



**BMBF-Verbundprojekt  
DIANA  
Schlussbericht**

***Durchgängige Diagnosefähigkeit  
in Halbleiterbauelementen und übergeordneten Systemen  
zur ANALyse von  
permanenten und sporadischen Elektronikausfällen  
im Gesamtsystem Automobil  
(Kurztitel: DIANA)***

**Zuwendungsempfänger:**

AUDI AG (OEM)  
Continental AG (TIER1)  
Infineon Technologies AG (Halbleiterhersteller)  
ZMD AG (Halbleiterhersteller)

**Unterauftragnehmer:**

BTU Cottbus  
FAU Erlangen  
Fraunhofer IIS/EAS Dresden  
Universität Stuttgart  
Universität der Bundeswehr München

## Kurzfassung

Für den potentiellen Käufer eines Automobils spielt heute, neben den zu erwartenden Kosten und vielen anderen Faktoren, die Zuverlässigkeit eine maßgebliche Rolle bei der Kaufentscheidung. Kosten und Zuverlässigkeit werden wiederum ganz wesentlich durch die im Fahrzeug verwendeten elektronischen Systeme bestimmt. Nicht zuletzt durch gestiegene Sicherheits- und Umweltschutzanforderungen hat die Komplexität elektronischer Systeme in den vergangenen Jahren immer weiter zugenommen. Die eindeutige Identifikation fehlerverursachender Komponenten wird jedoch in solchen komplexen, verteilten Systemen immer schwieriger, ist aber gleichzeitig essentiell für hohe Qualität und sichere Systemverfügbarkeit.

Im Rahmen des BMBF-Verbundprojekts DIANA wurden erweiterte Diagnosefähigkeiten in Halbleiterbauelementen, Steuergeräten und dem Gesamtsystem der Automobilelektronik erforscht, um permanent oder sporadisch auftretende Fehler im Automobil schneller und zielgerichteter auffinden zu können. Dabei wurden Methoden und Verfahren entwickelt, die eine effiziente und unter den Rahmenbedingungen unterschiedlicher Anwendungsfälle wirtschaftliche Diagnose solcher Fehler erlauben. Diese Lösungen zur Fehlerdiagnose von hochkomplexen Steuergeräten der Automobilelektronik wurden so gestaltet, dass sie beginnend mit der Fehlermodellierung, über die Schaltungsentwicklung sowie den Produktionstest beim Halbleiterhersteller, und über die Umsetzung der in den Halbleiterbauelementen integrierten Verfahren beim Steuergerätehersteller, schließlich beim Endanwender zur Diagnose und Fehlerbeseitigung eingesetzt werden können. Dank dieses durchgängigen Ansatzes über alle Glieder der Wertschöpfungskette, bereitet DIANA den Weg für eine übergreifende Diagnosemethodik für die Automobilelektronik.

Zu den für diese Diagnosemethodik essentiellen Ergebnissen des Projektes, gehören unter anderem Verfahren, die unter Berücksichtigung der Schaltungsstruktur im System eine exakte Diagnose von Fehlern in digitaler Logik sowie in Flash- und SRAM-Speichern ermöglichen. Auch für die Diagnose von Fehlern in gängigen analogen Komponenten, wie beispielsweise Analog-Digital-Wandler, konnten vollkommen neue Verfahren erarbeitet werden.

An dieser Stelle möchten wir allen beteiligten Parteien aus Wissenschaft und Industrie danken für die erfolgreiche Kooperation im BMBF-Verbundprojekt DIANA.

Besonderem Dank gilt dem Bundesministerium für Bildung und Forschung (BMBF) und den Projektträgern Deutsches Zentrum für Luft - und Raumfahrt (DLR) und VDI/VDE Innovation + Technik für die konstruktive und wegweisende Unterstützung und Begleitung für eine "Durchgängige Diagnosefähigkeit in Halbleiterbauelementen und übergeordneten Systemen zur ANALyse von permanenten und sporadischen Elektronikausfällen".

