Institut für Technische Informatik
Universität Stuttgart
Pfaffenwaldring 47
D–70569 Stuttgart

Diplomarbeit Nr. 3245

Adaptive simulationsbasierte Diagnose von Verzögerungsfehlern in kombinatorischen Schaltungen

Eric Schneider

Studiengang:	Informatik
Prüfer:	Prof. Dr. Hans-Joachim Wunderlich
Betreuer:	Dipl.-Inf. Stefan Holst
begonnen am:	15. September 2011
beendet am:	16. März 2012
CR-Klassifikation:	B.8.1, C.4, J.6

Vorwort

Strukturen mit Dimensionen von wenigen Nanometern, wie man sie in modernen Chips findet, können nur noch mit erheblichem Aufwand in komplexen Herstellungsprozessen produziert werden. Hierbei können, in Abhängigkeit von Prozess-Parametern und Design, Defekte auftreten, die das Zeitverhalten der Schaltung beeinflussen und sowohl rein zufälliger, als auch systematischer Natur sein können. Durch die stetig steigenden Taktfrequenzen häuft sich dabei die Gefahr, dass kleine Verzögerungsfehler auftreten, welche im Vergleich zu statischen Fehlern nur unter Echtzeit-Bedingungen sichtbar werden.

Um die Chipausbeute bei der Herstellung zu erhöhen und Qualitätsanforderungen zu gewährleisten, ist Diagnose deshalb von essentieller Bedeutung. Defekte müssen lokalisiert und anfällige Stellen in fehlerhaften Schaltkreisen ausfindig gemacht werden. Dadurch können das Design und der Herstellungsprozess optimiert und die Kosten pro fehlerfreien Chip bei der Entwicklung gesenkt werden.

Die genaue Diagnose der kleinen Verzögerungsfehler stellt jedoch eine große Herausforderung dar, da das Verhalten und die Simulation dieser Fehler sehr komplex sind, und diese nicht mehr effektiv mit einfacheren Fehlermodellen, wie dem Transitionsfehlermodell [WLRI87] abgedeckt werden können. Zudem erschweren Variationen innerhalb der Schaltkreise die Diagnose.

Das Ziel dieser Arbeit ist die Entwicklung eines Verfahrens zur Diagnose von kleinsten Verzögerungsfehlern, welches Defektstellen effizient lokalisieren und die Defektgrößen der Fehler abschätzen kann. Dabei soll ein simulationsbasierter Ansatz mit einem Zeitsimulator verwendet werden, um die Fehler präzise auszuwerten und stabile Ergebnisse bei Präsenz von Variationen zu ermöglichen.

Ziel

- Entwicklung eines Verfahrens zur Diagnose von einzelnen kleinsten Verzögerungsfehlern:
  - Implementierung von gängigen fehlermodellunabhängigen Methoden zur Reduzierung der initialen Kandidaten.
  - Entwicklung einer simulationsbasierten Analyse zur Identifikation der Fehlers.
- Evaluierung des vorgestellten Verfahrens und Diskussion der Resultate.

Gliederung

Dieses Dokument ist folgendermaßen gegliedert:

Kapitel 1 – Grundlagen: Dieses Kapitel dient der Erläuterung der verwendeten Definitionen und Hintergründe.

Kapitel 2 – Small Delay Fehlermodell: Dieses Kapitel gibt eine kurze Einführung in das verwendete Fehlermodell. Zusätzlich werden gängige Diagnoseverfahren vorgestellt und diskutiert, sowie Hintergründe zur Diagnose vom SMALLDELAY Fehlern erläutert.

Kapitel 3 – Adaptive simulationsbasierte Diagnose: In diesem Kapitel werden die Schritte der entwickelten Diagnosemethode vorgestellt. Diese setzt sich aus zwei Teilen – einer strukturellen und einer simulationsbasierten Analyse – zusammen. Des Weiteren wird das Bewertungsverfahren der Fehlerkandidaten erläutert.

Kapitel 4 – Ergebnisse: Hier wird der Aufbau der durchgeführten Experimente vorgestellt, sowie die Ergebnisse präsentiert und diskutiert.

Am Schluss folgt eine Zusammenfassung.

Inhaltsverzeichnis

Liste verwendeter Symbole

$\Delta^T(o)$, Δ^T_{min}, Δ^T_{max}	Slack eines Ausgangs o, Kleinster bzw. größter Slack an einem Knotens
δ_f	Defektgröße eines SMALLDELAY-Fehlers
Δ_{err}	Fehler/Abweichung der Defektgröße
δ_{est}, δ_{fin}	Erste Abschätzung der Defektgröße und Korrektur
$\delta_{min}(s)$, $\delta_{max}(s)$	Kleinste/Größte Defektgröße eines Knoten s
ω_σ, ω_ι	Gewichte für die Evidenz-Komponenten
ω_U, $\omega_U(o)$	Fehlerwahrscheinlichkeit eines Ausgangs in der Umgebung U
Π	Testmustersatz, Testsatz
π	Testmuster, Verzögerungstest
σ_f, ι_f, τ_f	Komponenten der Globalen Evidenz eines Fehlers f
$\sigma_{f,\pi}$, $\iota_{f,\pi}$, $\tau_{f,\pi}$	Komponenten der Evidenz eines Fehlers f für ein Testmuster π
A, B, Z	Eingänge und Ausgang eines Gatters
A_T	Zusätzlicher Zustandsvektor eines Signals für mehrwertige Logik
A_V	Zustandsvektor eines Signals mit booleschen Werten
D_s	Defektgrößen-Intervall eines Knoten s
$e(f,\Pi)$	Globale Evidenz
$e(f,\pi)$	Evidenz eines Testmusters π
EA, LS	Ankunfts-, Stabilisierungszeit eines Signals
f	(SMALLDELAY-)Fehler
G, $G(V,E)$	Kombinatorischer Schaltkreis, Netzliste mit Knotenmenge V und Kanten E
I	Menge an Schaltungseingängen
i, j, k	Indizes
LB, UB	Untere/Obere Grenze eines Defektintervalls
N	Anzahl der Knoten bzw. Gatter im Schaltkreis
n, g	Ein Knoten bzw. Gatter
O, o	Menge an Schaltungsausgängen, ein Ausgang
q	Quantisierungsfaktor
S, s	Initiale Kandidaten, Kandidat

T, t	Abtastzeit, Zeitpunkt
U	Umgebung/Bereich um die Abtastzeit T

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

1 Grundlagen

1.1 Erklärung grundlegender Begriffe

Netzlisten

Eine **Netzliste** ist ein gerichteter *azyklischer* Graph $G(V, E)$ über einer Knotenmenge V, sowie einer Kantenmenge E, und dient der Beschreibung kombinatorischer Schaltkreise auf Logik-Ebene. Dabei repräsentiert jeder Knoten $n \in V$ ein bestimmtes Gatter des Schaltkreises. Die einzelnen Kanten $(x, y) \in E$ modellieren jeweils eine direkte Verbindung zwischen zwei Logikgattern $x, y \in V$, bei denen der Ausgang von x mit einem Eingang von y verknüpft ist. Dabei ist x ein direkter Vorgänger im *Fan-In* von y – umgekehrt ist y ein direkter Nachfolger im *Fan-Out* von x. Die Menge der Knoten ohne direkten Vorgänger (Nachfolger) bezeichnet die Menge der primären Eingänge (Ausgänge) eines Schaltkreises und wird im Folgenden mit I (O) notiert.

Um mit Hilfe von Netzlisten den Wert eines beliebigen Gatters $n \in V$ im Schaltkreis zu bestimmen, wendet man dessen assoziierte Funktion $\varphi(x_0, x_1, ...x_{k-1})$ auf die Werte der direkten Vorgänger an. Aufgrund der Datenabhängigkeiten setzt dies voraus, dass alle Vorgänger zuvor berechnet wurden und deren Werte bekannt sind. Eine Möglichkeit dies zu gewährleisten, ist die Evaluierung der einzelnen Knoten in V in topologisch sortierter Reihenfolge von Schaltungseingängen zu Ausgängen, indem V in sogenannte *Ebenen* (Level) $L_0, L_1, ...L_{k-1}$ partitioniert wird, sodass für alle Knoten $x \in L_i, y \in L_j$ gilt: $(x, y) \in E \Rightarrow i < j$.

Defekte, Fehler und Fehlermodelle

Als **Defekt** bezeichnet man eine Störung bzw. Veränderung in der physischen Struktur des Schaltkreises, die z.B. durch Verunreinigungen im verwendeten Material, durch auftretende Variationen im Herstellungsprozess oder durch einen fehlerhaften Entwurf entstanden sind. Solche Defekte können die Schaltung derart beeinflussen, dass das Verhalten von der Spezifikation abweicht. Um dieses Fehlverhalten aufgrund von Defekten beschreiben zu können, wird im Allgemeinen der Begriff „*Fehler*" verwendet. Ein **Fehler** dient der logischen Abstraktion und Einschränkung des Defektausmaßes auf einer höheren Ebene und repräsentiert für gewöhnlich eine ganze Klasse unterschiedlicher physischer Defekte, die aber allesamt ein und das selbe Fehlverhalten verursachen.

Fehler werden nach einem bestimmten **Fehlermodell** spezifiziert, welches die Zahl der möglichen Defektszenarien auf eine endliche Menge beschränkt. So modelliert das bekannte *Haftfehlermodell* (Stuck-At) für jede *Fehlerstelle* (wie z.B. Gatter oder Leitungen) einen *StuckAt-0* und einen *StuckAt-1* Fehler, sowie den fehlerfreien Zustand, während man beim *Brückenfehlermodell* für eine *wired-AND-* oder *wired-OR*-Brücke an einer Stelle verschiedener Kombinationen für ein *Aggressor*-Signal zur Wahl haben kann. [BA02] bietet eine Übersicht von unterschiedlichen Fehlermodellen für verschiedene Abstraktionsebenen.

Darüber hinaus unterscheidet man zwischen *Einzel-* und *Mehrfachfehlermodellen*. In letzterem wird die Möglichkeit des gleichzeitigen Auftretens verschiedener Fehler berücksichtigt, was die Komplexität des Modells letztlich aufgrund der steigenden Zahl der Fehler-Kombinationen erhöht. Die Wahl des Fehlermodells bestimmt somit sowohl die Präzision, aber auch die Komplexität der modellierten Defekte.

Fehlersimulation

Mit Hilfe von expliziter **Fehlersimulation** kann das Verhalten von Schaltkreisen unter Einfluss von Defekten gezielt getestet und beobachtet werden. Dadurch gewinnt man die Erkenntnis, ob ein Defekt in der Schaltung für bestimmte Eingabestimuli (Testmuster) einen Fehler am Ausgang produziert und welches Ausmaß dieser Fehler hat. Dieser Mechanismus kann dann für eine Reihe verschiedener Anwendungen verwendet werden [WWW06], wie der Bestimmung der *Fehlerabdeckung* von automatisch generierten **ATPG (Automatic Test Pattern Generator)** Testmustersätzen, oder im Rahmen einer *Fehlerdiagnose* z.B. durch die Generierung eines *Fehlerwörterbuchs*.

Abb. 1.1 zeigt die schematische Darstellung einer Fehlersimulation. Hier wird ein Testmuster an die primären Eingänge von zwei Instanzen eines Schaltkreises angelegt. Der untere Schaltkreis

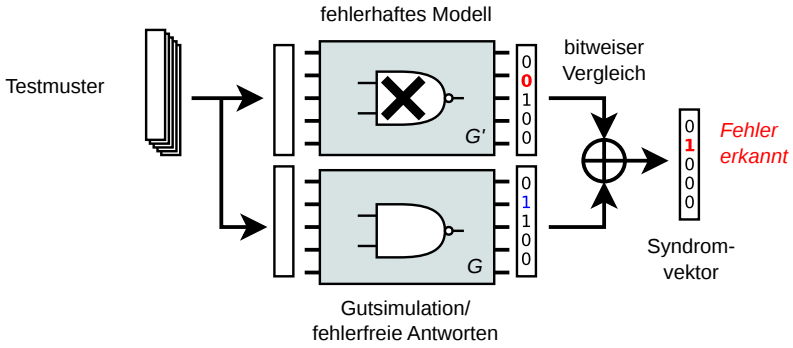


Abbildung 1.1: Schema eines Fehlersimulation. – Die Antworten der fehlerhaften Schaltung G' werden mit denen der Spezifikation G verglichen.

G repräsentiert eine Gutsimulation, die Referenzantworten gemäß der Spezifikation liefert. Schaltkreis G' wurde dagegen so modifiziert, dass das beabsichtigte Verhalten dem Einfluss eines bestimmten Fehlers entspricht. Diese Modifikation wird **Fehlerinjektion** genannt. Im Anschluss an die Simulation werden die Antwortmuster beider Schaltkreise von den primären Ausgängen abgelesen und miteinander verglichen. Durch eine bitweise XOR-Verknüpfung ($\oplus$) der Antworten erhält man ein **Symptom**, welches Aufschluss darüber gibt, ob ein Ausgang fehlerhaft ist oder nicht. Die vereinigten Symptome eines ganzen Testmustersatzes bilden das **Syndrom** des Fehlers.

Damit ein Fehler von einem Testmuster sichtbar gemacht werden kann, müssen die folgenden Bedingungen erfüllt sein:

- Der Fehler muss über die primären Eingänge **aktiviert** werden (*fault exitation*).
- Der Fehler muss zu den primären Ausgängen **propagiert** werden (*fault propagation*).

Manche Fehler haben solch komplexe Aktivierungs- und Propagierungsbedingungen, sodass diese bei Verwendung von pseudozufallsgenerierten Testmustern nur mit einer sehr geringen Wahrscheinlichkeit sichtbar werden. Hierfür müssen ATPG-Tools zur Testmustergenerierung verwendet werden, um die Detektierbarkeit der Fehler und somit die Fehlerabdeckung der Testmustersätze zu verbessern.

1.2 Verzögerungsfehler und Modelle

Im Gegensatz zu den Defekten aus *statischen* Fehlermodellen, wie dem Haftfehlermodell, beeinflussen Verzögerungsdefekte nicht die logische Struktur der Schaltung, sondern das **Zeitverhalten** der einzelnen Gatter und Leitungen. Dabei können Signalwechsel (Transitionen) an den defekten Stellen verlangsamt werden, wodurch es vorkommen kann, dass bestimmte Transitionen im Schaltkreis die *Zeitanforderungen* der Schaltung überschreiten, die durch die geforderten Taktfrequenzen der Schaltung bestimmt sind. In der Regel verletzt ein Signal die Zeitanforderungen, wenn die Differenz von *Abtastzeit* und *Stabilisierungszeit* am Ausgang negativ ist. Diese Differenz wird auch **Slack** genannt. Zum Zeitpunkt der Abtastung können die Signalwechsel an den Ausgängen nicht mehr registriert werden, was typischerweise einen Fehler zur Folge hat.

Um Verzögerungsfehler testen zu können, muss der Fehler durch eine Signaltransition an der Fehlerstelle aktiviert werden. Zur Erzeugung dieser Transitionen, benötigt man Stimuli in Form eines *Verzögerungstests*, der aus einer Sequenz von zwei Testmustern, einem **Initialisierungsvektor (IV)** und einem **Propagierungsvektor (PV)**, besteht. Der IV initialisiert dabei die Eingänge und die restlichen Knoten der Schaltung mit einem Signalwert. Durch den PV finden dann Signalwechsel an bestimmten Eingängen statt, die als Transitionen durch den Schaltkreis propagiert werden.

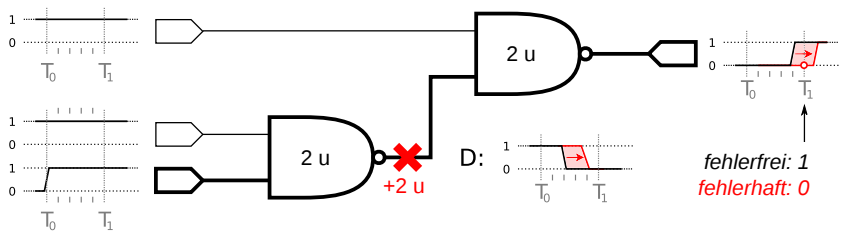


Abbildung 1.2: Beispiel eines Verzögerungstests. – Aktivierung und Propagierung eines Verzögerungsfehlers an der Leitung D mit $IV = (110)$ und $PV = (111)$. Verzögerungen sind in *Einheiten* u angegeben.

Fehler-Modelle

In der Literatur zu Test und Simulation taucht eine an Vielfalt möglichen Verzögerungsfehler-Modellen auf [KC98, MA98]. Zu den bekanntesten zählen dabei das *Transitionsfehlermodell* [WLRI87], das *Gatterverzögerungsmodell* [CIR87] und das *Pfadverzögerungsmodell* [Smi85]. Im Folgenden werden diese kurz vorgestellt:

Transitionsfehler Transitionsfehler modellieren für einen Test eine *unendliche* lokale Verzögerung an einer Leitung oder einem Gatter. Dabei wird zwischen Fehlern bei fallender (*Slow-to-Fall (STF)*) und steigender Flanke (*Slow-to-Rise (STR)*) unterschieden. In [WLRI87] wurde gezeigt, dass diese Transitionsfehler sehr effizient mit Hilfe von *temporären* Haftfehlern simuliert werden können und bei der Berechnung dabei keine Zeitinformation benötigen. So zeigt etwa ein aktivierter *STR*-Fehler an einem Gatter ein identisches Verhalten, wie ein *StuckAt-0* Fehler aus dem Haftfehlermodell. Dasselbe gilt auch für *STF* und *StuckAt-1* Fehler.

Gatterverzögerungsfehler Ein Gatterverzögerungsfehler nach [CIR87] liegt vor, wenn ein lokaler Defekt die Signalpropagierungszeiten eines Gatters – für entweder steigende oder fallende Flanken – zusätzlich um eine Defektgröße δ_f erhöht und der Fehler an mindestens einem sensibilisierten Ausgang im Ausgangskegel sichtbar wird. Im Gegensatz zum Transitionsfehlermodell ([WLRI87]), wird hier das Zeitverhalten einzelner Gatter mit berücksichtigt und durch Intervalle abgeschätzt. [IRW90] definiert mit Hilfe der Slacks an den einzelnen Knoten für jede Fehlerstelle eine Mindestdefektgröße, um die Testbarkeit zu gewährleisten.

Pfadverzögerungsfehler In diesem Modell von [Smi85] wird angenommen, dass sich ein Verzögerungsdefekt im Schaltkreis global entlang der Leitungen eines kompletten Pfades von primär Eingang nach primär Ausgang aufteilt. Der Pfadverzögerungsfehler (PDF) tritt dann auf, sobald eine Transition entlang dieses Pfades vom Eingang zum Ausgang propagiert. Dabei wird anhand der Polarität der Transition am Eingang zwischen steigenden und fallenden PDFs unterschieden.

1.3 Logische Diagnose kombinatorischer Schaltungen

Bei der logischen Diagnose eines kombinatorischen Schaltkreises versucht man anhand der Testantworten eines fehlerhaften Chips, auf die Defektursache im Schaltkreis zu schließen. Während der Chipentwicklung ist dies von besonderem Interesse, da Fehler, die in verschiedenen Chips aufgrund systematischer Defekte häufiger auftreten, auf fehleranfällige Stellen in der Schaltung hindeuten können. Mit der logischen Diagnose versucht man deshalb diese fehleranfälligen Stellen im Schaltkreis zu **identifizieren**, um Maßnahmen zur Anpassung des Designs bzw. des Layouts einzuleiten und zukünftige Chips gegen die Fehler abzuhärten.

Die logische Diagnose kombinatorischer Schaltungen hat sich hauptsächlich in zwei unterschiedliche Paradigmen aufgespalten: Dies sind die sogenannten *Cause-Effect-* und *Effect-Cause-*Verfahren [WWW06].

1.3.1 Cause-Effect Verfahren

Bei **Cause-Effect** Verfahren versucht man anhand der Antworten simulierter Fehler (*Causes*) aus einem festgelegten Modell, das Syndrom (*Effect*) des zu diagnostizierenden Chips, dem sogenannten **DUD (Device under Diagnosis)**, zu untersuchen. Hierfür werden *typischerweise* sogenannte **Fehlerwörterbücher** verwendet, in denen die Testantworten aller modellierten Fehler zum Vergleich abgespeichert sind. Ein Nachteil dieses Verfahrens ist die Abhängigkeit der Größe des Wörterbuchs von der Zahl der modellierten Fehler, der Anzahl der Ausgänge im Schaltkreis, sowie der untersuchten Testmuster. Zudem können nur solche Fehler genau diagnostiziert werden, die vom gewählten Modell abgedeckt werden und somit vom Wörterbuch erfasst sind [WWW06]. Abweichungen, die im Fehlermodell nur implizit oder garnicht abgedeckt sind, können für die Diagnose irreführend sein und sogar soweit führen, dass diese nicht diagnostiziert werden können.

1.3.2 Effect-Cause Verfahren

Effect-Cause Verfahren verwenden einen umgekehrten Ansatz. Hierbei geht man der Fehlerursache des DUDs auf den Grund, indem das Syndrom anhand der Informationen über den Schaltkreis genauer analysiert und die mögliche Defektursache auf eine Menge an Fehlerkandidaten eingeschränkt wird. Ein Vorteil des Effect-Cause Ansatzes ist die allgemeine Unabhängigkeit von einem Fehlermodell, da keine expliziten Annahmen über Fehler gemacht werden [WWW06].

Ein bekanntes Verfahren ist das **Strukturelle Pruning** [WL89], auch *Backconing* genannt. Hierbei werden die möglichen Fehlerkandidaten auf den Schnitt der Eingangskegel aller fehlerhaften Schaltungsausgänge logisch eingegrenzt.

Das **SLAT Verfahren (Single Location At a Time)** aus [BHHS01, Hui04] verfolgt einen sogenannten „*Inject & Evaluate*" Ansatz, bei dem Fehler unterschiedlichster Art mit Hilfe von einfacher Haftfehlersimulation lokalisiert werden. Hierbei werden für jeden fehlerhaften Test Haftfehler an Kandidatenstellen simuliert, um Testmuster mit einer sogenannten *SLAT-Eigenschaft* zu identifizieren. Bei einem Testmuster mit SLAT-Eigenschaft existiert ein Haftfehler, der das Syndrom durch einen *lokalen* Defekt für diesen Test erklären kann. Findet sich dabei keine Stelle, die alle Syndrome der fehlerhaften Tests durch SLAT-Mustern erklären kann, so wird die kleinstmögliche Menge an Fehlerstellen gesucht (*Multiplets*), die diese abdeckt.

1.4 POINTER Diagnose

Eine Diagnosemethode zur Analyse von Testantworten ist das **POINTER (Partially Overlapping Impact couNTER)** Verfahren aus [HW09], welches eine Erweiterung des SLAT-Verfahrens von [BHHS01, Hui04] bildet und sich zur *Effect-Cause* Diagnose von beliebigen Fehlern aus dem **CLF-Kalkül (Conditional Line Flip)** [Wun09] bewährt hat. Im CLF-Modell wird das Fehlverhalten eines Signals Z allgemein durch eine Bedingung mit beliebiger Komplexität modelliert, sodass $Z_f = Z \oplus (\textit{Bedingung})$.

