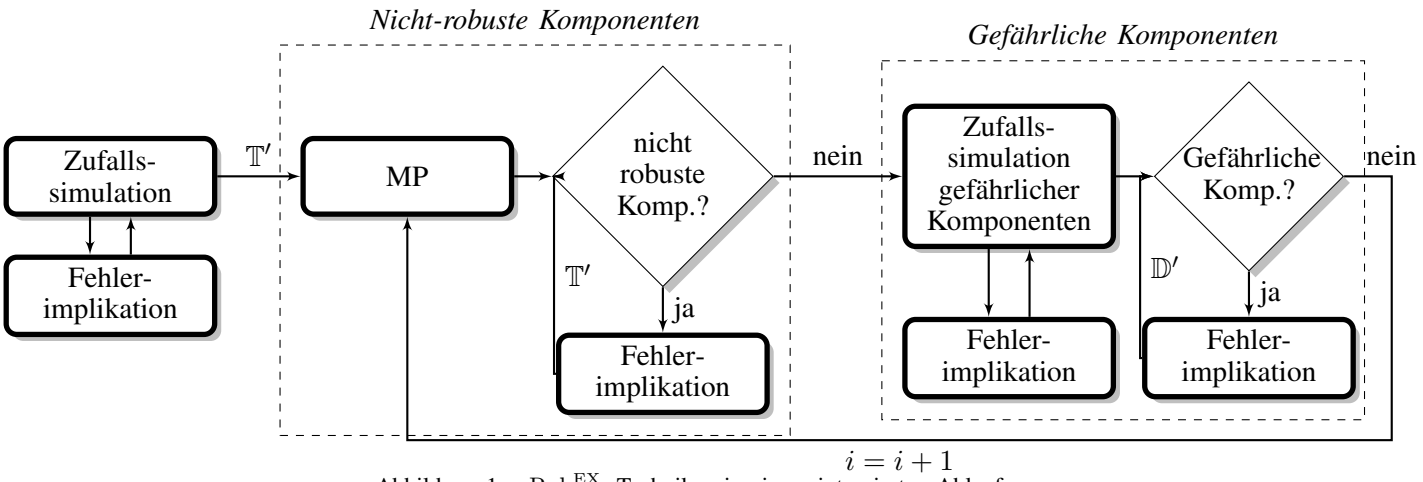

Abbildung 1. $\mathrm{Rob}^{\mathrm{EX}}$; Techniken in einem integrierten Ablauf

Schaltkreis schrittweise über die Zeit betrachtet bis ein vordefiniertes Beobachtungsfenster $t^d \in \mathbb{N}$ ausgeschöpft ist. Das bedeutet, in jeder Iteration können immer mehr Komponenten als nicht robust bzw. robust klassifiziert werden, da sich der Fehler letztendlich am Ausgang zeigt, korrigiert oder erkannt wird, was durch das Fehlersignal signalisiert wird.

Die Klassifikation für den Zeittakt i mit $1 \leq i \leq t^d$ sei durch $\mathbb{S}^i$ für nicht robuste, $\mathbb{D}^i$ für gefährliche, $\mathbb{T}^i$ für robuste Komponenten und $\mathbb{U}$ noch nicht betrachtete Komponenten dargestellt. Basierend auf der Klassifikation für den Zeittakt i kann ein Robustheitsmaß wie folgt angegeben werden:

$$R^i_{lb} = \frac{|\mathbb{T}^i|}{|\mathbb{C}|} = 1 - \frac{|\mathbb{S}^i \cup \mathbb{D}^i \cup \mathbb{U}|}{|\mathbb{C}|} \quad (1)$$

$$R^i_{ub} = \frac{|\mathbb{T}^i \cup \mathbb{D}^i \cup \mathbb{U}|}{|\mathbb{C}|} = 1 - \frac{|\mathbb{S}^i|}{|\mathbb{C}|} \quad (2)$$

Die Klassifikation ist vollständig bzgl. des Beobachtungsfensters, wenn $i = t^d$ gilt, d.h., wenn das gesamte Beobachtungsfenster betrachtet worden ist. Weiterhin ist die Klassifikation exakt, wenn für den Zustand der Fehlerinjektion $S(0)$ alle erreichbaren Zustände betrachtet werden. Eine Funktion zur Klassifikation der Komponente sei im Folgenden durch

$$\mathrm{Rob}(\mathbb{C}, i, S(0), \mathbb{S}, \mathbb{T})$$

dargestellt und liefert die Klassifikationen als Tupel zurück, d.h. $(\mathbb{S}^i, \mathbb{D}^i, \mathbb{T}^i) = \mathrm{Rob}(\mathbb{C}, i, S(0), \mathbb{S}, \mathbb{T})$. Die Parameter dabei sind: der Schaltkreis $\mathbb{C}$, das aktuelle Beobachtungsfenster i, die Bedingungen der zu betrachtenden Zustände für die Fehlerinjektion $S(0)$. Die Menge $\mathbb{S}$ und $\mathbb{T}$ sind bereits vorgegebene robuste und nicht-robuste Komponenten, die bei der Klassifikation nicht mehr berücksichtigt werden müssen.