In enger Kooperation konnten die Verbundpartner wissenschaftliche Ergebnisse erarbeiten, die den Grundstein bilden, auch in Zukunft den begonnenen erfolgreichen Dialog weiterführen zu können.

## Dokument-Historie

<b>Version</b>	<b>Datum</b>	<b>Änderung</b>	<b>Verantwortlich</b>
0,1	24.05.2013	Initiale Version	P. Engelke, IFX
0.2	04.09.2013	Ergebnisse der Arbeitspakete integriert	H. Obermeir, IFX
0.3	05.09.2013	Zwischenversion, Header, Footer, Bilder	H. Obermeir, IFX
0.4	08.10.2013	Review-Kommentare integriert	H. Obermeir, IFX
1.0	10.10.2013	Vorbereitet für Freigabe	H. Obermeir, IFX
1.1	16.10.2013	Neue Komentare von ZMD integriert	H. Obermeir, IFX

## Inhaltsverzeichnis

I. Einleitung .....	6
I.1. Problemstellung .....	6
II. Ziele und Aufgabenstellungen.....	8
II.1. Wissenschaftliche und technische Arbeitsziele .....	10
III. Beschreibung der Arbeitspakete .....	12
III.1. Arbeitspaket 1: Optimierte Fehlermodelle zur effizienten Fehlerlokalisierung in Halbleiterbauelementen .....	12
III.2. Arbeitspaket 2: Bauelemente Selbsttests und Selbstreparatur als Basis für die Diagnose des Gesamtsystems .....	14
III.3. Arbeitspaket 3: Fehleranalyse auf Steuergeräteebene & Demonstrator .....	17
III.4. Arbeitspaket 4: Erfassen und Verarbeiten von Diagnosedaten im Gesamtsystem ...	19
III.5. Partner und Unterauftragnehmer .....	21
IV. Verwertungsplan.....	22
V. Ergebnisse.....	24
V.1. Verbesserung der Diagnosequalität in der Automobilwerkstatt .....	26
V.2. Verbesserung der Lieferqualität an OEM und Endkunden .....	26
V.3. Verringerung der Entwicklungszeit für neue Fehlersuchverfahren .....	26
V.4. Reduzierung der Zeit für die Diagnose und die Fehlerbeseitigung .....	26
V.5. Integration der Diagnoseverfahren in den Produktionstest von Halbleiterbauelementen .....	26
VI. Ausblick .....	28
VI.1. Potentiale für die Automobilindustrie.....	28
VI.2. Zukünftige Handlungsfelder .....	28
VII. Ergebnisse zum Arbeitspaket 1 „Optimierte Fehlermodelle zur effizienten Fehlerlokalisierung“.....	30
VII.1. Ergebnisse zur Aufgabe 1.1 „Fehlerlokalisierung in Analog-, Mixed-Signal- und Digitalkomponenten auf Transistorlevel unter Berücksichtigung von Parameterschwankungen“ .....	30
VII.2. Ergebnisse zur Aufgabe 1.2 „Bewertung von Diagnoseprogrammen bezüglich der Fehlerlokalisierung und des Einflusses von Veränderungen im Herstellungsprozess“ .....	46
VII.3. Ergebnisse zur Aufgabe 1.3 „Fehlerdiagnose für Analog- und Mixed-Signal-Schaltungen“ .....	62
VII.4. Ergebnisse zur Aufgabe 1.4, „Verbesserung der Testgüte für Digitalschaltungen auf Basis erweiterter Fehlermodelle“ .....	78
VIII. Ergebnisse in Arbeitspaket 2: Bauelemente Selbsttests und Selbstreparatur als Basis für die Diagnose des Gesamtsystems.....	94
VIII.1. Ergebnisse zur Aufgabe 2.1: Algorithmen zur analogen Messdatenerfassung und –auswertung mit eingeschränkten Hardware Ressourcen .....	95