Zur Diagnose dieser Fehler verwendet das POINTER-Verfahren dabei Haftfehler, die in einer **Fehlermaschine (FM)** simuliert werden, um Referenzantworten zu generieren. Beim Vergleichen der Antworten mit dem DUD werden dabei für *jeden* Fehler und jedes untersuchte Testmuster sogenannte *Evidenzen* berechnet, die zur Bewertung der möglichen Fehlerkandidaten und zur Identifizierung der Defektstelle dienen. Die **Evidenz** eines Fehlers f für ein Testmuster π ist als Tupel $e(f,\pi) = (\sigma_{f,\pi}, \iota_{f,\pi}, \tau_{f,\pi}, \gamma_{f,\pi})$ definiert, dessen Komponenten durch bitweises Vergleichen des Antwortmusters aus FM und DUD extrahiert werden:

- $\sigma_{f,\pi}$ ist die Anzahl der vom Referenzfehler f *erklärten* fehlerhaften Ausgänge des DUDs.
- $\iota_{f,\pi}$ ist die Anzahl der von f falsch *vorhergesagten* fehlerhaften Antwortbits.
- $\tau_{f,\pi}$ ist die Anzahl der fehlerhaften Ausgänge im DUD, die vom Referenzfehler *nicht* erklärt werden können.
- $\gamma_{f,\pi}$ $(= \min(\{\sigma_{f,\pi}, \iota_{f,\pi}\}))$ ermöglicht eventuell Annahmen, ob ein Symptom von einem einzelnen oder von mehreren Haftfehlern erzeugt wurde.

Für einen ganzen Testmustersatz $\Pi = \{\pi_0, \pi_1, ..., \pi_{k-1}\}$ mit k Tests, werden die Evidenzen $e(f,\pi)$ aller Antwortmuster $\pi \in \Pi$ eines Referenzfehlers f komponentenweise aufsummiert, um die *globale* Evidenz $e(f,\Pi)$ des Fehlers zu berechnen:

$$e(f,\Pi) = (\sigma_f, \iota_f, \tau_f, \gamma_f) = (\textstyle\sum_\Pi \sigma_{f,\pi}, \sum_\Pi \iota_{f,\pi}, \sum_\Pi \tau_{f,\pi}, \sum_\Pi \gamma_{f,\pi}).$$

Nach Berechnung der globalen Evidenzen *aller* Fehlerkandidaten, werden diese mit Hilfe der folgenden **GSI (Gamma-Sigma-Iota)** Bewertungsfunktion sortiert:

1. $\gamma_{f_1} > \gamma_{f_2} \Rightarrow \text{Rang}(f_1) > \text{Rang}(f_2)$,
2. $\sigma_{f_1} > \sigma_{f_2} \Rightarrow \text{Rang}(f_1) < \text{Rang}(f_2)$, wenn $\gamma_{f_1} = \gamma_{f_2}$,
3. $\iota_{f_1} > \iota_{f_2} \Rightarrow \text{Rang}(f_1) > \text{Rang}(f_2)$, wenn $\gamma_{f_1} = \gamma_{f_2}$ und $\sigma_{f_1} = \sigma_{f_2}$.

Dabei wird zunächst nach aufsteigendem γ_f sortiert, um jene Kandidaten an die Spitze der Liste zu bringen, die als einzelne *bedingte* Haftfehler das Syndrom des DUDs erklären können. Anschließend werden die Kandidaten mit gleichem γ_f in absteigender Reihenfolge nach der Anzahl der erklärten Bits σ_f sortiert. In einem letzten Schritt werden gleiche Kandidaten nach aufsteigenden ι_f geordnet, um die Anzahl der Fehlvorhersagen zu minimieren.

Je niedriger der Rang eines Kandidaten ist, umso besser kann dieser das Syndrom des DUDs erklären und desto wahrscheinlicher verweist der Fehler auf die eigentliche Defektstelle.

1.5 *Wave* Zeitsimulator

In dieser Arbeit wird ein dynamischer Zeitsimulator namens „*Wave*" verwendet, welcher von Stefan Holst entwickelt wurde, um die Schaltaktivität in Schaltkreisen zur Abschätzung der Leistungsaufnahme messen zu können.

Da die Simulation von Schaltkreisen stark parallelisierbar ist und dynamische Zeitsimulation im Vergleich zur Booleschen Logiksimulation einen hohen Rechenaufwand hat, macht dieser dabei Gebrauch von der NVIDIA®CUDA™ Architektur [NVI12], um den Simulationsprozess mit Hilfe von Parallelisierung auf CUDA™-fähigen Grafikkarten zu beschleunigen. Durch das *Many-Threading* Paradigma kann der Simulator vorhandenen Struktur- und Daten-Parallelismus bei der Simulation ausbeuten. Dabei kann dieser eine große Anzahl an Gattern für mehrere Testmuster auf verschiedenen Rechenkernen der **GPGPU (General Purpose Graphics Processing Unit)** parallel verarbeiten und evaluieren

Der *Wave*-Simulationskern erlaubt dabei die präzise Modellierung und effiziente Evaluierung *vollständiger* Signalverläufe an Gattern durch sogenannte *Waveforms*. Ein **Waveform** beschreibt den Verlauf eines Signals anhand einer Liste von absteigend sortierten Zeitpunkten $t_i \in \mathbb{R}_0^+$, an denen jeweils Signalwechsel stattfinden. Nach Definition muss das durch das Waveform beschriebene Signal für $t \to \infty$ einen Wert von '0' haben, weshalb gegebenenfalls eine *künstliche* Transition (∞) hinzugefügt werden muss. Durch die umgekehrte Reihenfolge bei der Auswertung der Transitionen können die Waveforms bezüglich der Speicherallokation effizienter berechnet werden.

In Abbildung 1.3 sind Beispiele von Signalverläufen mit der dazugehörigen Waveform-Repräsentation illustriert.

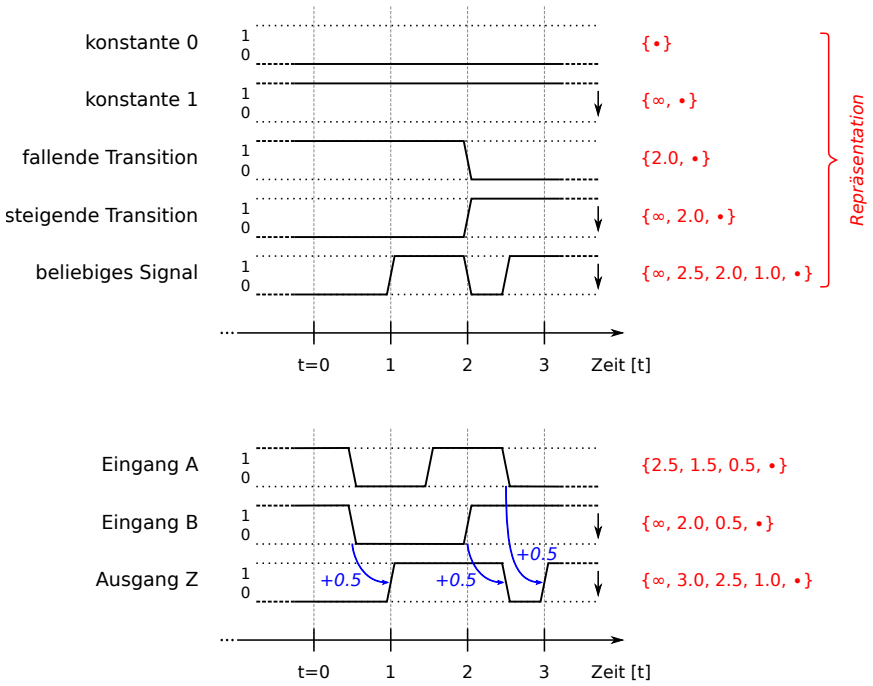


Abbildung 1.3: Beispiel der Waveform-Darstellung. – Das Symbol • wird als Terminalzeichen verwendet, während mit ∞ die Signalverläufe für $t \to \infty$ *künstlich* auf '0' gesetzt werden. In der unteren Grafik wird ein NAND-Gatter mit einer Verzögerung von $\delta = 0.5$ evaluiert.

Der *Wave*-Simulator verwendet eine GPGPU-gestützte Testmusterkonvertierung und ein schlankes Interface, um den Kommunikationsaufwand zwischen Host-CPU und GPGPU minimal zu halten und die Simulationszeit zu reduzieren. Bei jeder Auswertung können dabei unterschiedliche nominal Schaltzeiten und normalverteilte Variation der Gatterverzögerungen für steigende bzw. fallende Flanken berücksichtigt werden. Eine Puls-Filterung unterdrückt zusätzlich kurz aufeinanderfolgende Signalwechsel.

2 SmallDelay Fehlermodell

Dieses Kapitel stellt die Ergebnisse einer vorangegangenen Arbeit [Sch11] vor, auf die in dieser Arbeit zurückgegriffen wird. In [Sch11] wurde ein Modell für Verzögerungsfehler in Schaltkreisen entwickelt und der GPGPU-beschleunigte *Wave*-Zeitsimulator zur Simulation dieser Fehler erweitert. Des Weiteren wurde das POINTER Diagnoseverfahren für Verzögerungsfehler evaluiert und eine Methode vorgeschlagen, um die Diagnose unter Verwendung des Fehlersimulators zu verbessern.

Im Folgenden wird das implementierte Verzögerungsfehlermodell vorgestellt, welches in dieser Arbeit verwendet wird und der Ansatz zur Diagnose vorgestellt.

2.1 Hintergrund

Das in dieser Arbeit verwendete Fehlermodell ist eine verallgemeinerte Variante des Gatterverzögerungsmodells, um kleinste Verzögerungsfehler (sogenannte SmallDelays) zu repräsentieren [Sch11]. Hierbei handelt es sich um ein Einzelfehlermodell, in dem jeder Fehler f die Verzögerungszeit des assoziierten Gatters um eine Defektgröße δ_f erhöht, sodass die Zeitbedingungen eines oder mehrerer Pfade des Schaltkreises **verletzt** werden. Die berechneten Defektgrößen werden hierbei anhand der nominal Gatterschaltzeiten im Schaltkreis bestimmt und über die *Slacks* der verschiedenen Pfade durch einen Knoten berechnet (siehe Abb. 2.1).

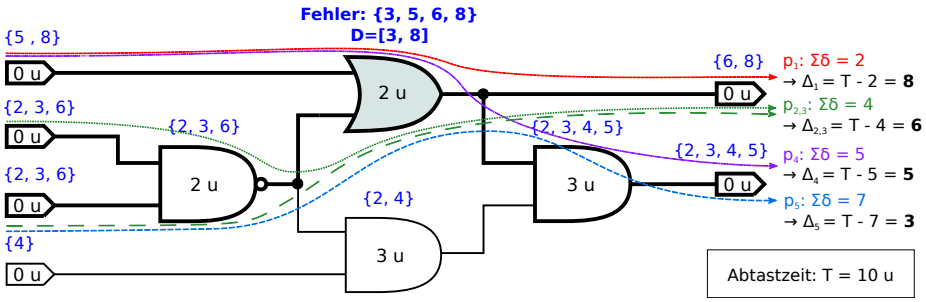


Abbildung 2.1: Herleitung einzelner Defektgrößen δ_f für einen Knoten im SmallDelay-Fehlermodell. – Die Slacks Δ_{p_i} eines jeden Pfades p_i durch den Knoten dienen als mögliche Fehlergrößen.

Der **Slack** Δ_p eines Pfades p ist definiert als die Differenz von Abtastzeit und der kumulativen Verzögerung δ_p der einzelnen Gatter von p. Ist $\Delta_p > 0$, so sind die Zeitbedingungen des Pfades erfüllt[1]. Ein SMALLDELAY-Fehler der Größe δ_f eines Knoten n verletzt die Zeitbedingungen aller Pfade p durch diesen, die eine kumulative Verzögerung von $T - \delta_f$ oder mehr besitzen, da $\Delta'_p = \Delta_p - \delta_f = (T - \delta_p) - \delta_f \leq T - T + \delta_f - \delta_f = 0$. Dadurch hat jeder aus dem Modell hergeleitete Fehler an einem Knoten ein **einzigartiges** Fehlverhalten [Sch11]. Hierbei ist noch zu erwähnen, dass eine Verletzung der Zeitbedingungen entlang eines *sensibilisierten* Pfades nicht zwingend einen Fehler am Ausgang verursachen muss. Durch nicht robuste Signalpropagierung können sogenannte **Hazards** entstehen, die ein Signal über einen *begrenzten* Zeitraum invertieren. Dabei kann es passieren, dass Ausgangssignale, die von einem Hazard betroffen sind, mit einem *guten* Wert abgetastet werden und der Fehler nicht erkannt wird.

Bei einem SMALLDELAY-Fehler wird sich zudem nicht auf eine Polarität der Transitionen beschränkt, wodurch der Fehler einen physischen Defekt modelliert, der sowohl steigende als auch fallende Flanken durch die Defektstelle beeinflusst.

Da die Anzahl der Pfade in einem Schaltkreis im schlimmsten Fall exponentiell mit der Anzahl der Gatter steigen kann [WWW06] und die Initialisierung der vollständigen Fehlermenge aufgrund der hohen Fehlerzahl (siehe Tabelle 4.1 in Kapitel 4) sehr zeitaufwändig ist, wurde in [Sch11] eine Methode vorgeschlagen, um durch **Quantisierung** sogenannter *Defektgrößen-Intervalle* verschiedene Fehlergrößen zu extrahieren. Ein **Defektgrößen-Intervall** D_n beschränkt hierbei die *relevanten* Defektgrößen eines Knotens n mit Hilfe des kleinsten detektierbaren Defekts δ_{min} und einer Transitionsfehlergröße δ_{max}, welche durch den längsten und kürzesten Pfad durch diesen bestimmt sind. Als Transitionsfehlergröße eines Knotens wird hierbei die kleinstmögliche Defektgröße verwendet, die benötigt wird, um die Zeitbedingungen *aller* Pfade durch den Knoten zu verletzen.

2.2 Diagnose von Verzögerungsfehlern

2.2.1 Critical Path Tracing für Verzögerungsfehler

Ein häufig verwendetes Effect-Cause Verfahren bei der Diagnose kombinatorischer Schaltungen ist das **Critical Path Tracing (CPT)** von [AMM83], welches ursprünglich zur impliziten Fehlersimulation vorgeschlagen wurde und in [GLP92, GLP95] für die Diagnose von Verzögerungsfehlern erweitert worden ist. Im CPT werden für jeden Test die Signale fehlerhafter Ausgänge zu den Eingängen zurückverfolgt, um so die möglichen Fehlerkandidaten in der Schaltung *räumlich* einzuschränken. Dabei wird Anfangs eine Kandidatenliste mit allen Fehlerstellen des Schaltkreises initialisiert. Anschließend wird diese folgendermaßen reduziert:

[1] Signaltransitionen *zum Zeitpunkt der Abtastung* werden hier als eine Verletzung der Zeitbedingungen betrachtet.

$\overline{\wedge}$	**C0**	**C1**	**T0**	**T1**	**H0**	**H1**
C0	C1	C1	C1	C1	C1	C1
C1	C1	C0	T1	T0	H1	H0
T0	C1	T1	T1	H1	H1	T1
T1	C1	T0	H1	T0	H1	T0
H0	C1	H1	H1	H1	H1	H1
H1	C1	H0	T1	T0	H1	H0

Tabelle 2.1: Mehrwertige Logik. – Wahrheitstabelle eines NAND-Gatters mit zwei Eingängen für sechswertige Logik.

Für einen fehlerhaften Test wird eine Gutsimulation in **mehrwertiger Logik** durchgeführt, um die Signalpropagierung und die möglichen Transienten für den Test im Schaltkreis sichtbar zu machen. Häufig wird dabei auf eine sechswertige Logik nach [Hay86] zurückgegriffen, die sich zur Analyse von Transienten eignet.

Die Logik verwendet dabei die folgenden Symbole zur Repräsentation von Signalzuständen:

- C0: repräsentiert ein Signal, das über die Zeit konstant auf ’0’ bleibt. (*stabile* ’0’)
- C1: repräsentiert ein Signal, das über die Zeit konstant auf ’1’ bleibt. (*stabile* ’1’)
- T0: repräsentiert ein Signal, das ausgehend von einem initialen Wert ’1’ nach Stabilisierung den Wert ’0’ annimmt. (*steigende Transition*)
- T1: repräsentiert ein Signal, das ausgehend von einem initialen Wert ’0’ nach Stabilisierung den Wert ’1’ annimmt. (*fallende Transition*)
- H0: repräsentiert ein Signal, das ausgehend von einem initialen Wert ’0’ mögliche Transitionen haben kann, aber nach Stabilisierung wieder ’0’ ist. (*statischer 0-Hazard*)
- H1: repräsentiert ein Signal, das ausgehend von einem initialen Wert ’1’ mögliche Transitionen haben kann, aber nach Stabilisierung wieder ’1’ ist. (*statischer 1-Hazard*)

In Tabelle 2.1 ist die Wahrheitstabelle des NAND-Gatters aus der $\mathbb{H}_6$-Logik illustriert.

Nach der Gutsimulation werden die Signalleitungen und Gatter im Eingangskegel eines jeden fehlerhaften Ausgang im DUD anhand der Netzliste und den Transienten an den Knoten zurückverfolgt. Dabei werden an Gattern nur die Eingänge weiterverfolgt, die *Ereignisse* (z.B. Transitionen oder Hazards) bzw. Signalwechsel für das aktuelle Testmuster aufweisen [GLP92], da Leitungen und Gatter ohne Signalwechsel für den Test keine Verzögerungsfehler propagieren können und diese dadurch nicht zur Erzeugung des Syndroms beitragen. Alle Leitungen und Gatter, die von dem fehlerhaften Ausgang zurückverfolgt werden können, werden dann mit der Liste der verbliebenen Kandidaten *geschnitten*.

Diese Prozedur wird für jeden der zu untersuchenden Tests ausgeführt, um so die Kandidaten stetig reduzieren zu können. Mit der Einzelfehlerannahme befindet sich der gesuchte Kandidat dabei immer unter den verbleibenden Kandidaten, da dieser bei jedem fehlerhaften Test von allen fehlerhaften Ausgängen zurückverfolgt werden können muss. Erweiterungen und Optimierungen für die Rückverfolgung sehen vor, das Backtracing bei *kontrollierten* Gattern auf Gattereingänge mit *kontrollierenden* Ereignissen zu beschränken [GLP95]. Kontrollierende bzw. **dominante Ereignisse** sind Signalverläufe mit kontrollierendem *Finalwert*. Ein Gatter wird als **kontrolliert** bezeichnet, wenn ein oder mehrere Eingänge kontrollierende Ereignisse haben, d.h. sodass Signalwechsel an nicht kontrollierenden Eingängen zu keiner Änderung des Finalwerts am Gatterausgang führen. Die Idee ist hierbei, dass die Fehlerpropagierung von Eingängen mit nicht-dominierenden Ereignissen durch dominierende fehlerfreie Ereignisse maskiert wird, da letztere das Gatter in einen kontrollierten und fehlerfreien Zustand bringen.

2.2.2 Weitere Verfahren

Im Folgenden werden weitere Methoden zur Diagnose von Verzögerungsfehlern vorgestellt:

In [TBT98] werden zunächst fehlerhafte Ausgänge im Schaltkreis mit einer Backtracing-Prozedur zurückverfolgt, um eine Menge initialer Fehlerkandidaten zu berechnen. Die Kandidaten werden anschließend mit Hilfe expliziter Simulation von Gatterverzögerungsfehlern in mehrwertiger Logik reduziert. Dabei werden Fehler ausgeschlossen, wenn die Antworten nicht mit dem DUD übereinstimmen. Zusätzlich wird versucht über die Simulation von fehlerfreien Tests weitere verbleibende Kandidaten zu eliminieren.

[WHH02] verwendet sogenannte *Symbolische Fehlerpropagierung* mit mehrwertiger Logik zur Diagnose von Gatterverzögerungsfehlern. Hierbei wird die mögliche Propagierung eines Fehlers ausgehend von einer Fehlerstelle für alle fehlerhaften Testmuster des DUDs beobachtet. Eine Bewertungsfunktion wird dann dazu verwendet, um die Übereinstimmung mit dem DUD zu messen.

Das Verfahren aus [ATH+07] wird zur Diagnose kleinster Verzögerungsfehler vorgeschlagen und verwendet neben mehrwertiger Logiksimulation mit Backtracing zusätzlich die Zeit der letzten Transition an einem jeden Knoten. Hierbei werden Fehler mit der kleinsten detektierbaren Defektgröße an den einzelnen Kandidatenstellen für *fehlerhafte* und *fehlerfreie* Tests beobachtet und bewertet.

In [MMSTR08] wird eine initiale Kandidatenmenge durch Backtracing der fehlerhaften Ausgänge berechnet und die möglichen Defektgrößen der Knoten mit Hilfe einer *Zeitsimulation* durch eine Unter- und Oberschranke eingegrenzt. Die Information über die Slacks an *fehlerhaften* und *fehlerfreien* Ausgängen wird dann dazu verwendet die Kandidatenmenge systematisch zu reduzieren.

Die Verfahren aus [RBG+07, BGPV10] verwenden mehrwehrtige Logiksimulation und Critical Path Tracing, um die möglichen Fehlerstellen eines DUDs einzuschränken. In einem zweiten Schritt wird für jede Kandidatenstelle eine Menge an möglichen Fehlertypen bestimmt, die anhand der Symbole der Logiksimulation berechnet werden.

[PB07] verwendet sogenannte *Segment Netzwerk Fehler (SNF)* zur Darstellung von Verzögerungsfehlern. Bei der Diagnose werden nach einem konservativen Backconing alle verbleibende Kandidatenstellen mit einer ‚X'-Propagierung simuliert, um die Fehlerpropagierung ausgehend von den Stellen zu beobachten und eine Fehlersignatur zu bestimmen. Die Signaturen dieser Fehler werden dann zu einem *Fehler-Cluster-Graphen* zusammengesetzt und analysiert, um *passende* SNFs zu extrahieren.

In [YB08, YB10] wurde ein Verfahren zur Diagnose von Mehrfachfehlern vorgestellt. Hierbei wird nach einem Backtracing an den verbliebenen Kandidaten ‚X'-Markierungen injiziert, die zu den Ausgängen propagiert werden. Anschließend werden durch eine konservative *Rückwärtsimplikation* ausgehend von fehlerfreien Ausgängen weitere Fehlerkandidaten ausgeschlossen.

Ein statistischer Ansatz zur Diagnose von Verzögerungsfehlern wurde in [KWCL03] präsentiert. Hierbei werden *statistische statische Zeit-Analyse (SSTA)* und *dynamische Zeitsimulation* verwendet, um über die Fehlerwahrscheinlichkeiten der Ausgänge die statistische Übereinstimmung von Kandidaten zu berechnen und zu vergleichen.

Die meisten der vorgestellten Methoden verwenden ein mehrstufiges Diagnoseverfahren und nutzen dabei das Backtracing von fehlerhafter Ausgänge, um die initiale Kandidatenmenge einzuschränken. Insbesondere bei den simulationsbasierten Ansätzen kann der Aufwand durch die Vorarbeit stark verringert werden, da Fehlersimulation je nach Modell sehr teuer werden kann. Nur wenige der Diagnoseverfahren verwenden dabei jedoch die Zeitdaten der Schaltungen. Zudem wird oftmals die Information über fehlerhafte *und* fehlerfreie Ausgänge verwendet, um Kandidaten auszuschließen. Hierbei werden leider keine dynamischen Hazards in den Signalverläufen berücksichtigt, was z.B. bei nicht-robuster Fehlerpropagierung in die Irre führen kann. Simulationsbasierte Ansätze mit Auswertung der Signalverläufe durch explizite Zeitsimulation sind dem Autor nicht bekannt.