Zur exakten Berechnung der Robustheit muss die gesamte Menge der erreichbaren Zustände betrachtet werden, dargestellt durch S^*. Die Berechnung dieser Menge ist allerdings sehr platz- und zeitaufwändig. Die Robustheitsprüfung kann jedoch mit Approximationen dieser Menge umgehen, d.h. $S^{\downarrow} \subseteq S^* \subseteq S^{\uparrow}$. $S^{\downarrow}$ stellt eine Unterapproximation und $S^{\uparrow}$ eine Überapproximation erreichbarer Zustände dar. Seien die Mengen der Klassifikationen basierend auf einer Überapproximation bzw. Unterapproximation wie folgt bezeichnet:

- $\check{\mathbb{S}}^i$, $\check{\mathbb{D}}^i$ und $\check{\mathbb{T}}^i$ die Mengen der Klassifikationen des Zeittakts i auf Basis einer Unterapproximation
- $\hat{\mathbb{S}}^i$, $\hat{\mathbb{D}}^i$ und $\hat{\mathbb{T}}^i$ die Mengen der Klassifikationen des Zeittakts i auf Basis einer Überapproximation
- $\dot{\mathbb{S}}^i$, $\dot{\mathbb{D}}^i$ und $\dot{\mathbb{T}}^i$ die Mengen der Klassifikationen des Zeittakt i auf Basis der exakten Menge der erreichbaren Zustände

Dazu gelten folgende Beziehungen: $\check{\mathbb{S}}^i \supseteq \dot{\mathbb{S}}^i \supseteq \hat{\mathbb{S}}^i$ und $\check{\mathbb{T}}^i \subseteq \dot{\mathbb{T}}^i \subseteq \hat{\mathbb{T}}^i$ [7].

Die Schranken der Robustheit werden durch Algorithmus 1 berechnet. Dazu werden der Schaltkreis $\mathbb{C}$ und das Beobachtungsfenster als Eingabe übergeben. Zunächst wird die Robustheitsprüfung auf Basis der Unterapproximation $S^{\downarrow}$ der erreichbaren Zustände durchgeführt (Zeilen 2–4). Anschließend wird eine Überapproximation $S^{\uparrow}$ zur Berechnung verwendet (Zeilen 5–7). Der Algorithmus liefert letztendlich als Ergebnis die Schranken der Robustheit des Schaltkreises $\mathbb{C}$ mit Hilfe der Formeln (1) und (2).

III. Ablauf

In diesem Abschnitt wird der Ablauf $\mathrm{Rob}^{\mathrm{EX}}$ zur Robustheitsprüfung vorgestellt, welcher verschiedene neue Techniken und das vorgestellte Verfahren aus Abschnitt II-C integriert.

Zunächst ist in Abbildung 1 der Ablauf veranschaulicht. Die formale Analyse wird durch Rauten dargestellt. Im Gegensatz zu Algorithmus 1 ist die Berechnung der einzelnen Zeittakte in diesem Ablauf eng miteinander verbunden. Zu Beginn der Analyse wird eine Zufallssimulation als Vorverarbeitung ausgenutzt, um einfach

zu klassifizierende Komponenten zu finden. In Verbindung damit steht das Ausnutzen der Fehlerimplikation, um weitere Komponenten ohne weiteres Simulieren zu klassifizieren. Alle hierbei nicht robust klassifizierten Komponenten ($\mathbb{T}'$) müssen in den Folgeschritten nicht weiter betrachtet werden, was den Suchraum deutlich verkleinern kann.

Im Folgenden wird die Klassifikation der verbleibenden Komponenten gestartet. Dazu werden zuerst die nicht robusten Komponenten und anschließend die gefährlichen Komponenten klassifiziert.

Folgende Techniken werden in den nächsten Abschnitten vorgestellt.

- *Fehlerimplikation* beschleunigt die Klassifikation nicht robuster und gefährlicher Komponenten.
- *Zufallssimulation* dient als schnelle Vorverarbeitung für eine grobe Klassifikation.
- Der *Minimale Propagationspfad (MP)* bietet die Möglichkeit die Klassifikation für ein bestimmtes Beobachtungsfenster zu überspringen, wenn aus struktureller Sicht keine Propagation bis zu den Ausgängen innerhalb dieses Fensters möglich ist.

Die Integration der Techniken beschleunigt die Klassifikation robuster und nicht robuster Komponenten bei gleich bleibend hoher Qualität. Die einzelnen Techniken werden im Folgenden detailliert vorgestellt.

IV. Fehlerimplikation

Das Ausnutzen von Äquivalenzrelationen aus dem ATPG kann nicht direkt in der Robustheitsprüfung angewandt werden. Beim ATPG wird vorab eine Fehlerliste erzeugt und durch Anwenden der Äquivalenzrelationen minimiert. Das Erzeugen der Fehlerliste für die Robustheitsprüfung ist nicht erforderlich, da nach keinem konkreten Fehler gesucht wird. Ein beliebiger Fehler reicht aus um eine Komponente als nicht robust zu klassifizieren. Dennoch kann die Äquivalenzrelation ausgenutzt werden, um die Klassifikationen der Komponenten zu beschleunigen. Die generelle Idee besteht darin, die Äquivalenzregel auszunutzen, sobald ein konkreter Fehler einer Komponenten gefunden wurde. Im Gegensatz zum ATPG, wo die Fehlerliste vorab statisch minimiert wird, wird dies dynamisch während der Klassifikation durchgeführt. Das bedeutet, sobald eine Komponente klassifiziert worden ist, können ggf. weitere anhand des konkreten Fehlers impliziert werden. Im Folgenden wird dies *Fehlerimplikation* genannt.