VIII.2. Ergebnisse zur Aufgabe 2.2(Teil1): Online Speicher-Selbsttest und Selbstreparatur in der Applikation für RAM-Speicher .....	111
VIII.3. Ergebnisse zur Aufgabe 2.2 (Teil2): Online Speicher-Selbsttest und Selbstreparatur in der Applikation für Flash-Speicher .....	133
VIII.4. Ergebnisse zur Aufgabe 2.3 - Test und Diagnose digitaler Logik in der Applikation unter Verwendung von On-Chip Produktionstestlogik und Standard-Hochgeschwindigkeits-Schnittstellen .....	143
VIII.5. Ergebnisse zur Aufgabe 2.4: Entwicklung einer Methodik, um die DC eines FSBST für einen Microcontroller Core nachzuweisen, sowie deren Anwendung.....	168
VIII.6. Ergebnisse zur Aufgabe 2.5: Selbstreparatur-Funktionen für digitale Standard-Logik und Verbindungsstrukturen.....	204
IX. Ergebnisse zum Arbeitspaket 3: „Fehleranalyse auf Steuergeräteebene & Demonstrator“ .....	217
IX.1. Ergebnisse zur Aufgabe 3.1: „Demonstrator zur Absicherung der Praxistauglichkeit der Ergebnisse des Gesamtprojekts“ .....	217
IX.2. Ergebnisse zur Aufgabe 3.2: „Produkt FMEA auf Steuergeräteebene unter Berücksichtigung erweiterter Fehlermodelle“ .....	227
IX.3. Ergebnisse zur Aufgabe 3.3: „Diagnose von Komponenten und Modulen im CAN-basierten Systemverbund“ .....	239
X. Ergebnisse in Arbeitspaket 4 .....	256
X.1. Aufgabe 4.1 Datenerfassung und Diagnose in Automotive Systemen .....	256
X.2. Aufgabe 4.2 Methoden zur optimalen Diagnoseauflösung auf Systemebene.....	261
X.3. Aufgabe 4.3 Methoden und Algorithmen zur optimalen Diagnoseauflösung für analoge und gemischt digital-analoge Baugruppen .....	274
X.4. Aufgabe 4.4 Software basierter Selbsttest für Test und Diagnose digitaler Logik in Automotive Systemen .....	278
XI. Abkürzungen .....	283
XII. Anhang .....	287
XII.1. Veröffentlichungen und Konferenzbeiträge.....	287
XII.2. Patente und –anmeldungen.....	291
XII.3. Anhang: Einführung in die Architektur des verwendeten Microcontrollers.....	291

## I. Einleitung

Das Automobil ist ein essentieller Bestandteil des modernen Lebens und seine einwandfreie Funktionalität für den typischen Fahrzeugnutzer von hoher Wichtigkeit. Der Stellenwert und der, im Vergleich zu anderen Konsumgütern, hohe Preis für ein Neufahrzeug führen dazu, dass Funktionsstörungen oder Pannen vom Nutzer als intensiver Eingriff in sein Wohlbefinden wahrgenommen werden, selbst wenn sie nur eine Komfortbeeinträchtigung zur Folge haben.

Zudem stellt die Fehlerfindung und -behebung in einem so komplexen System wie dem Automobil heute die Reparaturbetriebe vor eine nicht unerhebliche Herausforderung. Ohne eine Berücksichtigung dieser Belange bereits in der Entwicklungsphase gestaltet sich die Eingrenzung der für einen Ausfall ursächlichen Komponente langwierig und kompliziert.

Für jeden potentiellen Käufer eines Automobils spielt deshalb, neben Design, Farbe und Kosten, die zu erwartende Zuverlässigkeit bei der Kaufentscheidung eine maßgebliche Rolle. Kosten und Zuverlässigkeit werden heute ganz wesentlich durch die im Fahrzeug verwendeten elektronischen Systeme bestimmt. Sie bilden die Grundlage für eine große Spannbreite von Funktionen, angefangen von umweltschonender Motorsteuerung bis zu Fahrerassistenzsystemen, die den Fahrer beim Erkennen und Meistern kritischer Fahrsituationen aktiv unterstützen.