2.3 POINTER Diagnose mit Resimulation

Um das POINTER-Verfahren für SMALLDELAY-Fehler zu beobachten und auszuwerten, wurde der parallele Fehlersimulator in [Sch11] zunächst dazu genutzt, um fehlerhafte Testantworten von DUDs zu generieren. Mit Hilfe von einer Fehlermaschine zur Simulation von einzelnen Haftfehlern wurde das Syndrom eines DUDs ausgewertet. Da SMALLDELAY-Fehler an einem Gatter zwar unabhängig von der Polarität der Transitionen aktiviert werden können, aber die Aktivierung der Haftfehler vom aktuellen Signalwert abhängig ist, hatten sich die Evidenzen des Fehlers auf die jeweiligen Haftfehler-Repräsentanten (*StuckAt-0*, *StuckAt-1*) am Knoten

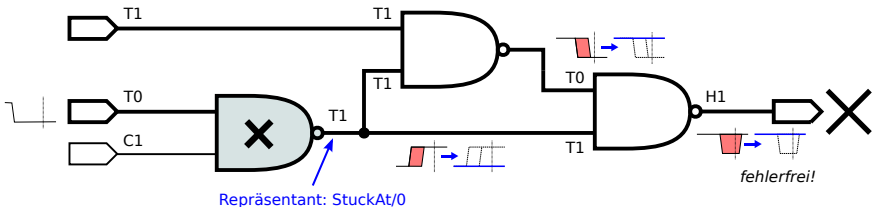


Abbildung 2.2: Beispiel eines Verzögerungsfehlers, der aufgrund von nicht robuster Signalpropagierung für das Testmuster nicht durch den Haftfehler-Repräsentanten am Gatter ersetzt werden kann.

aufgeteilt. In Abhängigkeit der Häufigkeit der Polarität bei fehleraktivierenden Transitionen, hatte sich hierbei gezeigt, dass die Evidenzen *einzelner* Haftfehler nicht zur Diagnose geeignet sind. Deswegen wurden die Evidenzen der Haftfehler-Repräsentanten eines Gatters durch komponentenweises addieren in sogenannte **Lokationsevidenzen** zusammengefasst [Sch11].

Da Haftfehler unabhängig von einer Defektgröße sind und ohne Transitionen aktiviert werden, haben die Lokationsevidenzen sehr viele Fehlvorhersagen produziert. Solche Haftfehler, die näher an den Ausgängen waren und kleinere Ausgangskegel hatten, wurden typischerweise besser bewertet als jene, die im Eingangsbereich lagen, wodurch diese bei kleineren Verzögerungsfehlern irreführende Ergebnisse geliefert hatten. Weil SMALLDELAY-Fehler an einem Knoten in Abhängigkeit der Defektgröße oftmals nur einen bestimmten Anteil aller sensibilisierten Pfade durch diesen verletzen, ist häufig der Fall aufgetreten, dass $\sigma_{f,\pi} > 0$ und $\iota_{f,\pi} > 0$, wodurch $\gamma_{f,\pi} > 0$. Aus diesem Grund musste die **Bewertungsfunktion** der Evidenzen auf **Sigma-Iota-Gamma** geändert werden, da sehr viele falsche bzw. unlogische Fehler (z.B. $\sigma_f = 0 \Rightarrow \gamma_f = 0$) besser gewichtet worden sind und die eigentlichen Kandidaten sehr weit von den ersten Rängen verdrängt hatten [Sch11].

Bei der sogenannten **Resimulation** wurden die Fehlerstellen der sortierten und bewerteten Lokationsevidenzen mit Hilfe des Zeitsimulators nachsimuliert, um das Verhalten von Verzögerungsfehlern an den Kandidatenstellen zum Abtastzeitpunkt zu beobachten. Aus Gründen der Performance wurde dabei für eine beschränkte Anzahl an Knoten (die besten 25 berichteten Fehlerstellen) jeweils der Transitionsfehler und die mittlere Defektgröße des Defektgrößen-Intervalls simuliert. Dabei konnte mit der Resimulation eine Verbesserung der Diagnoseergebnisse erzielt werden [Sch11].

2.4 Neuer Ansatz

Das in dieser Arbeit präsentierte Verfahren setzt gänzlich auf SMALLDELAY-Referenzfehler und wendet sich damit von der Haftfehlersimulation ab, da Haftfehler die Verzögerungsfehler

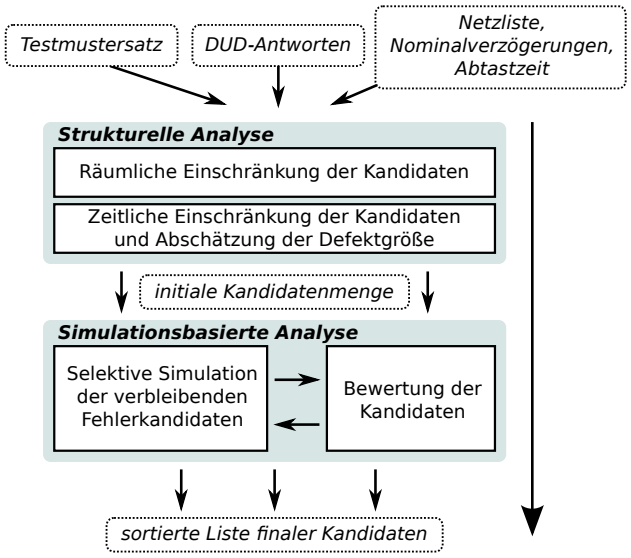


Abbildung 2.3: Übersicht des zweistufigen Diagnoseansatzes.

nicht in einem ausreichenden Maß repräsentieren können. Dies ist an einem Beispiel in Abbildung 2.2 illustriert. Hier produziert ein Verzögerungsfehler für eine steigende Transition einen fehlerhaften Ausgang. Da sich die Fehlerpropagierung am folgenden Verzweigungsstamm auf zwei Pfade verteilt und diese später (mit umgekehrter Polarität) rekonvergieren, ist das Ausgangssignal nur für einen kurzen Moment als Hazard sichtbar. Der zugehörige Haftfehler-Repräsentant (*StuckAt-0*) kann dieses Symptom jedoch nicht reproduzieren, da dieser das Ausgangsgatter konstant auf einem kontrollierten Wert hält. Für das Ausgangsgatter findet sich hier jedoch ein Haftfehler (*StuckAt-1*) um das Symptom zu reproduzieren, wodurch dieser nun besser bewertet wird als der eigentliche Kandidat.

Um die Simulation der SMALLDELAYs effizient zu gestalten, soll dabei zunächst die Menge der logischen Kandidaten eingeschränkt werden. Als nächsten Schritt sollen in **selektiven Verfahren** die zu simulierenden Kandidaten *adaptiv* anhand des Evidenzverhaltens bestimmt werden, um die Zahl der Simulationen gering zu halten.

Ein weiteres Problem beim Vergleichen von Referenzfehler-Antworten mit dem Syndrom eines DUDs entsteht, wenn eine hohe **Dynamik** in den Ausgangssignalen vorliegt. So können dann bereits kleinste Abweichungen im Abtastzeitpunkt zwischen *fehlerhaft* und *fehlerfrei* unterscheiden. Insbesondere führen Variationen der Gatterverzögerungen dazu, dass das Fehlermodell das Verhalten des Defekts im DUD nicht mehr akkurat nachahmen kann, wodurch ein direkter Vergleich zusätzlich erschwert wird. Da der verwendete Zeitsimulator die Signalverläufe im Schaltkreis präzise modellieren kann, sollen diese genauer untersucht werden, um Aufschluss über die Stabilität von Ausgangssignalen zu geben und diese in die Bewertung von Kandidaten mit einfließen zu lassen.

3 Adaptive simulationsbasierte Diagnose

Dieses Kapitel beschreibt ein Verfahren zur Diagnose von Verzögerungsfehlern, das anhand der Testantworten des zu diagnostizierenden Chips (DUDs) sowohl die Defektstelle einzelner SMALLDELAY-Fehler lokalisieren als auch die Defektgröße bestimmen kann. Aufgrund der Komplexität des Verzögerungsfehlermodells, wurde das Verfahren aus zwei Teilen zusammengesetzt: Einer *strukturellen* und einer *simulationsbasierten* Analyse.

Der erste Abschnitt dieses Kapitels befasst sich mit der **strukturellen Analyse**. Hierbei werden fehlermodellunabhängige *Effect-Cause*-Methoden zur *räumlichen*, sowie *zeitlichen* Einschränkung der möglichen initialen Fehlerkandidaten eingesetzt und eine erste Abschätzung der Defektgröße durchgeführt.

Der zweite Teil beschreibt die **simulationsbasierte Analyse**. Dabei wird eine neuartige *Cause-Effect*-Methode vorgestellt, die mit Hilfe expliziter Simulation von SMALLDELAY-Fehlern die verbleibenden Kandidaten genauer untersucht. Die zu simulierenden Fehler werden dabei *adaptiv* bestimmt, um die Gesamtzahl der Simulationen zu begrenzen.

Im dritten Teil wird ein Verfahren zur **Bewertung** der Fehlerkandidaten eingeführt, welches die Stabilität in den Ausgangssignalen mit berücksichtigt und die Diagnoseresultate im Falle von Variationen im Schaltkreis verbessert.

3.1 Strukturelle Analyse

Die Anzahl der möglichen SMALLDELAY-Fehler in einem Schaltkreis hängt von der Zahl der Pfade ab [Sch11] und kann im schlimmsten Fall *exponentiell* mit der Gatterzahl N steigen, weshalb eine erschöpfende Fehler-Simulation zur Diagnose mit allen Kandidaten praktisch nicht durchsetzbar ist [WWW06]. Da industrielle Schaltkreise — insbesondere *Full-Scan*-Designs — typischerweise kurze Pfade und kleine Kegel besitzen, haben Verzögerungsfehler an einem Gatter meist jedoch nur auf einen relativ kleinen Bereich der Schaltung Einfluss. Dadurch sind Fehler häufig nur an einer kleinen Anzahl von Ausgängen sichtbar. Umgekehrt lässt sich die Fehlerquelle anhand der fehlerhaften Ausgänge ebenso auf einen Bereich der Schaltung *räumlich* eingrenzen. Fehlerkandidaten, die sich außerhalb dieser Bereiche befinden, können ohne explizites Überprüfen durch Fehlersimulation von den Untersuchungen ausgeschlossen werden. Die strukturellen Analyse hat nun als Aufgabe diese Bereiche zu identifizieren. Als Ausgangspunkt dienen dabei die jeweiligen Testantworten der zu diagnostizierenden Chips.

3.1.1 Räumliche Einschränkung der Kandidaten

Damit eine Fehlerstelle zur Menge der potentiellen Kandidaten angehören kann, müssen bei dem Knoten die folgenden beiden Punkte für *jeden* fehlerhaften Ausgang der Testmuster erfüllt sein:

Fehler-Aktivierung Es kommen nur solche Knoten als Kandidaten in Frage, die bei *allen* fehlerhaften Verzögerungstests mindestens eine Transition im Signalverlauf aufweisen.

Fehler-Propagierung Ausgehend von der Defektstelle der Kandidaten, müssen bei aktiviertem Fehler die Transitionen an der Fehlerstelle zu *jedem* betroffenen Ausgang des fehlerhaften Tests propagieren können.

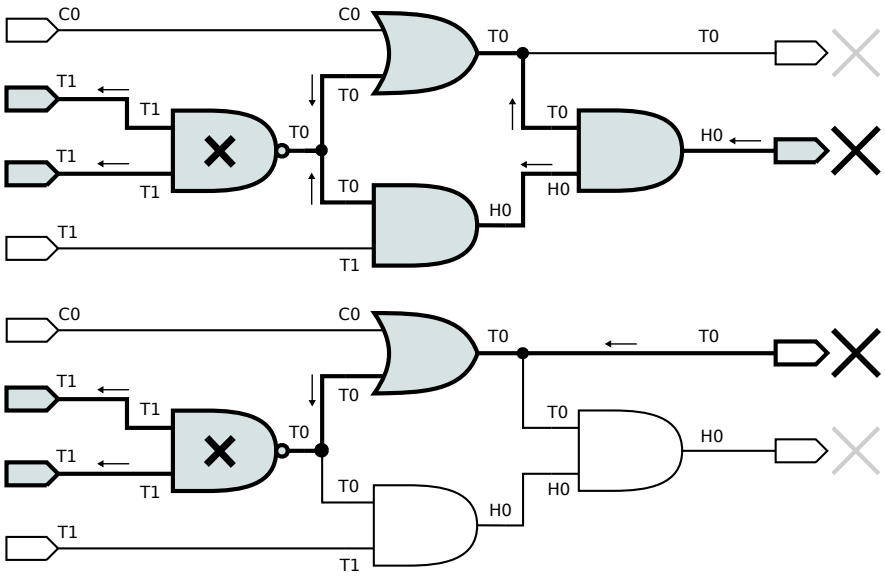
Alle Knoten, die mindestens einmal gegen diese Bedingungen verstoßen, können von den Kandidaten logisch ausgeschlossen werden. Um die Bedingungen zu überprüfen, wurde das in Abschnitt 2.2.1 vorgestellte *Critical Path Tracing* für Verzögerungsfehler [GLP95] gewählt.

Critical Path Tracing

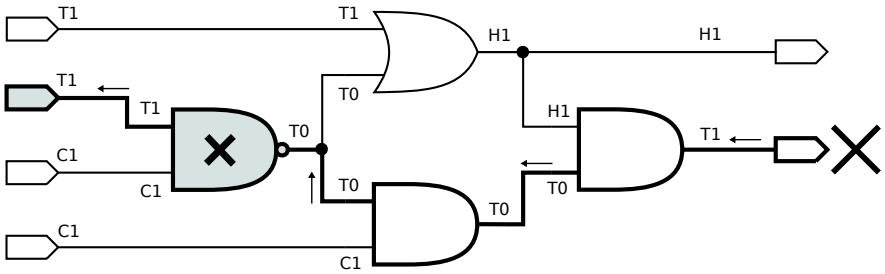
Das implementierte Critical Path Tracing (CPT) beruht auf Grundversion von [GLP95]. Ausgehend von einer Logiksimulation der fehlerhaften Verzögerungstests in sechswertiger Logik [Hay86], werden die fehlerhaften Ausgänge des DUDs über die sensibilisierten Leitungen in der Schaltung bis zu den Eingängen verfolgt und markiert. Von jeden fehlerhaften Ausgang eines jeden Testmusters werden die Gatter mit möglichen Transienten (T0, T1, H0, H1) in einer Backtracing-Iteration traversiert. Im Rahmen einer Optimierung aus [GLP95] werden dabei Signale mit **dominierenden** Ereignissen bevorzugt, d.h. wenn ein Gatter an den Eingängen sowohl dominierende als auch nicht-dominierende Ereignisse aufweist, dann wird das Backtracing nur an den jeweiligen *dominierenden* Eingängen fortgesetzt.

Nach der Rückverfolgung eines fehlerhaften Ausgangs werden die markierten Gatter in einer Liste S_i zusammengefasst und mit den verbleibenden Kandidaten *geschnitten*, um eine initiale Kandidaten-Liste S $(= \cap S_i)$ zu erhalten. Jeder dieser Kandidaten $s \in S$ erfüllt dabei die Aktivierungs- und Propagierungsbedingungen für das Syndrom des DUDs, während die Gatter aus $V \setminus S$ gegen mindestens eine dieser Bedingungen verstoßen haben.

In Abbildung 3.1 ist das Critical Path Tracing an einem Beispiel illustriert. Hier wurde das CPT auf zwei Verzögerungstests angewendet. Dicken Linien markieren die Teile der Schaltung, die beim Backtracing zurückverfolgt werden konnten. Beim ersten Verzögerungstest (1–2) muss die Prozedur für jeden der beiden fehlerhaften Ausgänge separat aufgerufen werden. Nach Rückverfolgung des zweiten Tests (3) reduziert sich die logische Kandidatenmenge auf zwei Stellen. Alle verbleibenden Kandidaten nach einem Schritt sind schattiert dargestellt.



a) Test 1 mit *zwei* fehlerhaften Bits.



b) Test 2 mit *einem* fehlerhaften Bit.

Abbildung 3.1: Einfaches *Critical Path Tracing* am Beispiel mit zwei fehlerhaften Verzögerungstests. – Das defekte Gatter ist mit einem ‚X' markiert. Die Symbole der Logiksimulation vom untersuchten Test sind an den Leitungen dargestellt.

Kollabierung struktureller Äquivalenzen

Verzögerungsfehler an unterschiedlichen Gattern in einem Schaltkreis können für einen Test oder einen ganzen Testmustersatz ein identisches Syndrom verursachen und sich dadurch *äquivalent* verhalten. Dies kann dabei mehrere Ursachen haben:

- Der Testmustersatz kann die Fehler nicht weiter unterscheiden, weil der Fehlereffekt z.B. aufgrund blockierender Seitenpfade nicht an alle Ausgänge propagiert werden kann.
- Die Gatter und deren assoziierte Fehler sind *strukturell äquivalent* und produzieren für **alle** Testmuster ein identisches Syndrom.

Im ersten Fall können die Äquivalenzen aufgelöst werden, indem man versucht die Fehlerinformation, die von einer möglichen Fehler-Stelle aus propagiert wird, zu maximieren. Hierbei könnte man mit Hilfe von **ATPG-Tools (Automatic Test Pattern Generation)** einen geeigneten Testmustersatz erzeugen, der Transitionen über alle möglichen Pfade durch die Gatter verschickt und so ein vollständiges Bild des Fehlerausmaßes erzeugt. In der Praxis ist dies aber aufgrund der hohen Anzahl an möglichen Pfaden im Schaltkreis nicht durchführbar.

Im Fall einer **strukturellen Äquivalenz** lässt sich die Fehlerinformation dagegen nicht durch die Wahl eines anderen Testmustersatzes verbessern. Fehler mit fester Defektgröße können an diesen Gattern aufgrund des äquivalenten Verhaltens nicht unterschieden werden, wodurch die diagnostische Auflösung *beschränkt* wird. Deshalb werden die im Anschluss an das Critical Path Tracing verbleibenden Fehlerkandidaten von strukturell äquivalenten Knoten in *Äquivalenzklassen* eingeteilt, sodass der Diagnosealgorithmus die Symptome eines DUDs nun zu bestimmten Fehlerklassen zuordnen kann. Die Gatter der *Äquivalenzklassen* werden für SMALLDELAYs dabei gemäß einem Vorschlag aus [Sch11] zusammengefasst, der auf einer Methode aus [WLRI87] zur Bestimmung der strukturellen Äquivalenzen bei Transitionsfehlern basiert. Hierfür werden die folgenden Regeln verwendet:

(R1) Sei n ein Gatter mit einem einzigen Fanout g. Dann sind n und g *strukturell äquivalent*, wenn g einen einzigen Eingang hat.

(R2) Seien n und g bzw. g und h strukturell äquivalent, dann sind n und h ebenfalls strukturell äquivalent (*Transitivität*).

Mit g als einzigem Nachfolger von n sind die beiden Gatter Teil eines Fanout-freien Bereichs. Da g keine zusätzlichen *Nebenpfade* (Off-Path Signale) hat, ist die Menge an Pfaden durch den Knoten n identisch mit denen durch g, wodurch alle injizierten SMALLDELAY-Fehler dieselben Aktivierungs- und Propagierungsbedingungen haben. Dabei bleibt auch die Puls-Filterung der Signale entlang der Kette konsistent, sodass das Verhalten von Fehlern gleicher Größe an n und g identisch ist. Die Gatter können somit durch keinen Verzögerungstest unterschieden werden und sind deshalb *strukturell äquivalent* (R1). Mit Hilfe der *transitiven* Eigenschaft (R2) werden die strukturell äquivalenten Gatter nun folgendermaßen in Äquivalenzklassen eingeteilt:

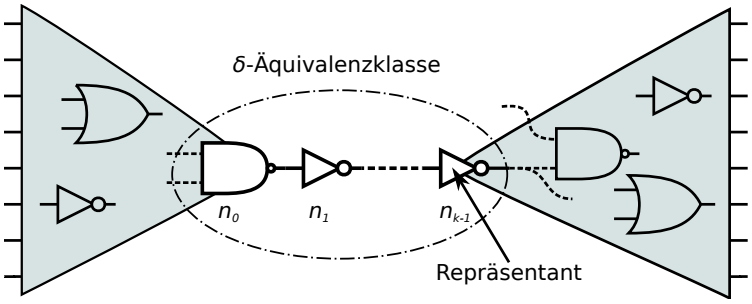


Abbildung 3.2: Strukturelle Äquivalenz. – Kollabierung einer δ-Äquivalenzklasse zu einem Repräsentanten.

Definition 3.1 (δ-Äquivalenzklasse)**.** *Sei $H \subseteq G$ ein verbundener Graph bestehend aus einer Kette von k strukturell äquivalenten Gattern $n_0, n_1, ..., n_{k-1}$. Ist H maximal und gemäß den vorigen Regeln, dann bilden die Knoten von H eine δ-Äquivalenzklasse.*

Die Gatter der Äquivalenzklassen und deren assoziierte Fehler werden nun zu einzelnen *Repräsentanten* reduziert (*Fault-Collapsing*), die das Verhalten der Fehler in ihrer Klasse widerspiegeln. Als **Repräsentant** wurde das vorderste Glied der Kette einer jeden Äquivalenzklasse gewählt. Einzelne Gatter, die nicht nach Definition 3.1 in Gruppen zusammengefasst werden konnten, bilden jeweils eigenständige Klassen. Die Beschränkung der CPT Kandidaten auf die Repräsentanten von Äquivalenzklassen verringert dabei den *Aufwand* weiterer logischer Diagnose-Verfahren. Falls die logische Diagnose zur Steuerung einer physischen Untersuchung (z.B. mit dem Elektronenraster-Mikroskop) verwendet werden soll, so müssen die einzelnen Fehlerstellen der Äquivalenzklassen dennoch explizit untersucht werden, um die defekte Stelle auszumachen. Abbildung 3.2 illustriert das Schema einer Kollabierung von strukturell äquivalenten Gattern und zeigt die Äquivalenzklasse mit dem dazugehörigen Repräsentanten.

3.1.2 Zeitliche Einschränkung der Kandidaten

Mit dem Critical Path Tracing und der Kollabierung von strukturellen Äquivalenzen können die initialen Fehler-Kandidaten eines Schaltkreises *räumlich* reduziert werden. Da SMALLDELAYS mit ihrer individuellen Verzögerung auch eine Zeitkomponente besitzen, kann es vorkommen, dass manche Symptome aufgrund bestimmter Pfadlängen bei der Signalpropagierung nur für bestimmte Defektgrößen an den verbleibenden Knoten sichtbar sind. Dadurch kann zusätzlich eine globale **zeitliche** Abschätzung der Defekte *aller* Knoten durch Eingrenzung der DUD-Fehlergröße δ_f mit Hilfe einer Unter- (LB) und einer Obergrenze (UB) durchgeführt werden, sodass $\delta_f \in [LB, UB]$. Des Weiteren kann die Fehlerinformation dazu verwendet werden, um eine erste Abschätzung der Defektgröße zu gewinnen.