Die Äquivalenzrelationen, aus dem ATPG werden dazu auf das Fehlermodell der transienten Fehler übertragen. Die Implikationen, die beispielsweise für ein UND-Gatter mit $o = \text{UND}(i_1, \ldots, i_k)$ durchgeführt werden können, werden im Folgenden beschrieben. Angenommen der Wert des Ausgangs ist $o = 1$, d.h. alle Eingänge sind ebenfalls mit einer 1 belegt, d.h. $i_l = 0$. Eine Fehlerinjektion invertiert den Wert am Ausgang, um einen transienten Fehler zu simulieren, d.h. $o' = 0$.

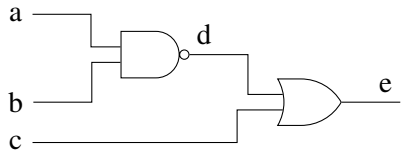

Abbildung 2. Beispielschaltkreis

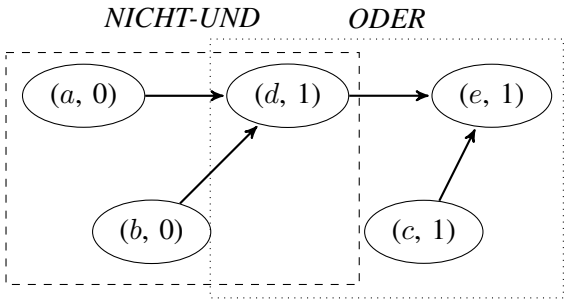

Abbildung 3. Implikationsgraph für Beispielschaltkreis

Wird die Komponente als nicht robust mit diesem Fehler klassifiziert, können alle Komponenten, die mit dem UND-Gatter verbunden sind, d.h. die Eingänge, ebenfalls als nicht robust klassifiziert werden, da sie die selben Testmuster haben und alle Eingänge der Komponente $i_1, \ldots, i_k$ mit einer 1 belegt sind. Diese Implikationen werden nun rekursiv mit den jeweiligen Gattern in Richtung der Eingänge fortgeführt, sodass Schritt für Schritt alle Äquivalenzrelationen ausgenutzt werden. Analoge Implikationsregeln werden für andere Gattertypen sowie für die Klassifikation gefährlicher Komponenten aufgestellt.

Definition 1 (Implikationsgraph): Ein *Implikationsgraph* (IG) ist ein gerichteter azyklischer Graph (DAG) $G = (V, E)$ mit:

- V ist die Menge alle möglichen transienten Fehler für alle Komponenten, d.h. $(g, s) \in V$ wobei g eine Komponente und $s \in \{0, 1\}$ ein Fehler ist,
- $E \subseteq V \times V$ und ein Tupel (v_1, v_2) mit $v_1, v_2 \in V$ ist genau dann in E, wenn eine Implikationsregel von v_1 nach v_2 gilt.

Der Implikationsgraph für den Beispielschaltkreis aus Abbildung 2 ist in Abbildung 3 dargestellt.

Der Fehler $(a, 0)$, sowie der Fehler $(b, 0)$ implizieren beide den Fehler $(d, 1)$. Dies ergibt sich aus der Implikationsregel für ein NICHT-UND Gatter. Für das ODER Gatter ergeben sich die Verbindungen zwischen dem Fehler $(c, 1)$ und $(e, 1)$, sowie $(d, 1)$ und $(e, 1)$, da die Implikationsregel für ODER aussagt, dass wenn ein Fehler mit einer 1 auf einem Eingang liegt, der Fehler mit einer 1 am Ausgang impliziert wird.

Der Algorithmus für die Erstellung eines Implikationsgraphen ist in Algorithmus 2 skizziert. Zunächst wird eine topologische Sortierung rückwärts von den Ausgängen aller Komponenten im Schaltkreis aufgebaut. Anschließend werden die einzelnen Komponenten in

Algorithmus 2: Generierung des Implikationsgraphen IG

```
   Input : ℂ, Implikationsregeln I
   Output : Implication Graph IG
1  Q_ℂ = createTopologicalOrder(ℂ);
2  foreach component g ∈ Q_ℂ do
3      foreach (i_val, o_val) ∈ getImplicationRules(g, I) do
4          foreach (i, o) ∈ inputs(g) × outputs(g) do
5              node_i = get_or_create_node(IG, i, i_val);
6              node_o = get_or_create_node(IG, o, o_val);
7              add_edge(IG, node_i, node_j);
8          end
9      end
10 end
```

aufsteigender Reihenfolge behandelt. Sofern die Implikationsregeln für diesen Komponententyp eine nicht leere Menge ist, wird für jede Eingang-Ausgang-Kombination die Implikationsregel angewendet (vgl. Zeile 4) und eine entsprechende Kante dem Implikationsgraphen hinzugefügt (vgl. Zeile 7).