### I.1. Problemstellung

Um die Funktionalität von hochkomplexen elektronischen Systemen im Betrieb sicherzustellen, ergeben sich für die Hersteller vielfältige Herausforderungen. Obwohl bei der Herstellung der elektronischen Systeme umfangreiche Produktionstests die Funktionsfähigkeit der eingesetzten Baugruppen sicherstellen, ist es möglich, dass eine sehr geringe Anzahl nicht erkannter Produktionsfehler bzw. Einflüsse des Systembetriebs, die die Zuverlässigkeit der Baugruppen beeinträchtigen, zu Systemausfällen im Fahrzeug führen können. Ein idealer Produktionstest, der alle denkbar möglichen Produktionsfehler und alle im Betrieb entstehenden Zuverlässigkeitsprobleme ausschließt, wird von allen Herstellern elektronischer Systeme für die Automobilindustrie angestrebt. Die Praxis zeigt jedoch, dass eine geringe Restfehlerwahrscheinlichkeit bleibt, die sich auf Basis des heute verfügbaren Wissens nicht vollständig beseitigen lässt. Ein gangbarer Weg dagegen ist, mit Hilfe von Diagnoseverfahren Rückschlüsse auf mögliche Ausfälle im Automobil zu ziehen. Diese Erkenntnisse können dann direkt in Entwicklung und Produktion elektronischer Systeme zurückfließen, um entsprechende Fehlerursachen zukünftig auszuschließen.

Die steigende Komplexität der verwendeten Fahrzeugelektronik und die damit einhergehende zunehmende Abhängigkeit der Fahrzeugfunktionalität von der Elektronik war Motivation für die Projektpartner, neue Wege für eine effektivere und schnellere Erkennung von Fehlerursachen zu untersuchen. Nur ein schnelles und detailgenaues Verständnis einer Fehlerursache ermöglicht die rechtzeitige und nachhaltige Fehlerbeseitigung.

Daher wurde für die Fahrzeugelektronik eine erweiterte Analyse- und Diagnosefähigkeit angestrebt, welche den Automobilhersteller unterstützt insbesondere sporadisch auftretende, sicherheitskritische Fehler im System aufzufinden und zu diagnostizieren. Eine solche Diagnosefähigkeit ist allerdings nur dann effizient umzusetzen, wenn sich die technische Lösungen über alle Systemebenen erstrecken, angefangen bei den Halbleiterbauelementen über die Steuergeräte bis zum Gesamtsystem Automobil.

Um den Erfolg deutscher Automobilhersteller am Weltmarkt auch weiterhin zu gewährleisten, ist es unerlässlich, dass die in DIANA entwickelten Basistechnologien, die die Qualität und die Sicherheit der Automobile weiter verbessern, verfügbar gemacht werden. Da dies nur möglich sein wird, wenn die daraus entstehenden zusätzlichen Kosten gering bleiben, wurde im Verlauf von DIANA großer Wert auf die Betrachtung der Wirtschaftlichkeit einer durchgängigen Diagnosemethodik geachtet. Sowohl die Hersteller von Steuergeräten, als insbesondere auch die Halbleiterhersteller sind nach Projektabschluss besonders gefordert, ihren Beitrag verstärkt innerhalb der Wertschöpfungskette eines erfolgreichen Automobilbaus in Deutschland zu leisten. Die Bereitstellung von detaillierten Diagnosedaten im Gesamtsystem ist eine radikal neue Anforderung an die Funktionalität der Halbleiterbauelemente, für die in DIANA die notwendigen Basistechniken erarbeitet wurden. Die Ergebnisse aus DIANA zeigen, dass sich aus dieser Anforderung erhebliche Auswirkungen auf die gesamte Entwicklungskette ergeben, beginnend bei der Architektur, über den Schaltungsentwurf, bis zu den Testkonzepten der Bauelemente.