Bestimmung einer Untergrenze

Zur Bestimmung einer **Defekt-Untergrenze** wird die *minimale* (EA) und *maximale* (LS) kumulative Signal-Propagierungszeit von den Eingängen bis zu den Knoten durch den Schaltkreis propagiert [Sch11]. Die Zeiten errechnen sich über die Nominalschaltzeiten der jeweiligen Gatter: Sei $n \in V$ ein Gatter mit einer Nominalverzögerung von δ, dann gilt

$$EA(n) = \min_{i \in FanIn(n)} (EA(i)) + \delta \quad \text{und} \quad LS(n) = \max_{i \in FanIn(n)} (LS(i)) + \delta.$$

Die Eingänge $i \in I \subseteq V$ des Schaltkreises werden mit $EA(i) = 0$ und $LS(i) = 0$ initialisiert. An einem Schaltungsausgang $o \in O \subseteq V$ kann man aus den Zeiten $EA(o)$ und $LS(o)$ die kumulative Verzögerung des längsten bzw. kürzesten Pfades von den Schaltungseingängen bis Ausgang o ableiten. Mit Hilfe der **Slacks** $\Delta^T_{min} = T - LS(o)$ bzw. $\Delta^T_{max} = T - EA(o)$ dieser Pfade lässt sich dabei ein Defektgrößen-Intervall für den Ausgang o bestimmen [Sch11]:

$$D_o = [\Delta^T_{min}, \Delta^T_{max}].$$

Das **Defekt-Intervall** D_o vereint alle *relevanten* Defektgrößen eines Ausgangs o. So gibt $\min(D_o)$ die untere Schranke der detektierbaren Defektgrößen an. Defekte mit einer Verzögerung $\delta_f < \min(D_o)$ werden am Ausgang nicht erkannt, und müssen deswegen nicht weiter betrachtet werden. Die obere Schranke $\max(D_o)$ repräsentiert dagegen die kleinstmögliche Transitionsfehler-Größe an diesem Ausgang. Da jeder Transitionsfehler bereits den schlimmsten Fall eines Verzögerungsfehlers an einem Gatter modelliert, brauchen die Defekte $\delta_f > \max(D_o)$ im variationsfreien Fall ebenfalls nicht weiter betrachtet zu werden. Die Berechnung der Propagierungszeiten wird einmalig und mit einem Aufwand von $\mathcal{O}(N)$ durchgeführt.

Lemma 3.1.1. *Um die effektive Defekt-Untergrenze LB eines Fehlers anhand der fehlerhaften Ausgänge O_f zu bestimmen, wird im Folgenden die größte untere Schranke der Defekt-Intervalle D_o aller Ausgänge $o \in O_f$ berechnet:*

$$LB = \min(\cap_{o \in O_f} \{D_o\}) = \max(\{\min(D_o) : o \in O_f\}).$$

*Dadurch ist sichergestellt, dass die Zeitbedingung von **mindestens** einem Pfad durch einen jeden Ausgang verletzt ist.*

Beweis. Sei $LB = \max(\{\min(D_o) : o \in O_f\})$. Dann gibt es einen Ausgang $o \in O_f$, an dem ein längster Pfad p mit einem Slack $\Delta^T_{min} = LB$ endet. Nimmt man nun an, es existiere eine kleinere Schranke $LB' = LB - \epsilon$ für ein $\epsilon > 0$, dann ist die Slack-Differenz des längsten Pfades durch o und der neuen Schranke LB' gegeben mit $\Delta^T_{min} - \Delta^T_{LB'} = LB - LB' = \epsilon > 0$. Daraus folgt $o \notin O_f$ und ein Widerspruch. □

Beispiel 3.1. *Seien* $D_{o_1} = [3,5]$ *und* $D_{o_2} = [1,6]$ *die Defekt-Intervalle zweier fehlerhafter Ausgänge* $o_1, o_2 \in O_f$. *Dann ist* $LB = \max(\{\min([3,5]), \min([1,6])\}) = \max(\{3,1\}) = 3$ *die Mindestdefektgröße, die benötigt wird, um das Zeitverhalten von* o_1 *und* o_2 *zu verletzen.*

Für den variationsfreien Fall repräsentiert die hergeleitete Schranke LB immer eine **gültige** Untergrenze, mit der die Fehlermenge an den verbleibenden Gatter-Kandidaten zeitlich eingegrenzt werden kann.

Bestimmung einer Obergrenze

In [MMSTR08] werden die fehlerfreien Antworten dazu genutzt, um Aussagen über eine mögliche **Obergrenze** zu machen. Hierbei wird angenommen, dass Defekte, die einen Fehler über einen Pfad zu einem Ausgang $o \in O$ propagieren *können*, nicht groß genug sind, wenn o bei der Abtastung einen fehlerfreien Wert aufweist. Bei genauerer Betrachtung können **Glitches** und **Hazards** an den Schaltungsausgängen dabei jedoch einen *falschen Eindruck* verbreiten, wenn die Signale zum Zeitpunkt T der Abtastung einen gutartigen Wert besitzen. Als Konsequenz würde die Defekt-Obergrenze somit *unterschätzt* werden, wodurch der eigentliche Fehlerkandidat aus der Kandidatenmenge entfällt. Aus diesem Grund wurde in dieser Arbeit auf eine obere Beschränkung der Defektgröße durch fehlerfreie Antworten verzichtet.

Die Obergrenze UB wird deshalb mit der größten aller *kleinsten* Transitionsfehlergrößen $UB = \max(\cup_{o \in O_f} D_o)$ abgeschätzt, wodurch sichergestellt ist, dass alle betroffenen Ausgänge verletzt werden können und größere Defekte keine Änderungen am Syndrom verursachen.

Erste Abschätzung der Defektgröße

Um die Defektgröße in einer ersten Instanz *genauer* zu approximieren, wird mit dem Zeitsimulator zunächst eine **Gutsimulation** der Testmustermenge Π durchgeführt. Anschließend extrahiert man für jeden fehlerhaften Ausgang $o \in O_f$ eines fehlerhaften Verzögerungstests $\pi \in \Pi$ die *Stabilisierungszeit* $LS(o, \pi)$ im fehlerfreien Fall. Die **Stabilisierungszeit** eines Ausgangs gibt dabei den Zeitpunkt der letzten Transition im Signalverlauf an.

Mit Hilfe der Abtastzeit T kann man den **Slack** $\Delta^T(o) = T - LS(o, \pi)$ am Schaltungsausgang $o \in O_f$ eines fehlerhaften Verzögerungstests π berechnen und erhält somit eine mögliche Defektgröße, die zur Verletzung der Zeitbedingungen an o verwendet werden kann. Hierbei wird angenommen, dass die Zeitbedingungen an einem Knoten verletzt sind, sobald der Slack $\Delta^T(o) \leq 0$ ist. Da die Zeitbedingung eines jeden fehlerhaften Ausgangs $o \in O_f$ notwendigerweise verletzt sein **muss**, wird das **Maximum** der berechneten Slacks zur Abschätzung der möglichen Defektgröße δ_{est} genommen:

$$\delta_{est} = \max(\{\Delta^T(o) : o \in O_f\}).$$

Je nachdem, über welche Pfade die Verzögerungstests die Transitionen durch den Schaltkreis propagieren, können die Stabilisierungszeiten der Signale und die daraus resultierenden Slacks *variieren*.

Aufgrund der Berücksichtigung der Signalpropagierung in der Zeitsimulation kann die abgeschätzte Defektgröße δ_{est} die echte Defektgröße δ_f insbesondere bei *robusten* Verzögerungstests relativ gut approximieren.

Falls die Stabilisierungszeiten an einem Ausgang durch einen Pfad bestimmt ist, der nicht durch die Fehlerstelle führt, ist es möglich, dass δ_f **unterschätzt** wird. Das tritt hauptsächlich dann auf, wenn der Seiten-Pfad eine größere Propagierungzeit hat, als der Pfad durch die Fehlerstelle. Das Unterschätzen ist aber nicht weiter schlimm, da δ_{est} in diesem Fall immer noch eine gültige Untergrenze ist.

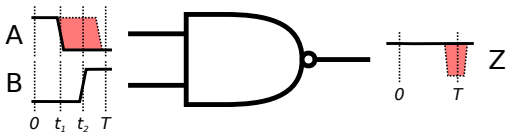


Abbildung 3.3: Überschätzung der Defektgröße durch blockierte Transitionen in der Zeitsimulation. Das Ausgangssignal ist hier konstant, wodurch $\delta_{est} = T$.

Bei *nicht-robuster* Signalpropagierung kann die Defektgröße durch Zeitsimulation allerdings auch **überschätzt** werden. Dies ist in Abbildung 3.3 am Beispiel eines NAND-Gatters zu sehen. Während der Gutsimulation erreicht das Eingangssignal A einen kontrollierenden Zustand zur Zeit $t_1 > 0$, noch bevor B diesen bei $t_2 > t_1$ verlässt. Die Propagierung der Transitionen wird *maskiert*, wodurch der Gatterausgang auf einem konstant kontrollierten Wert bleibt. Die Stabilisierungszeit von Z ist aufgrund der fehlender Transitionen gleich null und es ergibt sich eine Untergrenze von $\delta_{est} = T$. Da jedoch $t_1 > 0$ und der tatsächliche Defekt δ_f somit auch eine Größe von $\delta_f = T - t_1$ haben kann, würde dieser Defekt bei der Begrenzung der Kandidaten ausgeschlossen werden, da $\delta_f < \delta_{est}$. Aus diesem Grund ist δ_{est} **keine** gültige Untergrenze und eignet sich deshalb nicht zur Einschränkung der Kandidatenmenge.

3.2 Simulationsbasierte Analyse

Nachdem die initialen Fehler-Kandidaten durch die strukturelle Analyse räumlich und zeitlich reduziert worden sind, werden nun die verbleibenden Fehlerstellen durch explizite Fehlersimulation genauer untersucht, um so den eigentlichen Defekt lokalisieren zu können.

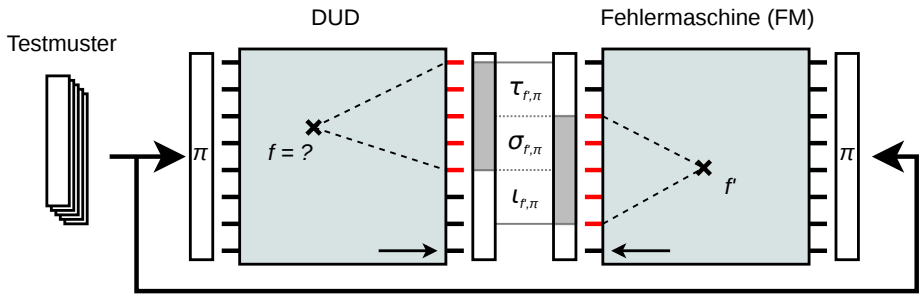


Abbildung 3.4: Berechnung der Evidenz-Komponenten ($\sigma_{f',\pi}$, $\tau_{f',\pi}$ und $\iota_{f',\pi}$) eines Referenzfehlers f' für ein Testmuster π durch Vergleich der fehlerhaften Ausgänge.

3.2.1 Aufbau

Abbildung 3.4 zeigt den allgemeinen Aufbau der simulationsbasierten Analyse. Hierbei wird ein Antwortmuster des zu diagnostizierenden Chips (DUD) mit denen der von einer **Fehlermaschine (FM)** simulierten Referenzfehler verglichen. Die FM ist hier ein Zeitsimulator, der einzelne SMALLDELAY-Fehler in die Schaltung injizieren und diese mit Hilfe der Signalverläufe (*Waveforms*) präzise an beliebigen Zeitpunkten evaluieren kann.

Für jeden Verzögerungstest π werden beim Vergleichen der Antworten eines Referenzfehlers f aus der Kandidatenmenge anhand die fehlerhaften Bits von DUD und FM sogenannte *Evidenzen* $e(f,\pi)$ berechnet. Evidenzen sind Tupel, die sich aus den Komponenten $\sigma_{f,\pi}$, $\tau_{f,\pi}$, sowie $\iota_{f,\pi}$ zusammensetzen (vgl. [HW07]). Durch komponentenweises Addieren der $e(f,\pi)$ eines jeden Testmusters $\pi \in \Pi$, erhält man die *globale* Evidenz $e(f,\Pi)$ des Referenzfehlers f für den gesamten Testmustersatz. Nach Simulation aller Referenzfehler werden die globalen Evidenzen mit Hilfe einer Funktion bewertet, um so eine sortierte Liste von Kandidaten zu erhalten, die den Defekt im DUD am besten erklären können.

3.2.2 Adaptive Simulation

Nach der strukturellen Analyse kann die Zahl der möglichen Fehlerkandidaten immer noch sehr groß sein. Diese hängt dabei sowohl von der Menge der verbleibenden Fehlerstellen S nach dem Critical Path Tracing, als auch von der Zahl der verschiedenen zu untersuchenden Defektgrößen an jedem Knoten ab. Bei der Fehlergenerierung im SMALLDELAY-Fehlermodell sind die einzelnen Defektgrößen eines Knoten durch die Menge der unterschiedlich langen Pfade durch diesen bestimmt [Sch11]. Jeder im Fehlermodell generierte Defekt eines Knotens verletzt dabei in Abhängigkeit der assoziierten Defektgröße δ die Zeitbedingungen aller Pfade mit der Länge größer oder gleich $T - \delta$.

Sei nun k die gemittelte Anzahl an verschieden langen Pfaden durch die Knoten in G, dann existieren durchschnittlich k verschiedene SMALLDELAY-Fehler für einen jeden Knoten im Modell, womit die Gesamtzahl der zu simulierenden Fehler mit $\mathcal{O}(k \cdot |S|)$ abgeschätzt werden kann. Da Fehlersimulation mit dem Zeitsimulator ein sehr aufwändiger Prozess ist und k in der Regel sehr groß ist, wäre ein simulationsbasierter Ansatz mit der vollständigen verbliebenen Kandidatenmenge S aus den $\mathcal{O}(k \cdot |S|)$ Fehlern somit immer noch sehr kostspielig. Aus diesem Grund muss die Zahl der Simulationen noch weiter gesenkt werden.

Untersuchung des Evidenz-Verhaltens

Um den Simulationsaufwand weiter reduzieren zu können, werden in dieser Arbeit Annahmen über das *qualitative* Verhalten der Evidenzen von Fehlern an den einzelnen Kandidatenstellen gemacht.

Hierbei wurde zunächst das Verhalten von Fehlern an der eigentlichen Defektstelle

in Abhängigkeit der Defektgröße beobachtet. Die Zahl der übereinstimmenden fehlerhaften Bits $\sigma_{f'}$ eines Referenzfehlers f' wies dabei für eine Defektgröße $\delta_{f'} \approx \delta_f$ ein globales Maximum auf. Für $\delta_{f'} < \delta_f$ fielen diese typischerweise geringer aus, da sich die Zahl der fehlerhaften Antwortbits für kleinere Defekte verringert. Ebenso wurde bei $\delta_{f'} > \delta_f$ ein Verlust an $\sigma_{f'}$ festgestellt, was sich durch die Präsenz von Hazards erklären ließ, die über den Zeitpunkt der Abtastung hinausgeschoben worden sind. Bei den Fehlvorhersagen $\iota_{f'}$ konnte man des Weiteren beobachten, dass sich die Anzahl bei Erhöhung der Defektgröße im qualitativen Verlauf bis zur Sättigung stetig vergrößert hat. Lediglich bei $\delta_{f'} \approx \delta_f$ wurde ein vorübergehender Einbruch im Verlauf festgestellt. Hier hat $\iota_{f'}$ ein lokales Minimum aufgewiesen, da die Defektgrößen $\delta_{f'}$ und δ_f übereingestimmt hatten.

Zudem wurde angenommen, dass Defekte *einer* Größe an den unterschiedlichen Gattern aufgrund von strukturellen Korrelationen ein *gleichartiges* Fehlverhalten aufweisen. Erste Beobachtungen hatten dabei ergeben, dass sich die qualitativen Verläufe der Evidenzkomponenten $\sigma_{f'}$ und $\iota_{f'}$ für unterschiedlichen Kandidatenstellen gleich verhalten, da die Gatter ähnliche Aktivierungs- und Propagierungsbedingungen haben. In Abhängigkeit von der Struktur des Schaltkreises (z.B. durch Rekonvergenzen), können sich die Verläufe einzelner Gatter dennoch von anderen unterscheiden. Allerdings wurde unter den *Ausreißern* ebenso ein Gruppen-Verhalten festgestellt, sodass auch hier von Ähnlichkeiten im qualitativen Verlauf gesprochen werden kann.

Abbildung 3.5 illustriert die Ähnlichkeiten der Evidenzen am Beispiel der Kandidaten S nach der strukturellen Analyse mit 2560 Testantworten eines DUDs. Jede Kurve entspricht dem Verlauf der übereinstimmenden (Sigma) bzw. falsch vorhergesagten (Iota) fehlerhaften Bits aller simulierten Defekte an den Kandidatenstellen.

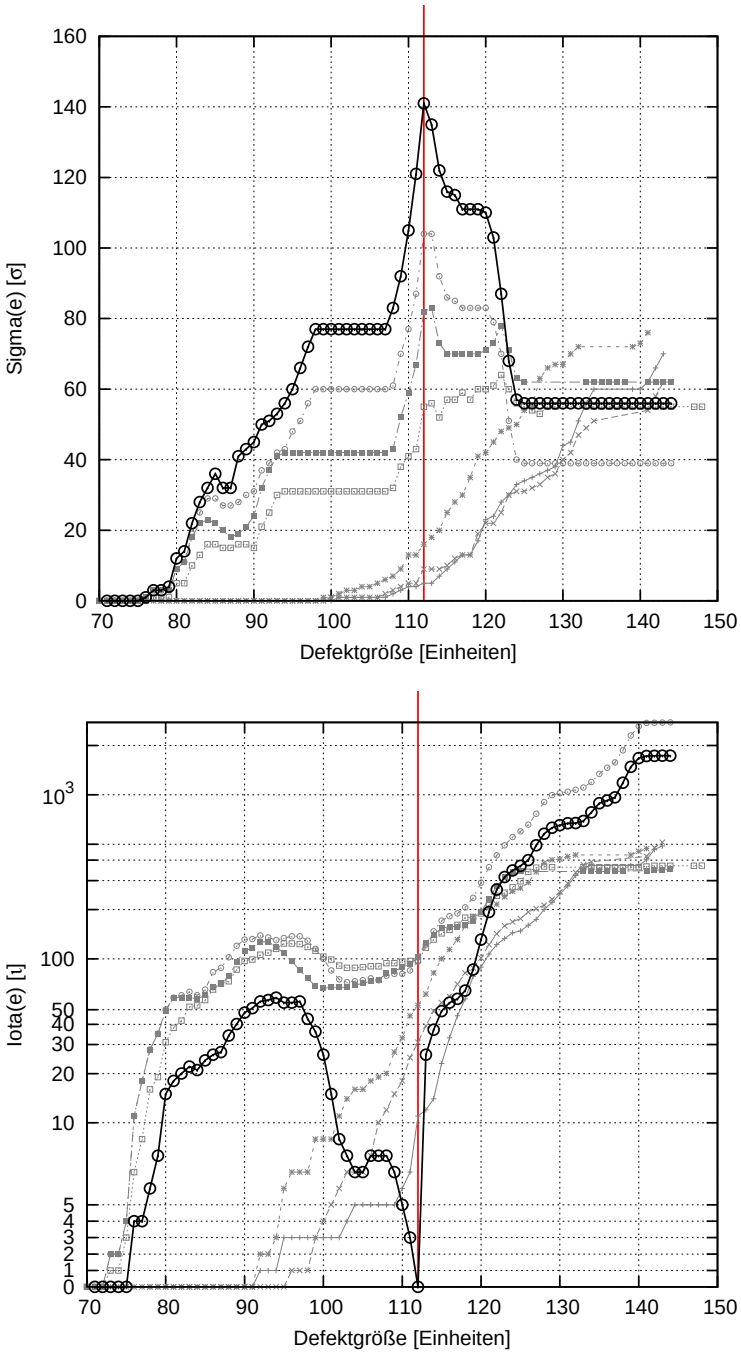


Abbildung 3.5: Sigma- und Iota-Werte von CPT-Kandidaten für einen Testsatz aus Pseudozufallsmustern. Die simulierten Defekte der eigentlichen Fehlerstelle sind mit schwarzen Kreisen eingezeichnet. Der senkrechte Strich markiert die eigentliche Größe des Defekts (112) im DUD.

Selektive Simulation der Kandidaten

Die Ähnlichkeiten in den Evidenzen werden nun ausgenutzt, um **selektiv** Kandidaten zu simulieren. Dabei werden die folgenden drei Schritte ausgeführt:

- Simuliere die in der ersten Instanz abgeschätzte Defektgröße δ_{est} an allen CPT-Kandidaten und bewerte die Evidenzen.
- Simuliere eine Menge verschiedener Defektgrößen an der Stelle des besten Kandidaten aus Schritt 1 und bewerte die Evidenzen.
- Simuliere die Defektgröße δ_{fin} des besten Kandidaten aus Schritt 2 an allen anderen CPT-Fehlerstellen und bewerte die Evidenzen.

Zunächst wird an allen Kandidaten S des Critical Path Tracing die in Abschnitt 3.1.2 abgeschätzte Defektgröße mit der Fehlermaschine simuliert. Die dabei berechneten Evidenzen werden anschließend mit Hilfe einer *Bewertungsfunktion* evaluiert (siehe nächster Abschnitt), um einen **Einstiegspunkt** $s \in S$ zu finden. Wenn die strukturelle Analyse bereits eine gute Approximation der Defektgröße liefert, sodass $\delta_{est} \approx \delta_f$, dann befindet sich der Eingangspunkt mit hoher Wahrscheinlichkeit an der eigentlichen Defektstelle.

Im zweiten Schritt wird anschließend die abgeschätzte Defektgröße δ_{est} **korrigiert**, indem das Verhalten einer Menge von Defekten *verschiedener* Größen am Einstiegspunkt s simuliert und beobachtet wird. Hinsichtlich der vielen Pfade im Schaltkreis ist die Berechnung einer vollständigen Liste mit allen Fehler eines Knotens nach dem Modell sehr aufwändig. Die zu simulierenden Fehler werden deshalb durch eine Quantisierung des Defekt-Intervalls am Einstiegspunkt gewonnen, welches folgendermaßen nach [Sch11] bestimmt wird: Sei $\delta_{min}(s)$ der kleinste detektierbare Defekt und $\delta_{max}(s)$ der Transitionsfehler des Knoten s, dann repräsentiert $D_s = [\delta_{min}(s), \delta_{max}(s)]$ das Defektgrößen-Intervall von s. Aus diesem lassen sich nun durch *Quantisieren* des Intervalls mit einem festen Faktor q insgesamt q verschiedene Defektgrößen in gleichen Abständen extrahieren. Diese können nun mit Hilfe der berechneten Ober- und Untergrenzen auf die Menge $D'_s = \text{QUANTISIERE}(D_s, q) \cap [LB, UB]$ reduziert werden.