In dem so erzeugten Implikationsgraphen wird für jede klassifizierte Komponente nachgeschlagen, welche weiteren Fehler durch den Fehler impliziert werden. Diese Komponenten können direkt klassifiziert werden, sodass eine eigenständige Prüfung entfallen kann.

V. Zufallssimulation

Die Zufallssimulation wird verwendet, um die Klassifikation hinsichtlich der nicht robusten und gefährlichen Komponenten zu beschleunigen. Das bedeutet, die Zufallssimulation berechnet eine grobe Klassifikation, die letztendlich durch die formalen Methoden vervollständigt wird, um eine exakte Klassifikation zu erreichen. Je mehr Komponenten die Simulation klassifiziert, desto weniger Komponenten müssen in der formalen Analyse betrachtet werden, was die Laufzeit deutlich reduzieren kann.

A. Fehlerinjektion

Eine nicht-deterministische Änderung eines Wertes einer Signalleitung in Anlehnung an das vorgestellte Fehler- und Komponentenmodell, erfordert eine spezielle Behandlung von Fehlerinjektionen bei der Zufallssimulation. Zum Beispiel: eine Komponente mit einem 32 Bit Ausgang, hat insgesamt $2^{32} - 1$ fehlerhafte Wertebelegungen. Aufgrund der großen Anzahl an Möglichkeiten, kann in der Praxis keine vollständige Analyse hinsichtlich der Fehlerinjektion durch die Simulation durchgeführt werden. Dabei werden in dieser Arbeit die Fehlerinjektionen zufällig gewählt.

Weiterhin darf im Hinblick auf transiente Fehler die Fehlerinjektion nur für einen Zeittakt erfolgen. Das bedeutet, es wird ein Fehler für einen Zeittakt injiziert und über mehrere Takte ohne erneute Fehlerinjektion propagiert.

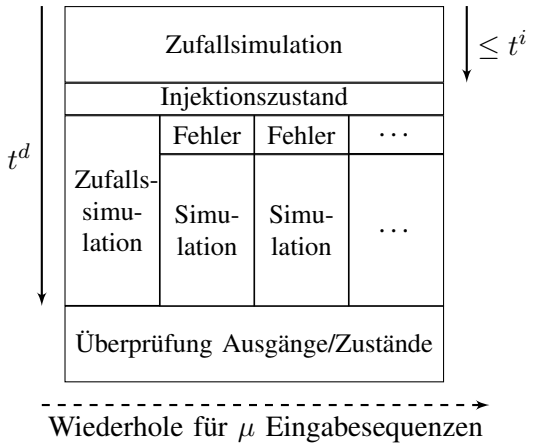

Abbildung 4. Ablauf der Zufallssimulation $\mathrm{Sim}(\mathbb{C}, t^d, t^i, M, \mu)$

B. Klassifikation

Die Zufallssimulation führt eine Klassifikation durch, indem zufällige Stimuli an den Schaltkreis angelegt und simuliert werden. Dabei wird überprüft, ob die Injektion eines Fehlers an einer Komponente an den Ausgängen bzw. an den Zuständen des Schaltkreises beobachtbar ist.

In Abbildung 4 ist dazu die Zufallssimulation vereinfacht dargestellt. Die Funktion $\mathrm{Sim}(\mathbb{C}, t^d, t^i, M, \mu)$, bekommt 1) den Schaltkreis $\mathbb{C}$, 2) das Beobachtungsfenster t^d, 3) die maximale Anzahl an Zeittakten nachdem ein Fehler injiziert wird t^i, 4) die noch nicht klassifizierten Komponenten M und 5) die maximale Anzahl an Stimuli μ als Eingabe.

Der Algorithmus startet mit der Simulation von bis zu t^i Zeittakten. Der Zustand der nach den t^i Takten eingestellt ist, dient als Ausgangspunkt für die Fehlerinjektion und der weiteren Simulation von t^d Zeittakten. Anschließend werden zufällige Eingaben generiert, die sowohl für die Simulation ohne Fehler, d.h. für den korrekten Wert als auch für die Simulation mit einem injizierten Fehler verwendet werden. Für jeden simulierten Takt wird nun überprüft, ob sich die Ausgänge bzw. die Zustände von den Referenzwerten unterscheiden. Sind die Ausgänge verschieden, dann wird die jeweilige Komponente als nicht robust klassifiziert. Sind hingegen die Zustände unterschiedlich wird die Komponente als gefährlich klassifiziert. Für jede Komponente in M wird dieses Verfahren durchgeführt. Der Vorgang wird mit bis zu μ zufälligen Stimuli wiederholt bzw. solange noch nicht alle Komponenten klassifiziert sind.