## II. Ziele und Aufgabenstellungen

Vorrangiges Ziel des Projekts DIANA war es, die Diagnosefähigkeiten von elektronischen Steuergeräten im Automobil so wesentlich zu erweitern, dass eine schnellere Fehlererkennung und Fehlerbehebung beim Automobilhersteller bzw. in der Werkstatt ermöglicht wird. Bisher konnten insbesondere die Ursachen sporadisch auftretender Fehler im Nachhinein in der Werkstatt nicht mehr zuverlässig diagnostiziert werden. Es blieb häufig keine andere Möglichkeit, als Systemkomponenten anhand der Fehlerbeschreibung des Kunden und einer geführten Fehlersuche systematisch und sukzessive auszutauschen, was nicht immer zum gewünschten Erfolg führte. Durch das neue Konzept der Diagnosefähigkeit der Halbleiterbauelemente, können relevante Informationen über mögliche Fehlfunktionen während des Betriebs von den Halbleiterbauelementen an übergeordnete Systemkomponenten weitergeben werden, die diese wiederum in ein für eine Werkstattdiagnose geeignetes Format umsetzen und so eine gezielte Fehlerdiagnose ermöglichen. Dazu ist es erforderlich, dass alle Zulieferer der gesamten Wertschöpfungskette von den Herstellern von Halbleiterbauelementen und Steuergeräten bis hin zum KFZ-Gesamtsystem bzw. der zugehörigen Werkstattdiagnosegeräte diese neuen Konzepte akzeptieren und die Einzelkomponenten darauf abgestimmt werden. Eine solche Diagnosekette, die auch detaillierte Informationen aus den Halbleiterbauelementen einbezieht, existierte zum Projektstart nicht. Ziel der Projektpartner in DIANA war es daher, ein übergreifendes Konzept für eine solche Diagnosekette zu erarbeiten und die benötigten Funktionalitäten nach Abschluss des Projekts in den Komponenten und Systemen der beteiligten Projektpartner zur Verfügung zu stellen.

Aus dem übergeordneten Projektziel und den damit verbundenen technischen Lösungen leiten sich konkrete, mittel- und langfristig zur Verfügung stehende Ergebnisse ab, die nach Ende des Projektes DIANA angestrebt werden.

Tabelle II-1 gibt eine erste Übersicht über die in DIANA bearbeiteten Themen und die nach Projektabschluss zu erreichenden langfristigen Verwertungsergebnisse. Eine detaillierte Beschreibung der angestrebten Verwertungsergebnisse findet sich in Abschnitt IV.



Ein wesentlicher Beitrag von DIANA ist es, den Weg für eine Zusammenarbeit entlang der gesamten Wertschöpfungskette bereitet zu haben. Diese Zusammenarbeit ermöglicht es, Diagnoseergebnisse aus dem Gesamtsystem, mittels der erarbeiteten technischen Lösungen, den Halbleiter- und Steuergeräteherstellern zur Verfügung zu stellen. Es wird erwartet, dass die Diagnoseergebnisse aus dem Gesamtsystem, die bei Projektstart in der für die heutigen Systeme notwendigen Vollständigkeit nicht verfügbar waren, zur Verbesserungen der Entwicklungs- und Herstellungsprozesse bei Halbleiter- und Steuergerätehersteller führen werden. Die technische Basis für die neuartige Strategie der Zusammenarbeit konnte von den Projektpartnern durch die im Rahmen des Projekts entwickelten Lösungen gelegt werden. Langfristig werden industrieweite Standards angestrebt, um eine schnellere und effektivere Fehlerbeseitigung über alle Ebenen des Gesamtsystems, basierend auch auf den im Rahmen von DIANA entwickelten Lösungen, zu erreichen. So wurden zum Beispiel im Projekt DIANA Vorschläge erarbeitet, in welchen Formaten die Diagnosedaten gespeichert und bearbeitet werden. Dadurch wird sichergestellt, dass die Diagnoseergebnisse für die gesamte Lieferkette handhabbar werden und die Erkenntnisse auf allen Ebenen in