> **Anmerkung:** Durch die Quantisierung des Defekt-Intervalls D_s in diskrete Werte kann es passieren, dass sich der eigentliche zu diagnostizierende Defekt nicht mehr unter den extrahierten Größen befindet, wodurch ein Fehler bei der Abschätzung eingeführt wird. Die Abweichung ist dabei abhängig vom gewählten Quantisierungsfaktor q, der die Abstände zwischen zwei aufeinanderfolgenden Defektgrößen in D'_s bestimmt. Der lokale Fehler der Abschätzung beträgt bei einer erfolgreichen Diagnose somit im Mittel $\Delta_{err} = |\delta_{fin} - \delta_f| \approx \frac{1}{2} \cdot \frac{1}{q}$ Prozent. Zeitgleich reduziert die Quantisierung jedoch die Komplexität des Fehlermodells in der Zeit-Komponente auf einen konstanten Quantisierungsfaktor $q < k$ und bietet einen Kompromiss zwischen Präzision und Aufwand.

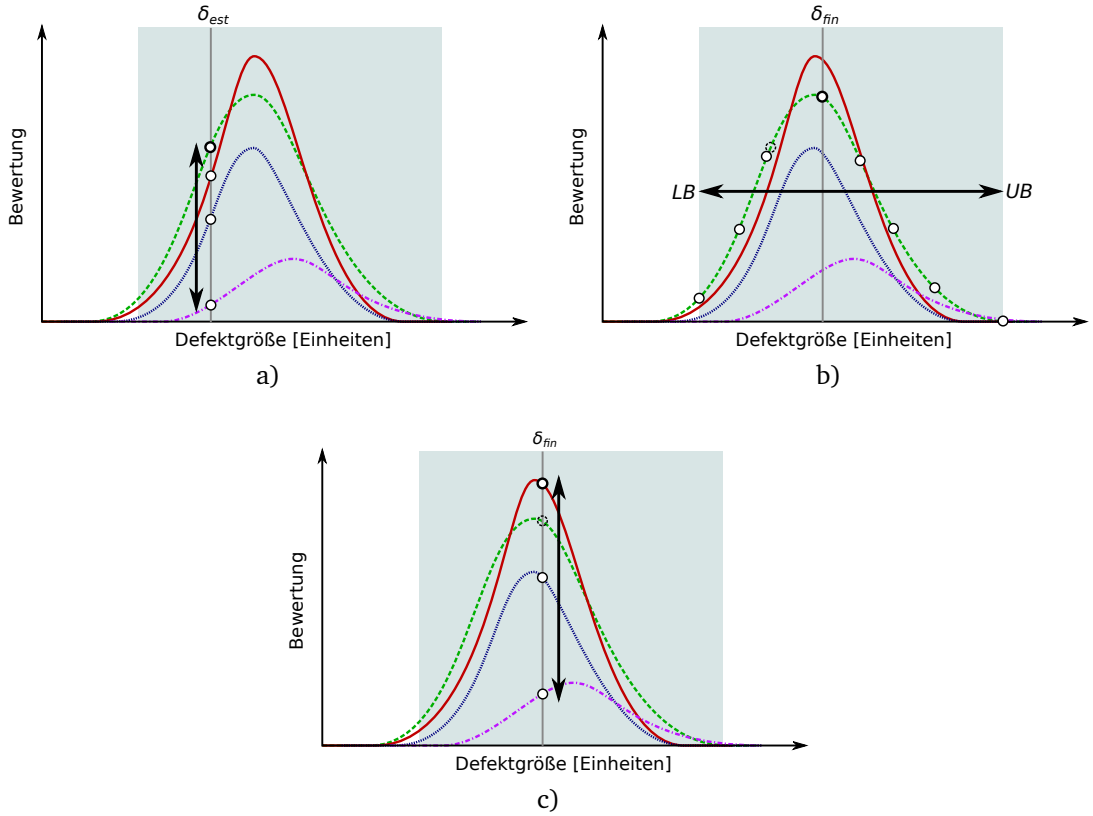


Abbildung 3.6: Illustration aller Schritte der selektiven Simulation. – a) Bestimmung des Einstiegspunktes. b) Korrektur der abgeschätzten Defektgröße. c) Finale Evaluierung der Kandidaten.

Die verbleibenden Defektgrößen in D'_s werden dann in der Fehlermaschine simuliert. Nach Bewertung der Evidenzen extrahiert man die Defektgröße δ_{fin} des besten Kandidaten, welche im Weiteren als neue Abschätzung von δ_f dient.

Im dritten und letzten Schritt werden die einzelnen CPT-Fehlerstellen unter der korrigierten Defektgröße δ_{fin} evaluiert und die Evidenzen berechnet. Da die korrigierte Defektgröße am Einstiegspunkt bereits im vorigen Schritt simuliert worden ist, muss diese hier nicht mehr zusätzlich ausgewertet werden. Nach einer letzten Bewertung der resultierenden Evidenzen erhält man dann die finale Rangliste mit den wahrscheinlichsten Kandidaten für den Defekt im DUD.

Zur Veranschaulichung der selektiven Simulationsmethode wurden die drei Schritte in Abbildung 3.6 illustriert. Im Vergleich zum vollen Simulationsansatz, müssen mit dieser Methode nur $q + 2 \cdot |S| - 1 \in \mathcal{O}(k + |S|)$ anstatt $\mathcal{O}(k \cdot |S|)$ Simulationen durchgeführt werden.

3.3 Bewertung der Kandidaten

Die einzelnen Komponenten σ_f, τ_f und ι_f der globalen Evidenz $e(f,\Pi)$ eines Kandidaten f geben Aufschluss darüber, wie gut dieser den Fehler im DUD erklären (Sigma) bzw. nicht erklären kann (Iota, Tau). Da Sigma und Tau durch FB[1] $= \sigma_f + \tau_f$ korrelieren (siehe Abb. 3.4), werden im Folgenden nur die Sigma und Iota Komponenten betrachtet. Mit Hilfe einer geeigneten Bewertungsfunktion können die globalen Evidenzen der Kandidaten verglichen werden, um eine sortierte Liste mit den wahrscheinlichsten Kandidaten zu erstellen. Dabei soll gelten: Je kleiner der Rang des Kandidaten in der Liste, umso besser erklärt er das Syndrom des DUD.

Simulierte Fehler an der eigentlichen Defektstelle können idealerweise alle Fehlerbits des DUD erklären und haben dabei keine Fehlvorhersagen. Dadurch haben diese typischerweise einen kleinen Rang. Die Bewertungsmethode der POINTER-Diagnose [HW07] wendet hierbei ein mehrstufiges Verfahren mit komponentenweiser Sortierung der Evidenzen an, welches Kandidaten zunächst nach den *meisten* erklärten Fehlerbits σ_f und anschließend nach den *wenigstens* Fehlvorhersagen ι_f sortiert. Bei einer hohen Dynamik in den Ausgangssignalen (z.B. Hazards) eignen sich diese Verfahren jedoch nicht zum Vergleichen von Verzögerungsfehlern, da bereits kleinste Abweichungen in der Defektgröße dazu führen können, dass nicht mehr alle fehlerhaften Antwortbits erklärt werden. Somit kann es passieren, dass der eigentliche Fehler aufgrund fehlender Sigmas von der Spitze der Kandidatenliste verdrängt wird und absteigt, unabhängig davon, ob der Kandidat Fehlvorhersagen produziert oder nicht.

3.3.1 *Reward & Penalty*

Hinsichtlich der oben erwähnten Problematik bezüglich der Dynamik in den Syndromen, wird hier für die Bewertung der Kandidaten eine **gewichtete Summe** über die einzelnen Evidenz-Komponenten verwendet. Die Summe berechnet einen *Score*-Wert, der eine gleichzeitige Auswertung der erklärten und falsch vorhergesagten Bits ermöglicht:

$$\text{score}(f) = \omega_\sigma \cdot \sigma_f + \omega_\iota \cdot \iota_f.$$

Dadurch ist es möglich, dass Kandidaten, die nicht alle Fehlerbits erklären, trotzdem die beste Bewertung haben können, wenn diese im Vergleich zu den anderen Kandidaten wenig Fehlvorhersagen aufweisen. Die Gewichte ω_σ und ω_ι wurden dabei so gewählt, dass für jedes erklärte Bit in σ_f ein Punkt an den Kandidaten verteilt und für jede falschen Vorhersage ι_f ein Teil als *Strafe* abgezogen wird. In der Literatur [WWW06] wird dieses Verfahren deshalb *„Reward and Penalty"* genannt.

Beim **Vergleich** der Evidenzen zweier Fehler f_1 und f_2 gilt somit:

$$\text{score}(f_1) > \text{score}(f_2) \Rightarrow \text{Rang}(f_1) < \text{Rang}(f_2).$$

[1] FB = Anzahl der fehlerhaften Bits im DUD Test.

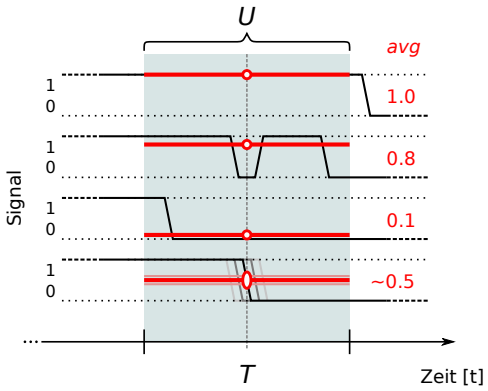


Abbildung 3.7: Mittelwerte avg(U) von Signalverläufen in einer Umgebung $U \ni T$.

3.3.2 Berücksichtigung der Signal-Stabilität

Wenn die Signalverläufe an den Schaltungsausgängen zum Zeitpunkt der Abtastung T eine gewisse Dynamik aufweisen, kann es passieren, dass bei kleinsten Abweichungen der Defektgröße eines Fehlers oder bei Variationen der Gatterverzögerungen im Schaltkreis, unterschiedliche Signalwerte abgetastet werden. Dadurch wird der direkte Vergleich des Syndroms eines simulierten Fehlers mit dem DUD erschwert. Aus diesem Grund werden die Ausgangssignale der Fehlermaschine im Folgenden in einem *Bereich* um den Abtastzeitpunkt T ausgewertet, um die Stabilität zu ermitteln.

Der Zeitsimulator liefert zunächst den genauen Verlauf eines Signals als *Waveform* signal(t), welches für alle $t \in \mathbb{R}_0^+$ evaluiert werden kann:

$$\text{signal} : \mathbb{R}_0^+ \to \{0,1\}.$$

Als einfaches Maß zur Bestimmung der Stabilität an einem Ausgang, betrachtet man diesen dann in einer Umgebung $U = [t_0, t_1] \subseteq \mathbb{R}_0^+$ um den Abtastzeitpunkt $T \in U$. Dabei berechnet man die Abweichung des *abgetasteten* Werts eines Ausgangs bei T vom **gemittelten** Wert des Signals über U. Da ein Signalverlauf eine partiell integrierbare Funktion bildet, kann der **Mittelwert** avg(U) $\in [0,1]$ folgendermaßen bestimmt werden:

$$\text{avg}(U) = \frac{1}{t_1 - t_0} \int_U \text{signal}(t)\ dt.$$

Zur Veranschaulichung wurden in Abbildung 3.7 Beispiele von berechneten Mittelwerten einzelner Signalverläufe illustriert. Der Mittelwert entspricht dabei gleichzeitig der Wahrscheinlichkeit $P[\text{„signal}(t) = 1\text{”}]$ bei gleichverteilt zufälligem Abtasten an einer Stelle $t \in U$.

Robuste Evaluierung der Syndrome

Bei der Evaluierung wird nun die Stabilität der berechneten Evidenzkomponenten verwendet, um eine **robuste** Auswertung der Syndrome im Falle von Variationen zu ermöglichen. Dazu wird im Folgenden zunächst der Begriff des Syndromsignals eines Ausgangs definiert:

Definition 3.2 (Syndromsignal)**.** *Das* ***Syndromsignal*** $\mathrm{syn}(t)$ *eines Schaltungsausganges ist eine Funktion des Ausgangssignals* $\mathrm{signal}(t)$ *in Abhängigkeit einer Zeitkomponente* $t \in \mathbb{R}_0^+$ *und dem stabilen Wert* $final = \lim_{t\to\infty} \mathrm{signal}(t) \in \{0,1\}$*. Es beschreibt die Boolesche Differenz von* $\mathrm{signal}(t)$ *und* $final$*:*

$$\mathrm{syn}(t) := \mathrm{signal}(t) \oplus final = \begin{cases} 1 - \mathrm{signal}(t) & \text{Wenn } final = 1, \\ \mathrm{signal}(t) & \text{sonst.} \end{cases}$$

Wie schon $\mathrm{signal}(t)$ ist auch $\mathrm{syn}(t)$ vollständig über $\mathbb{R}_0^+$ definiert und kann daher partiell integriert werden:

$$\int_U \mathrm{syn}(t)\ dt = \int_U (\mathrm{signal}(t) \oplus final)\ dt \quad = \begin{cases} (t_1 - t_0) - \int_U \mathrm{signal}(t)\ dt & \text{Wenn } final = 1, \\ \int_U \mathrm{signal}(t)\ dt & \text{sonst.} \end{cases}$$

Über den Mittelwert $\mathrm{avg}_{syn}(U)$ vom Syndromsignal eines Ausgangs $o \in O$ wird nun dessen **Fehlerwahrscheinlichkeit** $\omega_U(o)$ in der Umgebung $U \ni T$ für den aktuellen Test bestimmt. Da $\mathrm{syn}(t) = 1 \Leftrightarrow$ „$\mathrm{signal}(t)$ *fehlerhaft*", gilt $\omega_U(o) = \mathrm{avg}_{syn}(U) = P[$„Ausgang *fehlerhaft*"$]$. Bei der Auswertung des Testmusters fließt diese dann als Gewicht für die jeweiligen Evidenz-Komponenten ein.

Sei $\omega_U(o)$ die in einer Umgebung U gemessene Fehlerwahrscheinlichkeit eines Ausgangs $o \in O$ der FM. Ferner sei $\alpha_o = 1$ ($\alpha_o = 0$), wenn o im DUD *fehlerhaft* (*fehlerfrei*) ist. Beim Vergleich eines Tests π für einen Referenzfehler f wird für jeden Ausgang o des DUDs mit $\alpha_o = 1$ die Fehlerwahrscheinlichkeit $\omega_U(o)$ bestimmt. Aufgrund der Korrelation von $\sigma_{f,\pi}$ und $\tau_{f,\pi}$, werden diese um $\omega_U(o)$ bzw. $1 - \omega_U(o)$ erhöht, sodass

$$\sigma_{f,\pi} := \sum_{o \in O} (\alpha_o \cdot w_U(o))$$

und

$$\tau_{f,\pi} := \sum_{o \in O} (\alpha_o \cdot (1 - w_U(o))).$$

Da bei der Berechnung der $\sigma_{f,\pi}$ und $\tau_{f,\pi}$ alle fehlerhaften Bits des DUDs betrachtet werden, ermöglicht dies einen Teil der durch Variationen entgangenen Evidenzen mit $\omega_U(o)$ zu *rekonstruieren*.

Von den **fehlerfreien** Ausgängen des DUDs werden nur jene Ausgänge in der FM untersucht, die zum Zeitpunkt T einen fehlerhaften Wert aufweisen. Für jeden fehlerhaften Ausgang $o \in O_f$ wird $\iota_{f,\pi}$ dabei um $\omega_U(o)$ erhöht:

$$\iota_{f,\pi} := \sum_{o \in O_f} ((1 - \alpha_o) \cdot w_U(o))$$

Dies bedeutet im Fall, wenn das Syndromsignal eines zum Zeitpunkt T *fehlerhaften* Ausgangs der FM seinen Wert aufgrund einer oder mehrerer Transitionen innerhalb des *observierten* Zeitraums U wechselt, dass die Fehlerwahrscheinlichkeit für diesen Ausgang $P[\text{„}o \textit{ fehlerhaft} \text{ in } U\text{”}] = \omega_U(o) < 1$ ist. Mit $\omega_U(o)$ als Gewichtung können die von Hazards, Glitches und von nahe am Abtastzeitpunkt stattfindenen Transitionen verursachten Iotas der fehlerhaften Ausgänge somit reduziert werden. Dabei werden die Evidenzen an die Fehlerwahrscheinlichkeiten angepasst werden, wodurch eine robuste Auswertung bei Abweichungen der Defektgröße oder Variationen der Gatterverzögerungen ermöglicht wird. Die Fehlerwahrscheinlichkeiten der fehlerfreien Ausgänge der FM werden nicht weiter betrachtet. Eine zusätzliche Untersuchung dieser Ausgänge würde die Evaluierung durch zuviele Fehlvorhersagen negativ beeinflussen, da anhand der fehlerfreien DUD Antworten zum Zeitpunkt T mit dem Antwortmustersatz keine Annahmen über die Stabilität der jeweiligen Ausgänge gemacht werden können.

3.4 Zusammenfassung

In diesem Kapitel wurde ein Diagnose-Verfahren für kombinatorische Schaltungen vorgestellt, das einzelne SMALLDELAY-Fehler lokalisieren und deren Defektgröße abschätzen kann. Dabei kommen sowohl *Effect-Cause* als auch *Cause-Effect* Verfahren zum Einsatz.

Anhand der Fehlerinformationen des zu diagnostizierenden Chips, werden in einers strukturellen Analyse zunächst die initialen logischen Fehlerstellen mit dem Critical Path Tracing räumlich eingeschränkt. Des Weiteren wird eine globale zeitliche Eingrenzung, sowie eine erste genauere Abschätzung der möglichen Defektgröße unternommen.

Im Rahmen einer simulationsbasierten Analyse wurde eine neuartige Methode zur systematischen Findung der Fehlerursache in $\mathcal{O}(k + |S|)$ Schritten vorgestellt, welche mit Hilfe eines Zeitsimulators die Kandidaten effizient und präzise auswerten kann. Dabei wird die Stabilität der jeweiligen Ausgangssignale mit berücksichtigt, um im Falle von Variationen im Schaltkreis robuste Resultate zu liefern.

4 Ergebnisse

Dieses Kapitel beschreibt die Experimente und deren Ergebnisse, die im Rahmen dieser Arbeit durchgeführt wurden, um die Diagnostizierfähigkeit der präsentierten Methode für beliebige SMALLDELAY-Fehler zu untersuchen. Dabei wurden sowohl kleinere als auch größere industrielle Benchmark-Schaltkreise getestet. Zusätzlich wurde das Verhalten des Diagnosealgorithmus bei Gatter-Variationen und kompaktierten Test-Antworten überprüft.

4.1 Aufbau der Experimente

Timing-Daten der Schaltkreise

Der Zeitsimulator benötigt zur Simulation der Schaltkreise die Nominalschaltzeiten von allen Gattern. Diese müssen sowohl für steigende als auch fallende Flanken definiert sein. Im Rahmen dieser Arbeit wurden diese zur Vereinfachung in Form von Einheitsgrößen gewählt und sind für steigende und fallende Flanken **identisch**. Die Einheitsgrößen sind für die jeweiligen primitiven Gattertypen dabei wie folgt festgelegt worden:

- Gatter mit zwei Eingängen (AND, NAND, OR, NOR, XOR, XNOR) haben eine Verzögerung von 2 u.
- Gatter mit einem Eingang (INV, BUF) haben eine Verzögerung von 1 u.
- Schaltungseingänge und Ausgänge (INPUT, OUTPUT) haben keine Verzögerung (0 u).

Andere Gattertypen treten in den verwendeten Schaltkreisen dabei nicht auf.

Die **Abtastzeit** T an den Ausgängen wurde in Abhängigkeit der Tiefe h vom Schaltungsgraphen berechnet. Um einen genügend großen Abstand zwischen der maximalen Verzögerung des längsten Pfades und dem Abtastzeitpunkt einzuhalten, wurde zur Berechnung von T eine Verzögerung von drei Einheiten pro Ebene gewählt, sodass $T = (3 \cdot h)$ u. Dadurch ist sichergestellt, dass bei den fehlerfreien Schaltkreisen die Zeitbedingungen der Pfade auch unter Variationen einhalten werden können, da die maximale Verzögerungszeit eines Schaltungsknoten kleiner als die der Ebene ist.

Eine Übersicht mit den Daten aller untersuchten Schaltkreise befindet sich in Tabelle 4.1.

Schaltkreis		Eingänge	Ausgänge	Gatter	Tiefe	Pfade	SmallDelays
ISCAS '85	c432	36	7	216	29	$8.39 \cdot 10^4$	$2.05 \cdot 10^6$
	c499	41	32	246	14	$9.44 \cdot 10^3$	$1.30 \cdot 10^5$
	c880	60	26	435	30	$8.64 \cdot 10^3$	$1.89 \cdot 10^5$
	c1355	41	32	590	27	$4.17 \cdot 10^6$	$1.06 \cdot 10^8$
	c1908	33	25	1057	44	$7.29 \cdot 10^5$	$2.49 \cdot 10^7$
	c2670	233	140	1476	39	$6.80 \cdot 10^5$	$2.12 \cdot 10^7$
	c3540	50	22	1983	56	$2.87 \cdot 10^7$	$1.17 \cdot 10^9$
	c5315	178	123	2973	52	$1.34 \cdot 10^6$	$5.36 \cdot 10^7$
	c6288	32	32	2416	124	$9.89 \cdot 10^{19}$	$9.16 \cdot 10^{21}$
	c7552	207	108	4043	45	$7.26 \cdot 10^5$	$2.32 \cdot 10^7$
ISCAS '89	s35932	1763	2048	16353	29	$1.97 \cdot 10^5$	$5.02 \cdot 10^6$
	s38417	1664	1742	23537	48	$1.39 \cdot 10^6$	$4.52 \cdot 10^7$
	s38584	1464	1730	21462	59	$1.08 \cdot 10^6$	$4.72 \cdot 10^7$
ITC '99	b17	1452	1512	35549	103	$3.95 \cdot 10^{11}$	$2.43 \cdot 10^{13}$
	b20	522	512	21599	73	$2.90 \cdot 10^8$	$1.27 \cdot 10^{10}$
	b21	522	512	22055	74	$2.78 \cdot 10^8$	$1.22 \cdot 10^{10}$
	b22	767	757	32090	74	$4.37 \cdot 10^8$	$1.92 \cdot 10^{10}$
NXP	p35k	2912	2229	41443	72	$6.03 \cdot 10^9$	$3.57 \cdot 10^{11}$
	p81k	4029	3952	106450	55	$1.71 \cdot 10^8$	$5.77 \cdot 10^9$
	p89k	4632	4557	80963	109	$1.28 \cdot 10^8$	$6.55 \cdot 10^9$
	p100k	5902	5829	84356	104	$2.55 \cdot 10^{10}$	$1.69 \cdot 10^{12}$
	p141k	11290	10502	152808	92	$1.13 \cdot 10^9$	$5.21 \cdot 10^{10}$
	p259k	18713	18495	298796	204	$3.59 \cdot 10^{14}$	$4.32 \cdot 10^{16}$
	p269k	17333	16621	239771	105	$1.68 \cdot 10^7$	$5.69 \cdot 10^8$
	p286k	18351	17835	332726	178	$2.82 \cdot 10^{11}$	$2.38 \cdot 10^{13}$
	p378k	15732	17420	341315	108	$6.80 \cdot 10^8$	$2.26 \cdot 10^{10}$
	p418k	30430	29809	382633	234	$1.69 \cdot 10^{14}$	$1.60 \cdot 10^{16}$

Tabelle 4.1: Daten der im Rahmen dieser Arbeit untersuchten Benchmark-Schaltkreise von ISCAS '85, ISCAS '89, ITC '99 und NXP.