C. Klassifikation gefährlicher Komponenten

Alleinstehend kann die beschriebene Simulation sowohl nicht robuste als auch gefährliche Komponenten klassifizieren. Integriert in den Gesamtablauf $\mathrm{Rob}^{\mathrm{EX}}$ kann dieser zusätzlich die Anzahl der zu prüfenden gefährlichen Komponenten reduzieren. Zu einem beliebigen Zeitpunkt i, $1 \leq i \leq t^d$, sind die Komponenten in

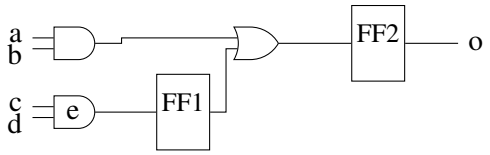

Abbildung 5. Einfacher sequentieller Schaltkreis

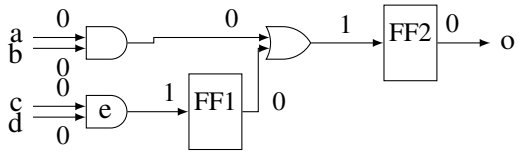

Abbildung 6. Schaltkreis aus Abbildung 5 mit Kantengewichten

M weder als robust noch als nicht robust klassifiziert worden. Alle nicht robusten Komponenten bis zum Zeittakt einschließlich i wurden bereits klassifiziert. Für den aktuellen Zeitschritt i wird nun erneut simuliert, wobei lediglich auf Veränderungen in den Zuständen geachtet werden muss. Alle dabei als gefährlich klassifizierten Komponenten müssen in der formalen Analyse nicht mehr betrachtet werden. Wurden durch die Simulation alle verbleibenden Komponenten als gefährlich klassifiziert kann gänzlich auf die formale Analyse für diesen Zeittakt verzichtet werden.

D. Unterapproximation

Die Zufallssimulation berechnet eine Unterapproximation der erreichbaren Zustände. Wenn das Beobachtungsfenster t^d identisch zur formalen Analyse konfiguriert wird, ist eine konsistente Klassifikation gegeben. Wählt man das Beobachtungsfenster der Simulation größer als das der formalen Analyse, kann die Simulation ggf. mehr Komponenten als nicht robust klassifizieren, als die formale Analyse. Dies kommt durch die längeren Propagationspfade, sodass der Fehler später sichtbar wird. Um ein konsistentes Maß bzgl. des Beobachtungsfenster t^d zu erzielen, wird die Simulation immer mit den gleichen Parametern ausgeführt wie die formale Analyse.

E. Überapproximation

Die Robustheitsprüfung mit einer Überapproximation klassifiziert nur robuste Komponenten exakt, d.h. nicht robuste Komponenten werden möglicherweise fehlerhaft klassifiziert, da nicht erreichbare Zustände mit betrachtet werden. Um diese Klassifikation bei der Überapproximation zu beschleunigen, werden bei der Zufallssimulation die Zustände zur Fehlerinjektion zufällig gewählt. Da die formale Analyse ebenfalls eine Überapproximation betrachtet, klassifiziert diese Zufallssimulation lediglich Komponenten als nicht robust, die letztendlich nicht mehr als robust bewiesen werden können. Komponenten, die durch die Zufallssimulation als nicht robust klassifiziert werden, müssen bei der formale Analyse nicht mehr betrachtet werden. Dadurch kann der Beweisprozess der robusten Komponenten deutlich beschleunigt werden.

Dazu muss der Ablauf in Abbildung 4 nur geringfügig angepasst werden. Es findet keine initiale Zufallssimulation statt. Der Injektionszustand kann ein beliebiger Zustand sein.

VI. Minimaler Propagationspfad

In Abbildung 5 ist ein sequentieller Schaltkreis mit zwei Flip Flops dargestellt. Eine Fehlerinjektion an der Komponente e bei der Betrachtung eines Zeittaktes kann aus struktureller Sicht nicht zu einer Abweichung am Ausgang o führen. Um eine Propagation strukturell zu ermöglichen, müssen in diesem Beispiel mindestens drei Zeittakte betrachtet werden. Anders ausgedrückt, führt der kürzeste Pfad der Komponente e zum Ausgang o über die beiden Flip Flops *FF1* und *FF2*. Das bedeutet, die Länge des kürzesten Pfades von e zu einem Ausgang betrachtet über alle Ausgängen gibt an, wie viele Zeittakte mindestens benötigt werden, um eine Propagation zu ermöglichen. Diese Anzahl an Zeittakten stellt demnach eine untere Schranke dar, die mindestens für eine erfolgreiche nicht robuste Klassifikation betrachtet werden müssen.

Das Berechnen der unteren Schranken kann sehr effizient durchgeführt werden. Dabei wird der Schaltkreis als gerichteter Graph mit Kantengewichten dargestellt. Alle eingehenden Kanten der Flip Flops bekommen das Kantengewicht *eins*, die restlichen Kanten erhalten eine *null*. Dieser Graph ist für den Beispielschaltkreis in Abbildung 6 dargestellt, wobei Ein- und Ausgänge des Schaltkreises zusätzlich durch Pfeile gekennzeichnet sind.

Der kürzeste Pfad einer Komponente zu den Ausgängen kann beispielsweise durch Dijkstra's Algorithmus zur Berechnung kürzester Pfade realisiert werden. Die Länge des kürzesten Pfad ist dann eine untere Schranke für eine notwendige Bedingung einer nicht robusten Klassifikation. Können die betreffenden Komponenten nicht als gefährlich klassizifiert werden sind diese robust.