Tabelle II-1: Übersicht über Verwertung der Ergebnisse von DIANA

In DIANA bearbeitete Themen	Ergebnisse 1 bis 2 Jahre nach Projektabschluss	Langfristige Ergebnisse der Verwertung
Verbesserung der Diagnosequalität in der Automobilwerkstatt	Abgestimmte Definition der Diagnosearchitektur vom Halbleiterbauelement bis zum Diagnosegerät	Anzahl der Werkstattaufenthalte nach Elektronikfehlern um mindestens einen Werkstattaufenthalt verringern
Verbesserung der Lieferqualität an OEM und Endkunden	Einbeziehen neuer Fehlermodelle und Diagnoseergebnisse aus dem System in den Produktionstest	Ausfallrate an frühzeitigen Geräteausfällen mindestens halbieren
Verringerung der Entwicklungszeit für neue Fehler-suchverfahren	Diagnosekonzept auf Basis gemeinsam akzeptierter Schnittstellen	Senkung um ein Viertel im Vergleich zur aktuellen Entwicklungszeit
Reduzierung der Zeit für die Diagnose und die Fehlerbeseitigung	Auswahl geeigneter Diagnosedaten auf Halbleiterebene und Komprimierung der Daten für die Verarbeitung auf Systemebene; maximal doppelte Diagnoselaufzeit in der Werkstatt trotz deutlich erhöhter Datenmenge	Zeit bis zur vollständigen Fehlerbeseitigung im System halbieren
Integration der Diagnoseverfahren in den Produktionstest von Halbleiterbauelementen	Integration der neuen Diagnoseverfahren in den Produktionstest mit max. 5% Kostenerhöhung	Optimierte Diagnoseverfahren als Teil des Produktionstests ohne Erhöhung der Testkosten

zukünftige Entwicklungen einfließen können. Langfristig haben die Datenformate das Potential als Industriestandard etabliert zu werden.

Bei Projektabschluss wurden alle wesentlichen, in DIANA erzielten Ergebnisse, anhand von Demonstratoren visualisiert. Diese sind selektiv aus Halbleiterbauelementen aufgebaut worden, welche die neuen Diagnosefunktionalitäten enthalten. Dabei basieren alle Demonstratoren auf der System Testplattform, die wiederum auf das Continental Getriebesteuergerät VL 381 zurückgeht. Dieses Steuergerät wurde von der Continental AG für das DIANA Projekt weiterentwickelt, es enthält einen Microcontroller der Infineon Technologies AG und wird in stufenlosen Getrieben eingesetzt, die in Automobilen der Audi AG verbaut werden. Das Getriebesteuergerät dient also nicht nur der Veranschaulichung der Projektergebnisse, sondern verbindet unmittelbar drei der an DIANA beteiligten Partner.

Die Ergebnisse des Projektes werden darüber hinaus in eine umfangreiche Aufwand/Nutzen-Analyse einfließen. Diese Analyse dient nach Ablauf des Projekts als Vorbereitung für eine Entscheidung, welche Erfahrungen und Ergebnisse als Anforderungen in die zukünftige Elektronikentwicklung der Audi AG einfließen werden.

Wesentlicher unmittelbarer Nutzen für den Bürger sind zuverlässigere Automobile und weniger und gleichzeitig effizientere Werkstattaufenthalte. Die sichtbare Zuverlässigkeit des Automobils und damit die Zufriedenheit des Nutzers steigen. Als Folge wird die Kundenbindung an deutsche Automobilmarken erhöht.