Fehler-Generierung

Um die Diagnose-Methode auszuwerten, wurde für verschiedene Schaltkreise der ISCAS '85, '89, sowie ITC '99 und NXP Benchmarks jeweils eine Liste mit zu diagnostizierenden SMALLDELAY-Fehlern bereitgestellt. Hierfür wurden über die Nominalschaltzeiten der Schaltkreise zunächst die *vollständigen* Fehlermengen nach dem pfadbasierten Modell [Sch11] *aufwändig* berechnet. Aufgrund von momentanen Einschränkungen in der Fehlersimulation, wurden in dieser Arbeit bei der Erzeugung der Fehler nur **Gatterausgänge** betrachtet [Sch11]. Ebenso werden Defekte an primären Schaltungseingängen und -Ausgängen bei der Generierung ausgeschlossen.

Die generierten Fehler der Liste sind anschließend in **zufälliger** Reihenfolge in eigenständige DUDs injiziert und im Zeitsimulator mit Verzögerungstests simuliert worden, um Testantworten zu erzeugen und die Detektierbarkeit der Fehler festzulegen. Für die Diagnoseexperimente wurden dabei jeweils eine Liste mit den ersten 1000 von den Testmustern erkannten Fehlern eines jeden Schaltkreises abgespeichert, die in den Folgeexperimenten als Antworten von *defekten* Chips wiederverwendet werden. Dadurch ist sowohl die Defektstelle als auch die Defektgröße der Fehler in den Listen zufällig.

Als Testmustersätze wurden die von ADAMA generierten **Pseudozufallsmuster** verwendet, welche in Paare aus Initialisierungs- und Propagierungsvektor unterteilt und zu Verzögerungstests zusammengesetzt wurden. Zum Testen der DUDs wurden dafür 10240 Testmuster generiert, was 5120 Verzögerungstests entspricht. Lediglich bei den ISCAS '85 Schaltkreisen sind aufgrund des Größenunterschieds nur halb so viele Muster verwendet worden.

Bewertung der Experimente

Bei der Bewertung des Diagnosealgorithmus werden die folgenden Kriterien betrachtet: Die *Diagnostizierbarkeit*, die *diagnostische Auflösung* und die *Abweichung* der geschätzten Defektgröße. Als **Diagnostizierbarkeit** wird dabei der Quotient aus (*Anzahl Erfolge / Anzahl Experimente*) bezeichnet. Die diagnostische **Auflösung** entspricht der Anzahl der vom Diagnose-Tool gemeldeten Kandidaten, welche im Anschluss an die logische Diagnose möglicherweise noch genauer untersucht werden müssen, um die eigentliche Defektstelle auf dem Chip zu lokalisieren. Die **Abweichung** der geschätzten Defektgröße gibt die prozentuale Abweichung des vorhergesagten Defekts relativ zur Breite des *globalen Defektgrößen-Intervalls* an, welches durch den kürzesten und längsten Pfad im Schaltkreis bestimmt ist.

Nach der Berechnung und Sortierung der Evidenzen aller finalen Fehlerkandidaten, wird die Liste in *Gruppen* aus Kandidaten mit identischen Evidenzen unterteilt, die anhand der Testantworten mit dem Algorithmus nicht weiter unterschieden werden können. Da die Reihenfolge der Kandidaten einer **Rang-Gruppe** je nach verwendetem Sortierverfahren unterschiedlich ausfallen kann, müssen die Evidenzen einer Gruppe deshalb gleichwertig behandelt werden. Deswegen wird hier zur Bewertung ausschließlich der *mittlere Rang* der Kandidaten verwendet. Der **Mittel-Rang** eines Kandidaten s ist der *durchschnittliche* Rang aller Kandidaten in seiner Rang-Gruppe. Sei $RG(s)$ die Menge der Kandidaten in der Rang-Gruppe von s, und Index($RG(s)$) der beste Rang in $RG(s)$, dann ist der Mittel-Rang von s (und auch allen anderen Kandidaten der selben Rang-Gruppe) definiert durch:

$$\text{MittelRang}(s) = \text{Index}(RG(s)) + \left\lfloor \frac{|RG(s)|}{2} \right\rfloor .$$

Je kleiner der diagnostizierte Mittel-Rang des echten Kandidaten ist, umso besser ist das Ergebnis der Diagnose. Die Fehlerdiagnose eines DUDs wird als **Erfolg** oder *erfolgreich* bezeichnet,

Rang	Evidenz	Rang-Gruppe	Index(RG)	Mittel-Rang
1	e_0	1	1	2
2	e_0	1	1	2
3	e_1	2	3	4
4	e_1	2	3	4
5	e_1	2	3	4
6	e_2	3	6	6

Tabelle 4.2: Beispiel. – Rang-Gruppe und Mittel-Rang einer sortierten Liste von Evidenzen.

wenn sich der Mittel-Rang des wirklichen Fehlerkandidaten zwischen *eins* und *zehn* befindet. Andernfalls wird der Fehler als *nicht diagnostizierbar* angenommen und sein Mittel-Rang auf *elf* gesetzt. Hat der Kandidat eines zu diagnostizierenden Fehlers einen Mittel-Rang von *eins*, so ist der Fehler *perfekt diagnostiziert* worden.

Da in dem hier vorgestellten Verfahren immer eine bewertete Liste von allen Kandidaten als Ergebnis ausgegeben wird, wurde bezüglich der diagnostischen Auflösung der durchschnittliche Mittel-Rang [HW07] der eigentlichen Kandidaten gewählt.

Parameter-Einstellungen

Vor Durchführung der Diagnose-Experimente mussten zunächst die benötigten Parameter der Heuristik festgelegt werden, um die Defektgrößen-Quantisierung, die Umgebung zur Evaluierung der Ausgänge und die Gewichte zur Bewertung der Evidenzen zu bestimmen. Dabei wurden die folgenden Einstellungen gewählt:

- Defekt-Intervall Quantisierungsfaktor: $q = 20$.
- ω_U–Umgebung: $U = [T - 5\%, T + 2.5\%] = [0.95 \cdot T, 1.025 \cdot T]$.
- Gewichtung der Evidenz-Komponenten : $\omega_\sigma = 1.0$, $\omega_\iota = -0.35$.

Die Umgebung U ist **asymmetrisch** um den Abtastzeitpunkt T verteilt und *jenseits* von T stärker beschränkt. Hierbei wird das Wissen ausgenutzt, dass sich Ausgangssignale für $t > T$ stabilisieren und letztendlich einen *guten* Wert annehmen. Bei den Gewichten der Evidenz-Komponenten ist anfangs festgestellt worden, dass die Resultate der Diagnose bei Variation des ω_ι-Parameters *variieren*. In einem frühen Experiment mit *Haftfehlern* wurden bei den ISCAS '85 Schaltkreisen verschiedene Gewichtungen getestet von denen $\omega_\iota = -0.35$ im Durchschnitt die höchste Rate an Erst-Rängen erzielt hatte. Aus diesem Grund wurde die Einstellungen in den weiteren Experimenten beibehalten.

Schaltkreis	Erfolg[a]	Gruppe 1	Rang 1	Auflösung	δ-Abw.
c432	99.3%	96.6%	90.8%	1.3	±3.6%
c499	97.2%	94.9%	84.7%	1.4	±12.8%
c880	98.8%	97.0%	77.1%	1.5	±4.0%
c1355	98.9%	96.5%	83.0%	1.4	±7.4%
c1908	96.1%	87.0%	67.6%	2.0	±9.0%
c2670	93.3%	83.4%	67.2%	2.4	±7.1%
c3540	97.0%	94.8%	85.7%	1.5	±2.9%
c5315	96.7%	92.5%	86.4%	1.6	±3.4%
c6288	81.7%	74.9%	69.4%	3.2	±6.2%
c7552	97.0%	91.9%	84.1%	1.6	±3.5%
s35932	99.8%	98.2%	94.9%	1.1	±5.7%
s38417	99.1%	96.6%	84.1%	1.3	±4.1%
s38584	98.8%	94.1%	73.2%	1.6	±3.9%
∅	96.4%	92.2%	80.6%	1.7	±5.7%

Tabelle 4.3: Ergebnisse der ISCAS ’85 und ’89 Benchmark-Schaltkreise. – Für jeden Schaltkreis wurden die Antwortmuster von 1000 fehlerhaften DUDs untersucht.

[a] Mittel-Rang 1–10.

Verwendete Hardware

Sämtliche Experimente wurden auf modernen handelsüblichen Desktop-Computern durchgeführt, welche jeweils mit einem Intel Core i7 Prozessor, 8 GB RAM (max. 4 GB verwendet) und einer NVIDIA®CUDA™GTX 480 Grafikkarte mit 1,6 GB globalem Speicher ausgestattet waren.

4.2 Diagnose von Verzögerungsfehlern

In Tabelle 4.3 sind die Diagnose-Ergebnisse der ISCAS ’85 und ’89 Schaltkreise mit nominalen Gatterverzögerungen dargestellt. In der linken Spalte steht der Namen der diagnostizierten Schaltkreise. Spalte zwei zeigt die Prozentzahlen der erfolgreich diagnostizierten Fehler. Spalte drei und vier zeigen den Anteil der Versuche, bei denen der eigentliche Fehler in der ersten Rang-Gruppe bzw. auf dem ersten Mittel-Rang (und somit *perfekt*) diagnostiziert worden ist. Die fünfte Spalte gibt die diagnostische Auflösung wieder. In der letzten Spalte steht die durchschnittliche Abweichung der abgeschätzten Defektgröße.

Die durchschnittliche *Erfolgsrate* der ISCAS Schaltkreise lag mit Nominalzeiten bei 96,4% und in 80,6% der Fälle konnte der Fehler perfekt diagnostiziert werden. Die diagnostische *Auflösung* beträgt somit etwa 1,7 Kandidaten. Im Durchschnitt sind 38,8 Fehlerstellen nach

Schaltkreis	Erfolg	Gruppe 1	Rang 1	Auflösung	δ-Abw.
c499	100.0%	98.7%	89.0%	1.1	$\pm$7.8%
c880	99.8%	94.5%	80.1%	1.4	$\pm$2.8%
c1355	99.9%	95.0%	89.4%	1.1	$\pm$5.9%
c2670	97.8%	84.0%	78.2%	1.7	$\pm$2.3%
c7552	99.5%	93.4%	90.9%	1.2	$\pm$1.8%
s35932	100.0%	95.8%	94.6%	1.1	$\pm$3.5%
s38584	99.3%	91.5%	77.7%	1.5	$\pm$2.7%

Tabelle 4.4: Ausschnitt der Diagnoseergebnisse der ISCAS Schaltkreise bei vollständiger Simulation der quantisierten Defekt-Intervalle ($q = 20$) eines jeden Kandidaten.

dem Critical Path Tracing (CPT) übrig geblieben und mussten in der simulationsbasierten Analyse untersucht werden. Am schlechtesten schnitt der Schaltkreis c6288 ab, der aufgrund seiner vielen Verzweigungsstämme und Rekonvergenzen schwer zu diagnostizieren ist und im Mittel etwa 196 CPT Kandidaten produziert hat. Bei 92,2% der DUDs wurde der Fehler in der ersten Rang-Gruppe diagnostiziert, was vermuten lässt, dass die Fehlerabdeckung des verwendeten Testmustersatzes nicht hoch genug war, um die Kandidaten zu unterscheiden. Hier könnte man versuchen, mit Hilfe von ATPG-Tools (Automatic Test Pattern Generation) weitere Testmuster zu erzeugen, sodass die Kandidaten der Gruppe unterschieden werden.

Die *Abweichung* der abgeschätzten Defektgröße betrug im Schnitt $\pm 5,7\%$ und lag damit mehr als doppelt so hoch, als der *erwartete Mittelwert* von $\pm 2,5\%$, welcher durch den gewählten Quantisierungsfaktor $q = 20$ eingeführt wird. Die erhöhte Abweichung lässt sich zum einen dadurch erklären, dass durch das *Integral des Syndrom-Waveforms* die Defektgröße **überschätzt** wird. Zum anderen haben falsche Einstiegspunkte gelegentlich zu Ausreißern geführt, bei denen die Aussagekraft der erklärten Bits (Sigma) zu sehr von den Fehlvorhersagen (Iotas) unterdrückt wurde. Der Algorithmus hatte anschließend die Defektgröße bei der Korrektur immer weiter reduziert, um die Iotas zu verringern und so eine bessere Bewertung zu erlangen. Dies hatte letztlich zur Folge, dass der Defekt **unterschätzt** wurde.

Der Zeitaufwand der Diagnose eines defekten Schaltkreises reichte bei den ISCAS '85 von 0,7 (c432) bis 22,6 Sekunden (c6288) für die 2560 Verzögerungstests. Im Schnitt dauerten die Versuche 5,5 Sekunden pro DUD, während der Median bei 2,1 Sekunden lag. Die ISCAS '89 Schaltungen benötigten etwa 11,1 Sekunden Rechenzeit bei doppelter Anzahl der Tests.

Zum Vergleich wurden in Tabelle 4.4 ein paar Ergebnisse bei Simulation aller *quantisierten* Defektkandidaten dargestellt. Die Diagnose dieser Schaltkreise hatte dabei einen Simulationsaufwand von durchschnittlich 545 Simulationen pro DUD und lag damit etwa neun Mal höher als der vorgestellte *adaptive* Ansatz mit 62 Simulationen.

Schaltkreis	Erfolg	Gruppe 1	Rang 1	Auflösung	δ-Abw.
b17	95.2%	87.6%	60.8%	2.1	$\pm$3.6%
b20	98.0%	93.2%	80.6%	1.6	$\pm$2.4%
b21	97.7%	92.9%	78.1%	1.6	$\pm$2.3%
b22	96.5%	91.0%	77.9%	1.7	$\pm$2.1%
p35k	93.3%	89.0%	59.2%	2.4	$\pm$4.7%
p81k	97.9%	95.7%	81.5%	1.5	$\pm$4.2%
p89k	98.2%	90.8%	63.7%	1.8	$\pm$3.0%
p100k	96.0%	89.6%	81.2%	1.8	$\pm$2.6%
p141k	98.6%	95.5%	79.9%	1.5	$\pm$3.2%
p259k	95.8%	90.3%	80.8%	1.7	$\pm$1.7%
p269k	99.0%	95.3%	80.5%	1.4	$\pm$4.4%
p286k	98.3%	93.3%	77.9%	1.5	$\pm$2.5%
p378k	97.5%	93.5%	90.8%	1.5	$\pm$3.5%
p418k	96.5%	90.7%	75.8%	1.8	$\pm$1.7%
$\varnothing$	97.0%	92.0%	76.3%	1.7	$\pm$3.0%

Tabelle 4.5: Diagnose-Ergebnisse der ITC '99 und NXP Schaltkreise. – Demonstration an größeren Benchmark-Schaltkreisen.

Skalierbarkeit

Um die Brauchbarkeit des Diagnose-Algorithmus für größere Schaltkreise zu testen, wurden Experimente mit ITC '99 und industriellen Benchmark Schaltkreisen von NXP durchgeführt. Die Ergebnisse sind in Tabelle 4.5 zusammengefasst.

Beim Vergleich mit den ISCAS Schaltkreisen ist hier bei der diagnostischen Auflösung und den Erfolgen ein qualitativ *ähnliches* Verhalten zu erkennen. Viele Schaltkreise wiesen einen hohen Anteil an strukturellen Äquivalenzen auf, sodass die Kollabierung der Äquivalenzklassen einen nicht-vernachlässigbaren Teil der Kandidaten eliminieren konnte. Die durchschnittliche Anzahl an Kandidaten nach dem CPT betrug 40,7 Fehlerstellen. Zusätzlich wurden durch die Kollabierung auch die Rang-Gruppen verkleinert, wodurch bei den NXP Schaltkreisen die durchschnittliche Rate der Erst-Ränge um über 17 Prozentpunkte verbessert werden konnte. Die Zahl der in der ersten Rang-Gruppe diagnostizierten DUDs lässt zudem wieder vermuten, dass die Fehlerunterscheidung des Testmustersatzes zu gering war, um die Kandidaten auseinanderzuhalten. Hierfür müssten auch zusätzliche Muster generiert werden, sodass diese Gruppen weiter untersucht werden können. Die geringere Fehler-Abweichung kommt durch die höhere Tiefe der NXP Schaltkreise zustande. Je tiefer ein Schaltkreis ist, umso größer ist in der Regel auch das globale Defektgrößen-Intervall, da dieses von der Differenz der Slacks vom längsten und kürzesten Pfad im Schaltkreis abhängt. Da die Abweichung eines Defekts an einem Knoten relativ zu den globalen Grenzen gemessen wurde, fallen die *lokalen* Abweichungen beim Vergleich dadurch weniger ins Gewicht, weil die untersuchten Defektgrößen-Intervalle

der einzelnen Knoten verhältnismäßig kleiner sind. Dieses Phänomen war insbesondere bei den Schaltkreisen p259k und p418k zu beobachten, bei denen die durchschnittliche Abweichung weit unter dem Erwartungswert lag.

Bei den Rechenzeiten wurden für die Versuche der ITC '99 Schaltkreise mit 5120 Verzögerungstests durchschnittlich 24,5 Sekunden pro DUD gemessen. Bei NXP waren es 158,5 Sekunden.

4.2.1 Stabilität des Verfahrens

Bislang wurden bei der Diagnose nur Variationen *in* der Defektgröße betrachtet, die durch die in der Referenz-Fehlermaschine verwendete *quantisierte* Fehlergenerierung eingeführt werden und Unterschiede in den Testantworten verursachen können. Dabei sind in DUD und FM jeweils die selben Nominalschaltzeiten der Gatter verwendet worden. In der Realität entstehen bei der Herstellung der Integrierten Schaltkreise jedoch Variationen, die dafür sorgen, dass die genauen Schaltzeiten im DUD nicht mehr präzise vorhergesagt werden können und der Vergleich mit dem *exakten* Modell unrealistisch wird.

Um die Stabilität der Diagnose-Methode in solch einem Fall auch untersuchen zu können, wurden die Gatterschaltzeiten mit zufälligen Variationen versehen. Als Wahrscheinlichkeitsverteilung für die Zufallskomponente wurde hierbei die **Gaußsche Normalverteilung** $\mathcal{N}(\mu, \sigma^2)$ verwendet. Der Erwartungswert μ stellt hierbei die nominale Verzögerung eines Gatters dar. Aufgrund der hohen Abtastzeiten wurden 25% des Nominalwertes als **Standardabweichung** gewählt ($\sigma = 0.25 \cdot \mu$), sodass die Zeitbedingungen der Pfade im *fehlerfreien Fall* nur mit geringer Wahrscheinlichkeit verletzt werden. Solch eine Verletzung würde im DUD zu einem Mehrfachfehler im DUD führen, bei dem das implementierte CPT unter Umständen eine leere Kandidatenmenge zurückliefern könnte und der Diagnosealgorithmus terminiert. Tatsächlich ist dies hier jedoch in *keinem* der durchgeführten Experimente vorgekommen.

Die Gatterverzögerungen der DUDs wurden hier in Abhängigkeit der Instanz, sowie der Gatter-ID, bei jeder Simulation neu festgelegt. Die Defektgrößen der injizierten Fehler sind dabei *unabhängig* von der Variation an den Gattern und wurden nicht zusätzlich beeinflusst. Aufgrund der Variationen verändern sich die kumulativen Verzögerungen entlang der einzelnen Pfade durch die Schaltkreise. Dies hat Veränderungen in den Syndromen zur Folge, da manch fehlerhafter Pfad im Referenzmodell, wieder fehlerfrei wird bzw. ein fehlerfreier Pfad nun gegen die Zeitbedingungen verstoßen kann. Ebenso können Transitionen sichtbar werden, die vorher durch kontrollierte Werte oder Pulsfilterung *geblockt* waren und umgekehrt. Durch die Variationen wurden die Fehler in 731 der 13000 DUDs nicht mehr erkannt.

In Tabelle 4.6 sind die Diagnose-Ergebnisse der fehlerhaften ISCAS Schaltkreise mit Variation abgebildet. Hierbei gibt es bei den Erfolgen, sowie den Erst-Rängen und der diagnostischen Auflösung nur *kleine* Differenzen, woraus sich schließen lässt, dass sich die vorgestellte Methode **robust** in der Präsenz von Variationen verhält. Die Zahl der ersten Rang-Gruppen ist um fast

Schaltkreis	Erfolg	Gruppe 1	Rang 1	Auflösung	δ-Abw.
c432	98.5%	94.9%	88.6%	1.4	$\pm$4.5%
c499	96.2%	88.7%	82.7%	1.6	$\pm$11.1%
c880	98.3%	93.9%	76.8%	1.5	$\pm$4.6%
c1355	97.7%	91.9%	81.3%	1.5	$\pm$9.7%
c1908	95.6%	85.4%	67.4%	1.9	$\pm$9.4%
c2670	92.7%	79.9%	66.7%	2.5	$\pm$7.9%
c3540	96.7%	92.5%	86.0%	1.5	$\pm$3.4%
c5315	96.5%	89.8%	84.7%	1.6	$\pm$4.0%
c6288	83.2%	74.3%	69.8%	3.0	$\pm$5.7%
c7552	94.4%	87.6%	80.8%	1.9	$\pm$4.1%
s35932	99.0%	90.5%	88.1%	1.3	$\pm$6.2%
s38417	99.0%	94.0%	83.0%	1.4	$\pm$4.5%
s38584	98.6%	90.4%	71.1%	1.6	$\pm$3.9%
$\varnothing$	95.9%	88.7%	79.0%	1.8	$\pm$6.1%

Tabelle 4.6: Diagnose-Ergebnisse der ISCAS '85 und '89 Schaltkreise bei normalverteilter Variation der Gatterverzögerungen mit 25% Standardabweichung.

vier Prozentpunkte gesunken, was sich anhand der veränderten Syndrome erklären lässt. Hier haben sich bei der Auswertung in der Referenzsimulationen aufgrund der Änderungen in den Syndromen neue Sigmas, Iotas und Taus ergeben, sodass die Kandidaten der ersten Rang-Gruppe separiert werden konnten.

> **Anmerkung:** Die Abtastzeit der Schaltungen ist in den Experimenten sehr hoch gewählt worden. Für kleinere Abtastzeiten nahe der Schaltungsverzögerungen wird vermutet, dass sich zusätzliche Filter-Effekte ergeben, die die Fehlerpropagierung maskieren und somit zu weniger fehlerhaften Antwortbits führen.

Diagnose auf kompaktierten Antwortmustern

In einem letzten Experiment wurde die Diagnose auf kompaktierten Antwortmustern nach [HW09] beobachtet. Dazu sind die *realen Ausgänge RO* eines Schaltkreises in k gleich lange *Scan-Ketten* partitioniert worden. Diese wurden dann mit einem einfachen Paritätsbaum (XOR) verknüpft, der die aktuellen Werte einer jeden Scan-Kette (*Vektor*) zu einem Antwortbit zusammenfasst. Jedes dieser Bits repräsentiert einen *virtuellen Ausgang VO*, deren Anzahl durch die maximale Länge der Ketten definiert ist. Die maximale Länge aller Ketten gibt dann die Zahl der virtuellen Ausgänge an, von denen die komprimierten Testmuster abgelesen werden. Die Kompressionsrate R_c der kompaktierten Antwortmuster ist dabei durch den Quotienten $R_c = |RO|/|VO|$ bestimmt. Falls sich die Ausgänge gleichmäßig und restlos auf die k gleich langen Ketten verteilen lassen ist $R_c = k$.