Definition 2: Gegeben ist ein Schaltkreis $\mathbb{C}$ und eine Komponente g. Der *minimale Propagationspfad* $\mathrm{MP}(\mathbb{C}, g)$ ist die Länge des kürzesten Pfads zu einem Ausgang.

Lemma 1: Gegeben seien ein Schaltkreis $\mathbb{C}$ und eine Komponente g. Der minimale Propagationspfad $\mathrm{MP}(\mathbb{C}, g) + 1$ gibt an, wie viele Zeittakte mindestens betrachtet werden müssen, sodass ein Fehler an einer Komponente g an mindestens einem Ausgang sichtbar wird. ■

Das bedeutet, werden bei der aktuellen Klassifikation weniger Zeittakte betrachtet, als die Komponente mindestens benötigt, kann die diese übersprungen werden.

Algorithmus 3: $\mathrm{boundsRob}^{\mathrm{EX}}(\mathbb{C}, t^d, o)$

```
   Input : ℂ, t^d ∈ ℕ, o ∈ {↓↑, ↑↓}
   Output : Bounds of Robustness
1  begin
2      if o =↓↑ then
3          (Š^{t^d}, T̆^{t^d}, D̆^{t^d}) = Rob^EX(ℂ, t^d, S^↓, ∅, ∅);
4          (Ŝ^{t^d}, T̂^{t^d}, D̂^{t^d}) = Rob^EX(ℂ, t^d, S^↑, Š^{t^d}, ∅);
5      else
6          (Ŝ^{t^d}, T̂^{t^d}, D̂^{t^d}) = Rob^EX(ℂ, t^d, S^↑, ∅, ∅);
7          (Š^{t^d}, T̆^{t^d}, D̆^{t^d}) = Rob^EX(ℂ, t^d, S^↓, ∅, T̂^{t^d});
8      end
9      return (R_lb^{t^d}, R_ub^{t^d})
10 end
```

VII. Berechnung der Schranken mit $\mathrm{Rob}^{\mathrm{EX}}$

Wie bereits in Abschnitt II-C erwähnt, wird die Robustheitsberechnung mit einer Unterapproximation zur Berechnung der oberen Schranke und anschließend mit einer Überapproximation zur Berechnung der unteren Schranke durchgeführt (siehe Algorithmus 1).

Anhand der Mengenbeziehungen aus Abschnitt II-C zwischen nicht robusten und robusten Komponenten, können bereits klassifizierte Komponenten aus der einen Berechnung in die andere übertragen werden, sodass keine erneute Klassifikation erfolgen muss. Daraus ergeben sich zwei Reihenfolgen, in denen die Analyse durchgeführt werden kann: Erst die Unterapproximation und anschließend die Überapproximation oder umgekehrt. In Algorithmus 3 wird der neue Ablauf $\mathrm{Rob}^{\mathrm{EX}}$ zusammen mit den beiden möglichen Reihenfolgen integriert.

Neben dem Schaltkreis $\mathbb{C}$ und einem Beobachtungsfenster t^d bekommt der Algorithmus zusätzlich einen Parameter o übergeben, der die Reihenfolge festlegt, in der die Schranken berechnet werden:

- $\downarrow\uparrow$ (Zeile 2): Zuerst wird die Klassifikation auf Basis einer Unterapproximation $S^{\downarrow}$ durchgeführt. Jene Komponenten die als nicht robust klassifiziert werden, sind auch bei Betrachtung einer größeren Menge, insbesondere einer Überapproximation der erreichbaren Zustände nicht robust und können bei der Berechnung mit einer Überapproximation übernommen werden.
- $\uparrow\downarrow$: Hier wird die Klassifikation mit einer Überapproximation $S^{\uparrow}$ begonnen. Alle als robust klassifizierten Komponenten sind ebenfalls bei der Betrachtung einer kleineren Menge, insbesondere einer Unterapproximation robust. Das bedeutet, auch diese Klassifikationen können übertragen werden.

Folglich können von beiden Berechnungen bereits durchgeführte Klassifikationen übernommen werden, was die Laufzeit insgesamt verringert.

VIII. Experimentelle Ergebnisse

Die experimentellen Ergebnisse des neu vorgestellten Ablaufs $\mathrm{Rob}^{\mathrm{EX}}$ werden in diesem Abschnitt beschrieben und gegen das Verfahren aus [7] verglichen. Als formale Analyse wurde ein sequentieller Äquivalenzvergleich verwendet der durch MiniSAT [8] gelöst wurde. Dabei nutzt das Verfahren Dominatoren aus um die Klassifikation zu beschleunigen. Der neue Ablauf wurde ebenfalls mit bzw. ohne diese Optimierung evaluiert.

Verschiedene ITC'99 Schaltkreise wurden um Schutzmaßnahmen zur Tolerierung von transienten Fehlern erweitert. Folgende Techniken wurden dazu implementiert:

- Paritätsprüfung: Die Parität an den Flip Flops und den primären Ausgängen wird überprüft. Die implementierten Schaltkreise sind mit dem Suffix `-par` gekennzeichnet.
- Dreifach-Redundanz (TMR): Der gesamte Schaltkreis wurde verdreifacht und mit einer Mehrheitsentscheidung an den jeweiligen Ausgängen der Module implementiert: einmal ohne Fehlersignal, gekennzeichnet durch den Suffix `-tmr` und einmal mit Fehlersignal, gekennzeichnet durch `-tmrflt`.

Die Ergebnisse wurden auf einem AMD Opteron Dual Core Prozessor mit 3GHz Taktfrequenz und 32GB Hauptspeicher unter Linux durchgeführt. Die Laufzeit der Zufallssimulation wurde auf maximal 20 Sekunden pro Aufruf begrenzt. Das Beobachtungsfenster wurde auf $t^d = 10$ festgelegt, d.h. es werden 10 Takte nach der Fehlerinjektion (einschließlich) betrachtet.

Die Daten zu den Schaltkreisen wie, Name, Anzahl der primären Ein- und Ausgänge, der Flip Flops und der Gatter sind in Tabelle II dargestellt. Weiterhin sind die berechneten Schranken der Robustheit R_{lb} und R_{ub} aufgelistet. Die Schaltkreise mit Paritätsprüfung (`-par`) und TMR-Schaltkreise mit Fehlersignal (`-tmrflt`) sind sehr robust, d.h. die meisten Komponenten bis auf sehr wenige sind robust. Die originalen ITC'99 Schaltkreise (`-orig`) sind wie erwartet sehr unrobust. Bei den TMR Schaltkreisen ohne Fehlersignal liegen die Schranken sehr weit auseinander, was daher kommt, dass die meisten Fehler lediglich maskiert aber nicht im System korrigiert werden, sodass sehr viele Komponenten als gefährlich klassifiziert werden, d.h. nur Änderungen an den Zuständen bewirken.

In Tabelle III sind die Laufzeiten des neuen Ablaufs mit verschiedenen Reihenfolgen ($\uparrow\downarrow$) und ($\downarrow\uparrow$) mit bzw. ohne das Ausnutzen von Dominatoren aufgelistet. Weiterhin werden als Vergleich die Laufzeiten des Ablaufs aus [7], welcher auf Basis reiner formaler Analyse klassifiziert, dargestellt. Alle Berechnungen der formalen Analyse wurden mit Dominatoren durchgeführt.

Die Analyse der Schaltkreise kann insgesamt durch geeignete Parameter deutlich beschleunigt werden. Insbesondere, die Analyse der schwierigsten Schaltkreise, `b12-par` und `b12-tmr`, wird deutlich verbessert. Für die sehr einfachen Schaltkreise ergeben sich kleine Verschlechterung der Laufzeiten. Der neue Ablauf ist besonders effektiv, wenn es um die Klassifikation der TMR Schaltkreise ohne Fehlersignal geht. Das ist damit

Tabelle II
UNTERE UND OBERE SCHRANKE DER ROBUSTHEIT

Schaltkreis	PI	PO	FF	$\|\mathbb{C}\|$	R_{lb} $(S^{\uparrow})$	R_{ub} $(S^{\downarrow})$
b09-orig	1	1	28	182	0.0%	48.9%
b10-orig	11	6	17	238	0.0%	1.7%
b11-orig	7	6	31	820	0.1%	9.3%
b12-orig	5	6	121	1,171	0.0%	54.4%
b13-orig	10	10	53	383	0.8%	53.8%
b09-par	1	2	30	574	85.5%	95.3%
b10-par	11	7	24	687	84.0%	85.9%
b11-par	7	7	38	1,793	80.5%	85.3%
b12-par	5	7	128	3,763	85.2%	94.7%
b13-par	10	11	64	1,236	89.7%	95.1%
b09-tmr	1	1	84	556	0.4%	99.6%
b10-tmr	11	6	51	779	1.5%	97.8%
b11-tmr	7	6	93	2,521	0.6%	99.5%
b12-tmr	5	6	363	3,572	0.3%	99.8%
b13-tmr	10	10	159	1,249	2.3%	99.0%
b09-tmrflt	1	2	84	567	99.7%	99.7%
b10-tmrflt	11	7	51	790	97.9%	97.9%
b11-tmrflt	7	7	93	2,532	99.5%	99.5%
b12-tmrflt	5	7	363	3,583	99.7%	99.8%
b13-tmrflt	10	11	159	1,260	98.4%	99.1%

Tabelle III
VERGLEICH DER LAUFZEITEN

Schaltkreis	$(\downarrow\uparrow)$	$(\downarrow\uparrow)^a$	$(\uparrow\downarrow)$	$(\uparrow\downarrow)^a$	nur formal
b09-orig	13.62	21.83	12.59	21.04	**2.92**
b10-orig	13.05	24.31	14.29	23.71	**3.56**
b11-orig	**25.46**	40.28	36.06	45.62	59.22
b12-orig	**60.66**	77.40	77.85	85.07	208.18
b13-orig	20.51	31.75	17.38	27.6	**12.49**
b09-par	22.49	33.5	20.13	25.68	**14.81**
b10-par	35.12	37.37	34.34	**31.90**	38.34
b11-par	441.61	277.44	930.20	**221.46**	508.49
b12-par	**1818.23**	3850.7	14836.35	3503.14	10540.35
b13-par	70.1	83.25	130.67	**68.64**	93.76
b09-tmr	28.49	39.29	**24.85**	34.5	75.78
b10-tmr	2448.49	4479.51	7424.69	13378.55	**1713.26**
b11-tmr	**5506.86**	9837.44	10739.4	22943.59	9540.75
b12-tmr	**8713.03**	15860.04	16861.06	22230.5	12881.59
b13-tmr	**229.23**	556.67	660.81	802.6	793.64
b09-tmrflt	15.63	23.5	17.22	24.35	**2.60**
b10-tmrflt	19.43	25.48	22.17	27.21	**6.57**
b11-tmrflt	945.36	85.01	923.7	73.61	**66.62**
b12-tmrflt	402.04	**71.89**	378.68	315.01	170.64
b13-tmrflt	29.24	27.71	56.36	29.93	**13.09**