Schaltkreis	R_c	Erfolg	Gruppe 1	Rang 1	Auflösung	δ-Abw.
c432	3.5	98.9%	95.7%	89.1%	1.3	±3.1%
c499	16.0	98.5%	94.9%	84.2%	1.4	±11.9%
c880	13.0	97.2%	93.8%	76.0%	1.6	±3.4%
c1355	16.0	97.9%	94.4%	80.1%	1.5	±8.3%
c1908	12.5	84.4%	75.3%	59.7%	3.0	±9.3%
c2670	15.6	93.1%	83.3%	67.3%	2.4	±6.5%
c3540	11.0	95.2%	91.9%	83.3%	1.7	±3.1%
c5315	15.4	95.4%	91.4%	85.7%	1.7	±3.4%
c6288	16.0	53.3%	44.4%	40.9%	6.0	±16.0%
c7552	15.4	89.7%	85.7%	74.4%	2.5	±6.0%
s35932	128.0	98.4%	98.0%	94.3%	1.2	±3.7%
s38417	124.0	96.6%	93.7%	81.9%	1.6	±3.3%
s38584	123.6	95.0%	90.0%	70.6%	2.0	±3.9%
∅		91.8%	87.1%	76.0%	2.2	±6.3%

Tabelle 4.7: Diagnose-Ergebnisse der ISCAS ’85 und ’89 Schaltkreise mit kompaktierten Antworten.

Tabelle 4.7 zeigt die Ergebnisse der Diagnose kompaktierter Antworten für die ISCAS Schaltkreise. Für die ’85er Serie wurde die Kompressionsrate so gewählt, dass die sich Anzahl der virtuellen Ausgänge für die meisten Schaltkreise auf zwei reduziert hatten. Bei c2670, c5315 und c7552 ist eine maximale Kompressionsrate von $R_c = 16$ verwendet worden. Bei den ISCAS ’89 Schaltkreisen wurde diese auf etwa 128 erhöht.

In den Experimenten ist dabei die durchschnittliche Zahl der möglichen Fehlerstellen nach dem Critical Path Tracing durch die Komprimierung auf 92,8 angestiegen. Das Gesamtbild der Ergebnisse ist jedoch *vergleichbar* mit der unkomprimierten Variante. Am schlechtesten schnitt wieder der c6288 Schaltkreis ab, bei dem aufgrund der vielen Rekonvergenzen und den zwei virtuellen Ausgängen durchschnittlich 571 Gatter untersucht werden mussten.

4.3 Evaluierung

In diesem Kapitel wurden die Ergebnisse der Versuche vorgestellt, die im Rahmen dieser Arbeit durchgeführt worden sind. Dabei wurde die Diagnostizierfähigkeit der präsentierten Methode für einzelne SMALLDELAY-Fehler beliebiger Größe sowohl in kleinen als auch in größeren industriellen Benchmark-Schaltkreisen demonstriert. Im Focus der Auswertung lagen neben der Lokalisierung der Defektstelle, auch die Abweichung der abgeschätzten Defektgröße der diagnostizierten Fehler.

Experiment	Fehler	Erfolg	Gruppe 1	Rang 1	Auflsg.	δ-Abw.
ISCAS '85	10000	95.6%	90.9%	79.6%	1.8	$\pm$6.0%
ISCAS '89	3000	99.2%	96.3%	84.1%	1.3	$\pm$4.6%
ITC '99	4000	96.9%	91.2%	74.4%	1.7	$\pm$2.6%
NXP	10000	97.1%	92.4%	77.1%	1.7	$\pm$3.1%
ISCAS Variation	12269	95.6%	88.7%	79.0%	1.8	$\pm$6.1%
ISCAS Kompaktierung	12610	91.8%	87.1%	76.0%	2.2	$\pm$6.3%
Gesamt	**51879**[a]	**95.3%**	**90.1%**	**78.0%**	**1.8**	$\pm$**5.2%**

Tabelle 4.8: Zusammenfassung der Ergebnisse aller durchgeführten Experimente. Die letzte Zeile betrachtet dabei die fehlerhaften DUDs aller Experimente im Ganzen.

[a] Erkannte Fehler von insgesamt 53000 DUDs.

Zur Übersicht wurden die Ergebnisse der jeweiligen Diagnose-Experimente in Tabelle 4.8 zusammengefasst.

Insgesamt konnten in den Experimenten 95,3% aller DUDs erfolgreich und 78,0% *perfekt* diagnostiziert werden. Mit der mittleren diagnostischen Auflösung von 1,8 befand sich der eigentliche Fehler somit für die meisten Fälle unter den ersten beiden Kandidaten. Dabei gilt im Allgemeinen: Je mehr Fehlerinformation ein Defekt für verschiedene Verzögerungstests produziert, umso leichter und besser lässt sich dieser diagnostizieren, da zum einen weniger Kandidaten nach dem Critical Path Tracing übrig bleiben und sich zum anderen die simulierten Fehler an der eigentliche Defektstelle durch stärkere Evidenzen behaupten können.

Mit Hilfe von normalverteilter Variation der Gatterschaltzeiten wurde das reale Verhalten einer Menge von Schaltkreisen nachgeahmt, wodurch sich die einzelnen DUDs vom Referenzmodell distanziert haben. Hierbei wurde die Stabilität des Diagnoseverfahrens gegenüber den Variationen im Schaltkreis demonstriert.

In einem letzten Experiment wurde noch ein Anwendungsfall untersucht, bei dem direkte Diagnose auf mit Hilfe von Paritätsbäumen kompaktierte Antwortmuster angewendet wurde.

5 Zusammenfassung

Heutzutage ist Fehlerdiagnose während der Entwicklung von integrierten Schaltkreisen unverzichtbar, um die Ausbeute und Qualität der Chips in der Produktion zu verbessern (*Yield-Ramping*) und die kumulativen Kosten pro hergestellten Chip zu senken.

Das Ziel dieser Arbeit war der Entwurf einer Diagnose-Methode von kleinsten Verzögerungsfehlern, die Defekte unter einer großen Mengen von möglichen Kandidaten ausfindig machen kann. Die präsentierte Methode kombiniert die Vorteile der Effect-Cause und Cause-Effect Paradigmen und greift auf einen effizienten GPGPU beschleunigten Zeitsimulator zurück, mit dem das Zeitverhalten in den Schaltkreisen präzise ausgewertet werden kann. Ein neuartiger simulationsbasierter Ansatz untersucht dabei nur einen Bruchteil aller möglichen Fehler, um den Gesamtaufwand der Diagnose zu reduzieren. Bei der Evaluierung von Fehlern mit dem Simulator werden die Signalverläufe der Schaltungsausgänge in einem Zeitraum um den Abtastzeitpunkt betrachtet, sodass die Fehlerwahrscheinlichkeiten einzelner Ausgänge bestimmt und zur Bewertung der Fehlerkandidaten verwendet werden können.

Experimente mit zufälligen Testmustern hatten gezeigt, dass der Diagnose-Algorithmus für kleine und größere Schaltkreise anwendbar ist und gute Ergebnisse auch im Fall von Variationen liefert. In den Versuchen wurde die eigentliche Defektstelle dabei im Schnitt unter den ersten beiden gemeldeten Kandidaten gefunden, bei einer geringen durchschnittlichen Abweichung der abgeschätzten Defektgröße.

Da Verzögerungsdefekte aufgrund von schrumpfenden Herstellungsprozessen und den steigenden Anforderungen bezüglich der Performanz der Schaltungen unvermeidbar sind und immer mehr zum Problem werden, ist die weitere Forschung an Diagnose-Methoden von essentieller Bedeutung.

Eine Möglichkeit zur Verbesserung der Präzision wäre hierbei ein komplexeres Auswertungsverfahren der Syndrome anhand der Waveforms im Simulator. Des Weiteren könnte die Generierung zusätzlicher Verzögerungstests mit ATPG zur Erhöhung der Fehlerabdeckung der verwendeten Tests, sowie zur weiteren Unterscheidung von Kandidaten einer Rang-Gruppe behilflich sein. Ebenfalls könnte man untersuchen, inwiefern sich mit Hilfe von maschinellem Lernen die Symptome klassifizieren lassen, um die diagnostische Auflösung durch Manipulation der Parameterkonfigurationen mit geeigneten Trainingsdaten zu verbessern. Des Weiteren könnte die Diagnostizierbarkeit von weiteren bzw. allgemeineren Fehlermodellen untersucht werden.

A ADAMA `dfdiagnose`

Das Diagnoseverfahren wurde in das vorhandene **ADAMA-Framework (Adaptive Diagnosis of Arbitrary Manifold Artifacts)** implementiert und benötigt zur Ausführung die Angabe des zu diagnostizierenden Schaltkreises. Sei *Model.lg* die Beschreibung, des zu diagnostizierenden Schaltkreises, dann wird das Basisaufruf der Diagnose folgendermaßen ausgeführt:

```
> adama dfdiagnose <Model.lg> {Optionen}
```

Ferner muss die Angabe ggf. mit dem absoluten oder relativen Pfad erweitert werden, falls sich der Schaltkreis nicht im aktuellen Verzeichnis befindet. Ohne die Angabe weiterer Optionen wird nach Aufruf zunächst die vollständige Menge der SMALLDELAY-Fehler des Schaltkreises generiert (nach [Sch11]) und anschließend nacheinander in DUDs injiziert, simuliert und unter Verwendung der Standardparameter-Einstellungen diagnostiziert.

Es folgt eine Liste aller Optionen und Parametern (Die Standardeinstellungen sind dahinter in Klammern angegeben):

-k <*Datei*>: *Kernel*. Absoluter/relativer Pfad des auszuführenden Zeit-Simulators (Standard java).

-d <*Wert*>: *Device*. Spezifiziert die ID der zu benutzenden CUDA-Grafikkarte (Standard 0).

-m <*Wert*>: *Memory*. Legt den maximalen globalen Speicher auf der GPGPU fest, der für den Waveform-Speicher reserviert wird (100).

-wc <*Wert*>: *Waveform capacity*. Waveform-Kapazität eines Basisregisters (34).

-i <*Wert*>: *Initial wave-memory*. Anteil des Waveform-Speichers der für die initialen Register reserviert wird. Der restliche Teil wird für Rekalibrierungen freigehalten (0.5).

-r <*Wert*>: *Random patterns*. Anzahl der Zufallsmuster. Da der Simulator Musterblöcke zu je 64 Mustern bearbeitet, wird dieser Wert auf den nächsthöheren durch 64 teilbaren Wert aufgerundet (Standardwert 64).

-p <*Datei*>: *Deterministic patterns*. Pfad zu Datei mit existiereneden deterministischen Testmustern.

-sdf <*Datei*>: Pfad zu Datei mit gespeicherten Nominalverzögerungen der Gatter $*.sdf$-Format.

-c <*Kompaktor-Spec*>: *Compactor mode*. Kompaktierung der Antwortmuster nach einer bestimmten Kompaktor-Spezifikation.

-cs <*Datei*>: *Cached sizes*. Initialisiert Waveform-Register mit gespeicherten Größen aus einer Datei.

-cu: *Cache update*. Aktualisiert gespeicherte Registergrößen in der Datei, falls diese während der Simulation verändert worden sind.

-fl <*Datei*>: *Fault-list*. Spezifiziert Name und Pfad einer existierenden Fehler-Liste für die DUD Defekte.

-cl <*Wert*>: *Crop list*. Nachbearbeitung einer Fehler-Liste: Beschränkung auf die ersten X Fehler mit `-cl X`, Beschränkung auf X zufällige Fehler mit `-cl rX` und Beschränkung der Liste auf ein Intervall von Fehler X bis einschliesslich Fehler Y `-cl X:Y`.

-fm <*Wert*>: *Fault model*. Vollständige Berechnung von Fehlern aus einem Fehlermodell nach [Sch11] zur DUD Defekt Generierung. Jeder Fehler wird in ein eigenständiges DUD injiziert und diagnostiziert. Wird eine Fehlerliste mit `-fl` übergeben, so ist diese Option *obsolet*.

-var <*Wert*>: Spezifiziert die prozentuale Standardabweichung aller nominalen Gatterverzögerungen im DUD. Der Simulator verwendet zufällige Verzögerungen mit Gaußscher Normalverteilung.

-q: Faktor zur Quantisierung der Defektgrößen im Referenzfehlermodell.

-all: Forciere vollständige Simulation aller CPT-Fehlerkandidaten.

-norep: Unterbindet Kollabierung der Fehler-Äquivalenzklassen

-wiota <*Wert*>: Gewicht der Iota Evidenz-Komponente (Standard -0.35).

Beispiel

Beispielbefehl für die Diagnose der ersten 1000 in einer Liste abgespeicherten Fehler des Schaltkreises c1908 mit 2560 pseudozufälligen Verzögerungstests und einer Defektintervall-Quantisierung von $q = 20$ des Referenzmodells:

```
> adama dfdiagnose c1908.lg -fl fehlerliste.fl -cl 1000 -r 5120 -q 20
```

Listing A.1 Beispielausgabe des Diagnoseprogramms.

```
0000486.323 [--] Defect 43 of 1000: SDF { NAND_2_481/0 92.0 } (46.341465% local, 42.424244% global)
0000486.323 [DD] ClassRepresentative SDF { NAND_2_481/0 92.0 } (1 faults)
0000486.323 [DD] DUD Simulation...
0000486.441 [--] Structural Analysis // Auswertung, CPT und Defektgroessenbestimmung
0000486.558 [--]   FailingBlocks 40
0000486.558 [--]   FailingPatterns 57
0000486.558 [--]   FailingBits 98
0000486.558 [--]   FailingVirtualOutputs 4
0000486.579 [DD] Elapsed Time
0000486.579 [DD]   6Valued Simulation 3 ms
0000486.579 [DD]   Critical Path Tracing 4 ms
0000486.579 [DD]   Lower Bound Estimation 21 ms
0000486.579 [DD]     Model [81.0 : 130.0] (units)
0000486.579 [DD]   Est. Size: 135.0 (units) // abgeschaetzte Defektgroesse
0000486.579 [DD]     OVERESTIMATED
0000486.579 [DD] CPT Target fault 98/98 traces (17 candidates): OK
0000486.580 [--] Simulation-based Analysis
0000486.580 [--]   Removed PIs/POs: 0 suspect(s) removed
0000486.580 [--]   Collapsed List: 1 suspect(s) removed
0000486.581 [--] Building Initial SDF Candidate Set...
0000486.581 [DD]   Using estimated LB: 130.0 (units)
0000486.581 [DD]   Initial Suspect List: 16 of 320 faults
0000486.581 [--] Horizontal Scan...
0000488.876 [--] Vertical Scan...
0000488.876 [--]   Entry Point NAND_2_651 scanned 12 times // Defekt-Intervall Quantisierung
0000488.876 [--]     Node Range [64.0:104.0]
0000489.947 [--]   Adjusted DefectSize 94.0 (units) // korrigierte Defektgroesse
0000489.947 [--] Horizontal Scan...
0000491.176 [DD]   Evidence Updates 15
0000491.176 [--] Total Simulations 43
0000491.176 [--] --- Suspect Evaluation ---
0000491.176 [--] Group 1 ... 1 : *SDF { NAND_2_481/0 94.0 }
0000491.176 [--] ----------
0000491.176 [--] ActualRank 1
0000491.177 [--] MiddleRank 1 // Diagnostizierter Mittelrang des eigentlichen Kandidaten
0000491.177 [--] RankGroup 1
0000491.177 [--] GroupSize 1
0000491.177 [--] BestEvidence SDF { NAND_2_481/0 94.0 }: sig 96.790115 (98) iot 3.4320989 (5) with
    score 95.58888
0000491.177 [--] ActualEvidence SDF { NAND_2_481/0 94.0 }: sig 96.790115 (98) iot 3.4320989 (5) with
    score 95.58888
0000491.177 [DD] Top-5 Suspects
0000491.177 [DD]   1. gamma 0 sigma 96.790115 (98) iota 3.4320989 (5) tau 0 (SDF { NAND_2_481/0 94.0 })
0000491.177 [DD]   2. gamma 12 sigma 40.901234 (42) iota 51.79012 (53) tau 56 (SDF { NAND_2_651/0 94.0
    })
0000491.177 [DD]   3. gamma 3 sigma 34.814816 (36) iota 34.87654 (40) tau 62 (SDF { NAND_2_531/0 94.0 })
0000491.177 [DD]   4. gamma 6 sigma 33.25926 (34) iota 35.864197 (37) tau 64 (SDF { BUF_1_670/0 94.0 })
0000491.177 [DD]   5. gamma 6 sigma 33.25926 (34) iota 35.864197 (37) tau 64 (SDF { NAND_2_659/0 94.0 })
0000491.177 [DD] Defect Size Deviation 4.878047% (local), 3.030303% (global) // Fehler der
    abgeschaetzten Groesse
```

B Notizen zur Implementierung

Mehrwertige Logiksimulation

Die Implementierung der sechswertigen Logik $\mathbb{H}_6 = \{\mathrm{C0}, \mathrm{C1}, \mathrm{T0}, \mathrm{T1}, \mathrm{H0}, \mathrm{H1}\}$ basiert auf dem Konstruktionsverfahren von [Hay86]. Dabei wird die interne Repräsentation der Logiksymbole mit Hilfe eines 3-Tupels realisiert. Hayes verwendet dazu je einen $\mathbb{B}_2$-Unterraum für Initial- und Finalwert, sowie dreiwertige Logikkomponente aus $\mathbb{E}_3 = \{0, 1, X\}$ zur Beschreibung des Zustands in der transienten Region, um Kompatibilität mit der komponentenweisen Anwendung der Logikoperatoren zu ermöglichen. In dieser Arbeit wurde jedoch anstatt des Signalzustands der Übergangsregion, eine **Transitionsmarkierung** $T \in \mathbb{B}_2$ verwendet, welche angibt, ob Signalwechsel innerhalb der Übergangsregion stattfinden (T) oder das Signal konstant ist ($\overline{T}$). Initialwert (I) und Finalwert (S) der Knoten sind dabei nach wie vor kompatibel mit Boolescher Logik. Die Transitionsmarkierung (T) muss mit Hilfe einer **Ereignispropagierungsfunktion** f_T berechnet werden.

Sei die Tupel-Repräsentation eines Symbols $w \in \mathbb{H}_6$ wie folgt notiert:

$$w = (I \times S \times T) = \left\{ \begin{bmatrix} I \\ S \end{bmatrix}, \begin{bmatrix} T \end{bmatrix} \right\} = \begin{bmatrix} I & S \\ - & T \end{bmatrix} \in \mathbb{H}_6.$$

Zur Berechnung eines Logikoperators $\circ$ für zwei beliebige Elemente $w_1, w_2 \in \mathbb{H}_6$ wird eine Funktion f_T zur Propagierung von Transitionen definiert werden, sodass gilt:

$$w_1 \circ w_2 = \begin{bmatrix} I_1 \circ I_2 & S_1 \circ S_2 \\ - & f_T(\circ, w_1, w_2) \end{bmatrix} \in \mathbb{H}_6.$$

Die Funktion f_T verwendet dabei das Wissen über die kontrollierenden Werte der Gatterfunktionen $\circ$ und wurde nach folgender Regel aufgestellt:

> **Regel 1.** Ein Knoten propagiert ein Ereignis (T) genau dann, wenn mindestens ein direkter Vorgänger ein Ereignis besitzt, und alle anderen Vorgänger **keinen** konstanten *kontrollierenden* Zustand (C0, C1) aufweisen.

Beispiel B.1. *(Ereignispropagierung) Betrachte man ein NAND-Gatter mit Eingangssignalen A und B und Ausgangssignal Z. Sei $w_A = T1$ und $w_B = T0$. Da Z von keinem Eingang aus konstant kontrolliert wird und mindestens eine Transition stattfindet, wird ein Ereignis propagiert:*

$$w_Z = w_A \overline{\wedge} w_B = \begin{bmatrix} 0 & 1 \\ - & 1 \end{bmatrix} \overline{\wedge} \begin{bmatrix} 1 & 0 \\ - & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ - & 1 \end{bmatrix} = H1.$$

Sei nun $w_B = C0$ ein konstantes kontrollierendes Signal, dann wird die Ereignispropagierung von A durch Z gestoppt, da der Knoten nicht sensibilisiert ist:

$$w_Z = w_A \overline{\wedge} w_B = \begin{bmatrix} 0 & 1 \\ - & 1 \end{bmatrix} \overline{\wedge} \begin{bmatrix} 0 & 0 \\ - & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ - & 0 \end{bmatrix} = C1.$$

Da jedes der verwendeten Symbole S, I und T mit einem einzelnen Bit repräsentiert werden kann, benötigt diese Implementierung der sechswertigen Logik im Vergleich zu [Hay86] nur drei statt $\lceil \log_2(|\mathbb{B}_2|) \rceil + \lceil \log_2(|\mathbb{E}_3|) \rceil + \lceil \log_2(|\mathbb{B}_2|) \rceil = 1 + 2 + 1 = 4$ Bits.

Um die Auswirkungen von Verzögerungsfehlern im Schaltkreis genauer untersuchen zu können, werden die Elemente der $\mathbb{H}_6$ Algebra nun mit einer **Defektmarkierung** erweitert. Da Verzögerungsfehler rein dynamischer Natur sind und die Fehlereffekte nur bei Knoten mit Transitionen auftreten können, werden die Defektmarkierungen auf Elemente mit Ereignissen beschränkt und bei konstanten Signalen (C0, C1) ignoriert. Jeder Zustand mit Ereignis kann somit entweder als **fehlerhaft** (D) oder **fehlerfrei** ($\overline{D}$) angesehen werden. Ein Signal wird dabei als *fehlerhaft* bezeichnet, wenn seine Stabilisierungszeit gegen die Zeitbedingungen verstoßen *kann*, während bei *fehlerfreien* Signalen dagegen *sichergestellt* wird, dass die Nominalzeiten unter allen Umständen eingehalten werden. Als Konsequenz erweitert sich die Grundmenge der Algebra um weitere vier – auf insgesamt zehn – Wertelemente und wird im Folgenden mit dem Symbol $\mathbb{W}$ notiert:

$$\mathbb{W} = \{\mathrm{C0}, \mathrm{C1}, \mathrm{T0}, \mathrm{T1}, \mathrm{H0}, \mathrm{H1}, \} \cup \{\mathrm{D0}, \mathrm{D1}, \mathrm{X0}, \mathrm{X1}\}$$

Zur Veranschaulichung und zum besseren Verständnis wurden die Logiksymbole in Tabelle. B.1 zusammengefasst. Innerhalb der grau angedeuteten Bereiche des Signalverlaufs findet der Schaltvorgang statt und es wird ein beliebiger Wert angenommen (Unbekannt ‚X').

Die interne Repräsentation dieser zehn Zustände benötigt mindestens $\lceil \log_2 10 \rceil = 4$ Bits. Zwar lassen sich damit bis zu $2^4 = 16$ verschiedene Symbole kodieren, die restlichen Kombinationen werden im Rahmen dieser Arbeit jedoch nicht verwendet. Die Kodierung der Logiksymbole wurde bei der Erweiterung so gewählt, dass Transitions- und Defektmarkierungen jeweils durch eigenständige Bits repräsentiert werden.