[a] ohne Ausnutzen von Dominatoren

zu begründen, dass Fehler an den Komponenten maskiert und nicht korrigiert werden, was sehr viele gefährliche Komponenten mit sich bringt. Insbesondere hier können Zufallssimulation und Fehlerimplikation sehr effektiv ausgenutzt werden. Für TMR-Schaltkreise mit Fehlersignal ist der neue Ablauf etwas schlechter. Dies kommt daher, dass fast alle Komponenten robust sind, was nur durch die formale Analyse bewiesen werden kann. Die Zufallssimulation und die Fehlerimplikation können hier nicht ausgenutzt werden. Insgesamt liefert der neue Ablauf mit der Reihenfolge $\downarrow\uparrow$, d.h. erst die Unterapproximation und anschließend die Überapproximation die meisten und besten Verbesserungen. Ohne die Optimierung durch Dominatoren können sogar Verbesserungen erzielt werden, wie sich bei der Reihenfolge $\uparrow\downarrow^a$ für Schaltkreise mit Paritätsprüfung zeigt. Insgesamt bringt der neue Ablauf mit der Reihenfolge $\downarrow, \uparrow$ und Dominatoren die meisten Verbesserungen.

In Tabelle IV sind genaue Details der Analyse für die beiden schwierigsten Schaltkreise `b12-par` und `b12-tmr` aufgelistet. Die Spalten geben dabei folgende Daten an: *Impl.* die Anzahl der durchgeführten Fehlerimplikationen, *MP* die Anzahl der Anwendungen des minimalen Propagationspfads, *Sim.* die durch die Simulation durchgeführten Klassifikationen und *Kl.* die gesamte Anzahl durchgeführter Klassifikationen. Die letzte Spalte gibt an, wieviel Klassifikationen durch neuen Ablauf durch die vorgestellten Techniken durchgeführt wurden. Hierbei zeigt sich, dass bei relativ geringem Verhältnis $< 10\%$ die Laufzeiten deutlich reduziert werden können.

Tabelle IV
DETAILLIERTE AUSWERTUNG FÜR B11-PAR UND B13-TMR

Schaltkreis	Impl.	MP	Sim.	Kl.	Red.
b12-par	2239	845	34	56022	5.57%
b13-tmr	134	514	1077	18271	9.44%

IX. ZUSAMMENFASSUNG

Diese Arbeit stellt einen hochoptimierten Ablauf zur Robustheitsprüfung vor. Gerade durch die Integration verschiedener Techniken konnte die Laufzeit der Robustheitsprüfung deutlich verringert werden. Die Ergebnisse zeigen deutlich, dass insbesondere die Analyse der schwierigsten Schaltkreise beschleunigt wird. Schaltkreise bei denen sich der Fehler sehr spät an den Ausgängen zeigt, können durch die Integration der neuen Techniken viel effizienter analysiert werden.

LITERATUR

[1] C. Zhao and S. Dey, "Improving transient error tolerance of digital VLSI circuits using RObustness COmpiler (ROCO)," in *Int'l Symp. on Quality Electronic Design*, 2006, pp. 133–140.

[2] R. Drechsler, S. Eggersgluss, F. Fey, A. Glowatz, F. Hapke, J. Schloeffel, and D. Tille, "On acceleration of SAT-based ATPG for industrial designs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 7, pp. 1329 –1333, 2008.

[3] M. Hunger, S. Hellebrand, A. Czutro, I. Polian, and B. Becker, "ATPG-Based grading of strong fault-secureness," in *IEEE International On-Line Testing Symposium*, 2009, pp. 269–274.

[4] J. Hayes, I. Polian, and B. Becker, "An analysis framework for transient-error tolerance," in *VLSI Test Symp.*, 2007, pp. 249–255.

[5] M. Bozzano, A. Cimatti, and F. Tapparo, "Symbolic fault tree analysis for reactive systems," ser. LNCS, vol. 4762, 2007, pp. 162–176.

[6] N. K. Jha and S. Gupta, *Testing of Digital Systems*. New York, NY, USA: Cambridge University Press, 2002.

[7] G. Fey, A. Sülflow, S. Frehse, and R. Drechsler, "Effective robustness analysis using bounded model checking techniques," *IEEE Trans. on CAD*, 2011, accepted.

[8] N. Eén and N. Sörensson, "An extensible sat-solver," in *SAT*, ser. Lecture Notes in Computer Science, E. Giunchiglia and A. Tacchella, Eds., vol. 2919. Springer, 2003, pp. 502–518.