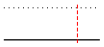
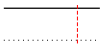
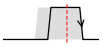
fehlerfrei		fehlerhaft	
Symbol	**Signal**	**Symbol**	**Signal**
C0		–	
C1		–	
T0		D0	
T1		D1	
H0		X0	
H1		X1	

Tabelle B.1: Auflistung aller Symbole der zehnwertigen Algebra. – Die graue Fläche markiert den *Übergangsbereich*, der senkrechte Strich repräsentiert den Signal-Abtastzeitpunkt in Relation zur Stabilisierungszeit.

Die Tupelrepräsentation eines Symbols in der $\mathbb{W}$-Algebra sieht somit nun wie folgt aus:

$$w = (I \times S \times T \times D) = \left\{ \begin{bmatrix} I \\ S \end{bmatrix}, \begin{bmatrix} T \end{bmatrix}, \begin{bmatrix} D \end{bmatrix} \right\} = \begin{bmatrix} I & S \\ D & T \end{bmatrix} \in \mathbb{W}.$$

Die Rechenregeln der erweiterten Algebra für die Komponenten I, S und T sind unabhängig von D und werden wie gehabt übernommen. Bei der Berechnung der Defektmarkierung im Falle einer zehnwertigen Simulation ist zu beachten, dass Fehlereffekte am Eingang eines Gatters durch Seiten-Pfade (Off-Path) maskiert werden können und die Weiterleitung gestoppt wird. Anders als bei der konservativen Transitionsberechnung, kann dies auch in Gegenwart von Ereignissen an Seiten-Pfaden geschehen, sofern diese den Knoten dominieren. In diesem Fall würde zwar die Transitionsmarkierung propagiert werden, nicht aber der Fehlereffekt.

Simulation

Implementiert wurden die Zustände eines Knoten mit 64-Bit Vektoren (Java `Long`-Typ) zur vektorweisen Bearbeitung im Stil des **Parallel Pattern Single Fault Propagation (PPSFP)** Paradigmas. Der Zustand eines Signals A wird dabei mit jeweils zwei Vektoren A_V und A_T repräsentiert. Der erste Vektor $A_V = \{S_{31}I_{31}...S_1I_1S_0I_0\}$ hält an 64 Bit-Positionen abwechselnd die I- und S-Bits der einzelnen Signalverläufe und kann somit die Information von bis zu 32 verschiedenen Verzögerungstests tragen. S_{31} befindet sich dabei an der höchstwertigen Bitposition

Algorithmus B.1 Evaluierung eines Gatters in sechswertiger Logik (PPSFP).

```
1:  procedure CALC(n)                    // Knoten mit Eingangssignalen A, B und Ausgang Z
2:      Berechne Z_V = A_V ∘ B_V (Boolesche Logiksimulation).
3:      Setze Z_T auf 0 zurück.
4:      /* Ereignispropagierungsfunktion f_T */
5:      if TYP(n) ∈ {AND, NAND} then          // Nicht-kontrollierender Wert = 1
6:          Z_T ← (A_T ∧ B_T) ∨ (A_T ∧ (B_V ∨ (B_V ≪ 1))) ∨ (B_T ∧ (A_V ∨ (A_V ≪ 1)))
7:      else if TYP(n) ∈ {OR, NOR} then       // Nicht-kontrollierender Wert = 0
8:          Z_T ← (A_T ∧ B_T) ∨ (A_T ∧ (¬B_V ∨ (¬B_V ≪ 1))) ∨ (B_T ∧ (¬A_V ∨ (¬A_V ≪ 1)))
9:      else if TYP(n) ∈ {XOR, XNOR} then
10:         Z_T ← (A_T ∨ B_T)
11:     else                                  // INV, BUF und OUTPUT
12:         Z_T ← A_T
13:     end if
14:     Setze D-Bits von Z_T zurück auf 0.
15: end procedure
```

des Vektors. Ein zweiter Vektor $A_T = \{T_{31}D_{31}...T_1D_1T_0D_0\}$ hält die Transitions- und Defektmarkierungen der Zustände fest. Der Zustand eines Schaltungseingangs A wird für beliebige Zuweisungen in einem Verzögerungstest aus Initialisierungs- (IV) und Propagierungswerten (PV) nun wie folgt bestimmt:

Sei $\pi = \{PV_{31}IV_{31}...PV_1IV_1PV_0IV_0\}$, dann setze die Initial- und Finalwerte des Knoten auf $A_V := \pi$. Um die durch den Test erzeugten **initialen Transitionen** zu übernehmen, muss zusätzlich $I_T := (I_V \oplus (I_V \ll {}^{1}1)) \wedge$ 0xAAAA ... AA_{16} gesetzt werden. Durch die bitweise XOR-Verknüpfung wird bei einem Signalwechsel von I nach S automatisch die Transitionsmarkierung erzeugt. Die Maskierung am Ende soll lediglich dazu dienen, um die durch den Shift-Operator entstandenen Defekt-Markierungen zu löschen. Nach Zuweisung aller Eingangssignale wird der Rest der Schaltung durch Auswerten der Gattersignale in topologisch sortierter Reihenfolge simuliert.

Bei der Simulation können die Berechnungen der S, I, T und D Markierungen getrennt durchgeführt werden. Zu beachten sind dabei die Datenabhängigkeiten der einzelnen Komponenten, weshalb zunächst die S und I Bits, dann T und zum Schluss D berechnet werden müssen. Algorithmus B.1 zeigt die Evaluierung eines einzelnen Gatter in sechswertiger Logik. Die Prozedur verwendet sowohl Boolesche Logiksimulation, als auch die Transitionspropagierung f_T. In einem Schaltkreis werden alle Gatter in topologisch sortierter Reihenfolge, beginnend ab den Eingängen, in Richtung der Ausgänge ausgewertet, um die Datenabhängigkeiten der einzelnen Knoten zu berücksichtigen.

[1] Bitweiser Schiebe-Operator (Bit-Shift) nach links.

Critical Path Tracing

Damit das Critical Path Tracing für einen fehlerhaften Musterblock des DUDs ausgeführt werden kann, muss dieser zunächst in der mehrwertigen Logik simuliert werden, um die internen Signale aller Knoten in der Schaltung zu initialisieren. Die Simulationsergebnisse werden dann mit den DUD Antworten verglichen, sodass alle fehlerhafte Ausgänge identifiziert werden können. Der folgende Algorithmus B.2 zeigt die implementierte Prozedur zur Rückverfolgung der fehlerhaften Ausgänge eines Musterblocks.

Algorithmus B.2 Critical Path Tracing für einen Test (PPSFP).

```
 1: procedure CPT(G, π)                                  // Schaltung G, Testantwort π.
 2:     /* Voraussetzung: Initialisierte Zustände aller Knoten durch Gutsimulation. */
 3:     for all o ∈ Ausgänge do
 4:         fehler ← (π[o] ⊕ o_V) ∧ 0xAAAA ... AA_16      // Berechne Syndrom von Ausgang.
 5:         if fehler ≠ 0 then
 6:             Lösche D-Markierungen von allen Knoten.
 7:             o_T ← o_T ∨ (fehler ⋙ 1)                  // Setze D von Ausgang.
 8:             for all Gatter n ∈ V in umgekehrter topologischer Reihenfolge do
 9:                 if MARKIERT(n) ≠ 0 then
10:                     BACKTRACE(n)
11:                     Aktualisiere Backtrace-Zähler von Knoten n.
12:                 end if
13:             end for
14:         end if
15:     end for
16: end procedure
```

Bei einem zu untersuchenden Ausgang werden zunächst alle noch vorhandenen Defektmarkierungen früherer CPT Iterationen im Schaltkreis gelöscht, und die des fehlerhaften Ausgangs mit dem Syndrom initialisiert. Anschließend wird in umgekehrt topologisch sortierter Reihenfolge für alle Gatter eine BACKTRACE-Funktion (Alg. B.3) aufgerufen, bei der versucht wird, die an einem Knoten vorhandene Defektmarkierung an seine Eingangsknoten weiterzuleiten.

Um festzustellen, ob ein Knoten für die Testmuster eines Musterblocks mit einer Defektmarke oder einer Transitionsmarke versehen ist, wird sein Zustandsvektor Z_T entsprechend maskiert. Dazu werden die folgenden Primitive verwendet:

- Überprüfe Defekt: MARKIERT$(Z) := Z_T \wedge$ 0x5555 ... 55_{16}
- Überprüfe Ereignis: EREIGNIS$(Z) := (Z_T \ggg {}^2 1) \wedge$ 0x5555 ... 55_{16}

[2] Bitweiser Schiebe-Operator (Bit-Shift) nach rechts.

Algorithmus B.3 Rückpropagierung der D-Markierung (PPSFP).

```
 1: procedure BACKTRACE(n)            // Knoten mit Eingangssignalen A, B und Ausgang Z
 2:     obs ← EREIGNIS(Z) ∧ MARKIERT(Z)
 3:     if TYP(n) ∈ {AND, NAND, OR, NOR} then
 4:         A_T ← A_T ∨ (obs ∧ EREIGNIS(A) ∧ (CTRLS(A, n) ∨ ¬CTRLD(n)))
 5:         B_T ← B_T ∨ (obs ∧ EREIGNIS(B) ∧ (CTRLS(B, n) ∨ ¬CTRLD(n)))
 6:     else if TYP(n) ∈ {XOR, XNOR} then
 7:         A_T ← A_T ∨ (obs ∧ EREIGNIS(A))
 8:         B_T ← B_T ∨ (obs ∧ EREIGNIS(B))
 9:     else                                          // INV, BUF und OUTPUT
10:         A_T ← A_T ∨ obs
11:     end if
12: end procedure
```

Die in Algorithmus B.3 vorgestellte BACKTRACE-Prozedur verwendet des Weiteren zwei Funktionen zur Bestimmung der *Kontrollierbarkeit* (CTRLS(A,n)) und *Kontrolliertheit* (CTRLD(n)) eines Knotens n.

Dabei untersucht CTRLS(A,n), ob der Finalwert des Eingangs A in Abhängigkeit des untersuchten Gattertyps von n einen **kontrollierenden** Wert aufweist:

$$\text{CTRLS}(A,n) := \text{0x5555 ... 55}_{16} \wedge \begin{cases} \overline{(A_V \ggg 1)} & \text{If TYP}(n) \in \{\text{AND, NAND}\}, \\ (A_V \ggg 1) & \text{else if TYP}(n) \in \{\text{OR, NOR}\}, \\ -1 & \text{else.} \end{cases}$$

In CTRLD(n) wird bestimmt, ob der Ausgang eines Knotens einen **kontrollierten** Wert besitzt, wodurch man folgern kann, dass mindestens ein Eingang einen kontrollierenden Wert hat:

$$\text{CTRLD}(n) := \text{0x5555 ... 55}_{16} \wedge \begin{cases} \overline{(Z_V \ggg 1)} & \text{If TYP}(n) \in \{\text{AND, NOR}\}, \\ (Z_V \ggg 1) & \text{else if TYP}(n) \in \{\text{NAND, OR}\}, \\ -1 & \text{else.} \end{cases}$$

Um die Schnittmenge der Kandidaten einer jeden Rückverfolgung zu berechnen wird für jeden Knoten $n \in V$ ein **Zähler** $count[n]$ verwendet, der die Anzahl der Rückverfolgungen für den Knoten speichert [TBT98, BGPV10]. Diese werden zu Beginn der Untersuchung auf *null* gesetzt. Der Zähler eines Knoten wird im Anschluss an die D-Propagierung mit der BACKTRACE-Prozedur aktualisiert. Dabei wird der Wert eines Knoten n um die Anzahl der D-Markierungen in seinem Zustandsvektor Z_T erhöht, welche mit Hilfe einer schnellen Methode zum Zählen

der Einsen in einem 64-Bit Vektor berechnet wird. Diese ist bereits in der Java Long-Klasse enthalten:

$$count[n] := count[n] + \text{Long.BITCOUNT}((Z_T \wedge \text{0xAAAA ... AA}_{16})).$$

Nach der Evaluierung aller Musterblöcke wird die initiale Kandidatenmenge S anhand der Zähler und der Gesamtzahl FB der fehlerhaften Antwortbits des DUDs bestimmt. Da im Einzelfehlerfall die Defektursache von jedem fehlerhaften Ausgang zurückverfolgt werden können *muss*, berechnen sich die Kandidaten wie folgt:

$$S := \{n \in V : count[n] == FB\}.$$

Simulationsbasierte Analyse

Bei der Defektgrößenabschätzung und der simulationsbasierten Analyse wird auf einen GPGPU-beschleunigten Fehlersimulator für SMALLDELAYs aus [Sch11] zurückgegriffen.

GPGPU-gestützte Berechnung des Syndroms

Die Auswertung der Referenzfehler mit den Integralen benötigt die Signalverläufe der Ausgänge, welche sich nach der Simulation der Verzögerungstests im globalem Speicher der GPGPU befinden. Für die *Waveform*-Signale werden jedoch im Durchschnitt über 90% des globalen Speichers der GPGPU allokiert. Da der Transfer dieses Speichers viel Zeit kostet, würde dies die Diagnose durch die nötige Kommunikation und Synchronisation stark verlangsamen. Des Weiteren benötigt die serielle Auswertung auf der Host-CPU zusätzliche Rechenzeit. Aus diesen Gründen wurde die Berechnung der Syndrom-Integrale als CUDA-Kernel implementiert, damit diese auf der GPGPU parallelisiert ausgeführt werden kann. Zudem müssen dann — dank der berechneten Integralwerte — nicht mehr die vollständigen *Waveform*-Signale der Ausgänge von der GPGPU kopiert werden, wodurch die transferierte Datenmenge stark reduziert wird.

Sei signal(Z) der Signalverlauf des zu untersuchenden Ausgangs Z und sei $U = [U_{min}, U_{max}] \in \mathbb{R}_0^+$ der zu integrierende Zeitbereich in der Zeitdomäne, dann ist die Fläche des Signalverlaufs gegeben durch

$$\text{Fläche}(Z) = \int_U \text{signal}_Z(t)\ dt.$$

Mit Hilfe des folgenden einfachen Algorithmus B.4 können die Signalverläufe in U partiell integriert und somit die Fläche berechnet werden. Da die Transitionszeiten der einzelnen *Waveforms* auf der GPGPU gemäß Definition in absteigender Reihenfolge gespeichert sind, werden diese in umgekehrter Reihenfolge evaluiert, wodurch die einzelnen Flächenstücke

Algorithmus B.4 Waveform Integral

```
1:  function WaveInt(W, U_min, U_max)
2:      /* Anmerkung: Auswertung der Waveforms geschieht in umgekehrter Reihenfolge. */
3:      Summe ← 0
4:      letzter ← U_max                                   // Start für partielles Integral.
5:      i ← max({j : t_j ∈ W, t_j ≤ U_max})               // Index der letzten Transition t ≤ U_max
6:      while (t_i ≥ U_min) ∧ (i ≥ 0) do
7:          if Transition t_i ist steigend then
8:              Summe ← Summe + (letzter − t_i)           // Fläche zwischen t_{i+1} und t_i.
9:          end if
10:         letzter ← t_i
11:         i ← i − 1
12:     end while
13:     if (i ≥ 0) ∧ (Transition t_i steigend) then
14:         Summe ← Summe + (letzter − U_min)             // Schließe Waveform, falls nötig.
15:     end if
16:     return Summe
17: end function
```

jeweils bei fallenden Flanken beginnen und bei den darauffolgenden steigenden Flanken wieder geschlossen werden.

Um den Algorithmus für das Waveform-Integral auf das Syndrom anzuwenden, wird die Polarität des Signalverlaufs eines jeden Ausgangs in Abhängigkeit seines stabilisierten Werts $\lim_{t\to\infty} \mathit{signal(t)} \in \{0,1\}$ bestimmt. Im Fall, dass der stabile Wert des Ausgangssignals '0' ist, ist das Waveform gleichzeitig das Syndrom und es muss nichts getan werden. Im anderen Fall muss die Polarität der Transitionen jedoch umgekehrt werden. Da im Simulator die Signalwerte aller Waveform-Repräsentationen nach Definition für $t \to \infty$ immer einen stabilen Wert von null haben müssen, werden Signale mit einem Finalwert von '1' durch eine *zusätzliche* Transition ∞ terminiert. Deshalb muss hier nur die Transition der Stelle $t = \infty$ *ignoriert* werden, um das Syndrom-Waveform zu generieren, da durch das „Auslassen" des Schaltvorgangs an t der eigentliche Wert des Signals invertiert wird.

Literaturverzeichnis

[AMM83] M. Abramovici, P. R. Menon, D. T. Miller. Critical Path Tracing - An Alternative to Fault Simulation. In *Proc. ACM/IEEE Design Automation Conf. (DAC '83)*, pp. 214–220. 1983. doi:10.1109/DAC.1983.1585651.

[ATH+07] T. Aikyo, H. Takahashi, Y. Higami, J. Ootsu, K. Ono, Y. Takamatsu. Timing-Aware Diagnosis for Small Delay Defects. In *Proc. IEEE Int. Defect and Fault-Tolerance in VLSI Systems Symp. (DFT '07)*, pp. 223–234. 2007. doi:10.1109/DFT.2007.30.

[BA02] M. L. Bushnell, V. D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, 2002. ISBN:0-7923-7991-8.

[BGPV10] B. Bosio, P. Girard, S. Pravossoudovitch, A. Virazel. A Comprehensive Framework for Logic Diagnosis of Arbitrary Defects. *IEEE Transactions on Computers*, 59(3):289–300, 2010. doi:10.1109/TC.2009.177.

[BHHS01] T. Bartenstein, D. Heaberlin, L. Huisman, D. Sliwinski. Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm. In *Proc. Int. Test Conf. (ITC '01)*, pp. 287–296. 2001. doi:10.1109/TEST.2001.966644.

[CIR87] J. L. Carter, V. S. Iyengar, B. K. Rosen. Efficient test coverage determination for delay faults. In *Proc. IEEE Int. Test Conf. (ITC '87)*, pp. 418–427. 1987.

[GLP92] P. Girard, C. Landrault, S. Pravossoudovitch. A novel approach to delay-fault diagnosis. In *Proc. ACM/IEEE Design Automation Conf. (DAC '92)*, pp. 357–360. 1992. doi:10.1109/DAC.1992.227778.

[GLP95] P. Girard, C. Landrault, S. Pravossoudovitch. An advanced diagnostic method for delay faults in combinational faulty circuits. *Journal of Electronic Testing*, 6:277–294, 1995. doi:10.1007/BF00996437.

[Hay86] J. P. Hayes. Digital Simulation with Multiple Logic Values. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 5(2):274–283, 1986. doi:10.1109/TCAD.1986.1270196.

[Hui04] L. M. Huisman. Diagnosing arbitrary defects in logic designs using single location at a time (SLAT). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(1):91–101, 2004. doi:10.1109/TCAD.2003.816206.

[HW07] S. Holst, H.-J. Wunderlich. Adaptive Debug and Diagnosis without Fault Dictionaries. In *Proc. IEEE European Test Symp. (ETS '07)*, pp. 7–12. 2007. doi: 10.1109/ETS.2007.9.

[HW09] S. Holst, H.-J. Wunderlich. A diagnosis algorithm for extreme space compaction. In *Proc. Design, Automation and Test in Europe Conf. & Exhibition (DATE '09)*, pp. 1355–1360. 2009.

[IRW90] V. S. Iyengar, B. K. Rosen, J. A. Waicukauski. On computing the sizes of detected delay faults. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(3):299–312, 1990. doi:10.1109/43.46805.

[KC98] A. Krstić, K.-T. T. Cheng. *Delay Fault Testing for VLSI Circuits*. Kluwer Academic Publishers, 1998. ISBN:0-7923-8295-1.

[KWCL03] A. Krstic, L.-C. Wang, K.-T. Cheng, J.-J. Liou. Diagnosis of delay defects using statistical timing models. In *Proc. VLSI Test Symp. (VTS '03)*, pp. 339–344. 2003. doi:10.1109/VTEST.2003.1197672.

[MA98] A. K. Majhi, V. D. Agrawal. Tutorial: Delay fault models and Coverage. In *Proc. IEEE Int. VLSI Design Conf. (VLSID '98)*, pp. 364–369. 1998. doi:10.1109/ICVD.1998.646634.

[MMSTR08] V. J. Mehta, M. Marek-Sadowska, K.-H. Tsai, J. Rajski. Improving the Resolution of Single-Delay-Fault Diagnosis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(5):932–945, 2008. doi:10.1109/TCAD.2008.917588.

[NVI12] NVIDIA. *NVIDIA CUDA Developer Zone*, 2012. `http://developer.nvidia.com/category/zone/cuda-zone/`.

[PB07] O. Poku, R. D. Blanton. Delay defect diagnosis using segment network faults. In *Proc. IEEE Int. Test Conf. (ITC '07)*, pp. 1–10. 2007. doi:10.1109/TEST.2007.4437602.

[RBG+07] A. Rousset, A. Bosio, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel. DERRIC: A Tool for Unified Logic Diagnosis. In *Proc. IEEE European Test Symp. (ETS '07)*, pp. 13–20. 2007. doi:10.1109/ETS.2007.16.

[Sch11] E. Schneider. CUDA-accelerated Delay Fault Simulation. Student Research Project: Universität Stuttgart, Institut für Technische Informatik (ITI), 2011. URL `http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=STUD-0078&engl=0`.

[Smi85] G. L. Smith. Model for delay faults based upon paths. In *Proc. IEEE Int. Test Conf. (ITC '85)*, pp. 342–349. 1985.

[TBT98] H. Takahashi, K. O. Boateng, Y. Takamatsu. Diagnosis of single gate delay faults in combinational circuits using delay fault simulation. In *Proc. Asian Test Symp. (ATS '98)*, pp. 108–112. 1998. doi:10.1109/ATS.1998.741599.

[WHH02] H.-B. Wang, S.-Y. Huang, J.-R. Huang. Gate-delay fault diagnosis using the inject-and-evaluate paradigm. In *Proc. IEEE Int. Defect and Fault Tolerance in VLSI Systems Symp. (DFT '02)*, pp. 117–125. 2002. doi:10.1109/DFTVS.2002.1173508.

[WL89] J. A. Waicukauski, E. Lindbloom. Failure diagnosis of structured VLSI. *IEEE Design and Test of Computers*, 6(4):49–60, 1989. doi:10.1109/54.32421.

[WLRI87] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, V. S. Iyengar. Transition Fault Simulation. *IEEE Design and Test of Computers*, 4(2):32–38, 1987. doi:10.1109/MDT.1987.295104.

[Wun09] H.-J. Wunderlich, editor. *Models in Hardware Testing*, volume 43 of *Frontiers in Electronic Testing*. Springer Berlin Heidelberg, 2009. doi:10.1007/978-90-481-3282-9.

[WWW06] L.-T. Wang, C.-W. Wu, X. Wen, editors. *VLSI Test Principles and Architectures*. Morgan Kaufmann Publishers, 2006. ISBN:0-12-370597-5.

[YB08] X. Yu, R. D. Blanton. Multiple defect diagnosis using no assumptions on failing pattern characteristics. In *Proc. ACM/IEEE Design Automation Conf. (DAC '08)*, pp. 361–366. 2008. doi:10.1145/1391469.1391567.

[YB10] X. Yu, R. D. Blanton. Diagnosis of Integrated Circuits With Multiple Defects of Arbitrary Characteristics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(6):977–987, 2010. doi:10.1109/TCAD.2010.2048352.

Alle URLs wurden zuletzt am 22. Februar 2012 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Eric Schneider)