Nach erfolgreicher Einführung der neuartigen Diagnosefähigkeiten in der Automobilindustrie, bieten sich weitere sicherheitsrelevante Anwendungsfelder für die dann verfügbaren und in einem Massenmarkt erprobten Lösungen an. So kann in einem nächsten Schritt über eine Anwendung z.B. in der Elektronik anderer Verkehrssysteme wie Bahn oder Flugzeug nachgedacht werden. Darüber hinaus liegen die Anwendungen z.B. in der Medizintechnik oder bei hochverfügbarer Kommunikationsinfrastruktur nahe.

## **II.1. Wissenschaftliche und technische Arbeitsziele**

Ziel des Projektes DIANA war es, erweiterte Diagnosefähigkeiten in Halbleiterbauelementen, Steuergeräten und Gesamtsystem der Automobilelektronik zu erforschen, um permanent oder sporadisch auftretende Fehler im Automobil schneller und zielgenauer auffinden zu können. Die neuentwickelten Methoden und Verfahren zur Fehlerdiagnose von hochkomplexen Steuergeräten der Automobilelektronik wurden so aufbereitet, dass sie beginnend mit der Fehlermodellierung, über die Schaltungsentwicklung sowie den Produktionstest beim Halbleiterhersteller, und über die Umsetzung der in den Halbleiterbauelementen integrierten Verfahren beim Steuergerätehersteller, schließlich beim Endanwender zur Diagnose und Fehlerbeseitigung eingesetzt werden können, siehe Abbildung II-1.



Abbildung II-1: Wertschöpfungskette der Automobilelektronik; durchgängige Diagnoseverfahren erhöhen die Zuverlässigkeit beim Endkunden

Mit diesem durchgängigen Ansatz über alle Glieder der Wertschöpfungskette wurde erstmals der Weg beschritten, einen übergreifenden Diagnoseansatz für die Automobilelektronik zu beschreiben. Erklärtes Projektziel war es, für die Diagnose des Gesamtsystems mittelfristig ohne den Einsatz zusätzlicher Laborausstattungen auszukommen.

Die neuen Fehlererkennungsverfahren, basierend auf Embedded-Software oder zusätzlicher Test-Hardware auf den Halbleiterbauelementen und Steuergeräten, wurden auch im Hinblick auf die Integration in den Produktionstest der Halbleiterbauelemente bzw. Steuergeräte entwickelt. Nur durch die ergänzende Berücksichtigung der neuen Fehlererkennungsverfahren kann gewährleistet werden, dass diese im Gesamtsystem keine relevanten Fehler in Halbleiterbauelementen detektieren, die im Produktionstest der Halbleiterbauelemente bzw. Steuergeräte hätten erkannt werden müssen.

Die neuen Verfahren erlauben sowohl Diagnosen aufgrund der klassischen Fehlermodelle, wie z. B. Haftfehler, als auch das Erkennen von parametrischen Abweichungen. Das Erkennen parametrischer Abweichungen im Gesamtsystem ist besonders wichtig, da Fehler aller Voraussicht nach in zunehmendem Maße von parametrischen Degradationen über die Betriebszeit des Gesamtsystems hervorgerufen werden.

Langfristig wird erwartet, dass statistisch relevante Fehlerdiagnoseergebnisse aus dem Gesamtsystem in den Entwicklungsprozess beim Halbleiter- und Steuergerätehersteller zurückfließen werden. Sofern die Diagnose im Gesamtsystem systematische Schwächen im Entwicklungsprozess der Vorprodukte aufdeckt, können diese in einem verbesserten Entwicklungsprozess ausgeräumt werden. Die Qualität des Gesamtsystems nimmt in einer solchen Regelschleife kontinuierlich zu.

### III. Beschreibung der Arbeitspakete

Das Projekt DIANA beinhaltet insgesamt vier Arbeitspakete, die jeweils von einem der Projektpartner geleitet wurden. Eine Übersicht der Arbeitspakete zeigt Abbildung III-1. Nachfolgend werden die Ziele der vier Arbeitspakete zusammengefasst. Zudem wird jeweils die Kurzbeschreibung der Aufgabenstellung aus der Vorhabensbeschreibung wiedergegeben.

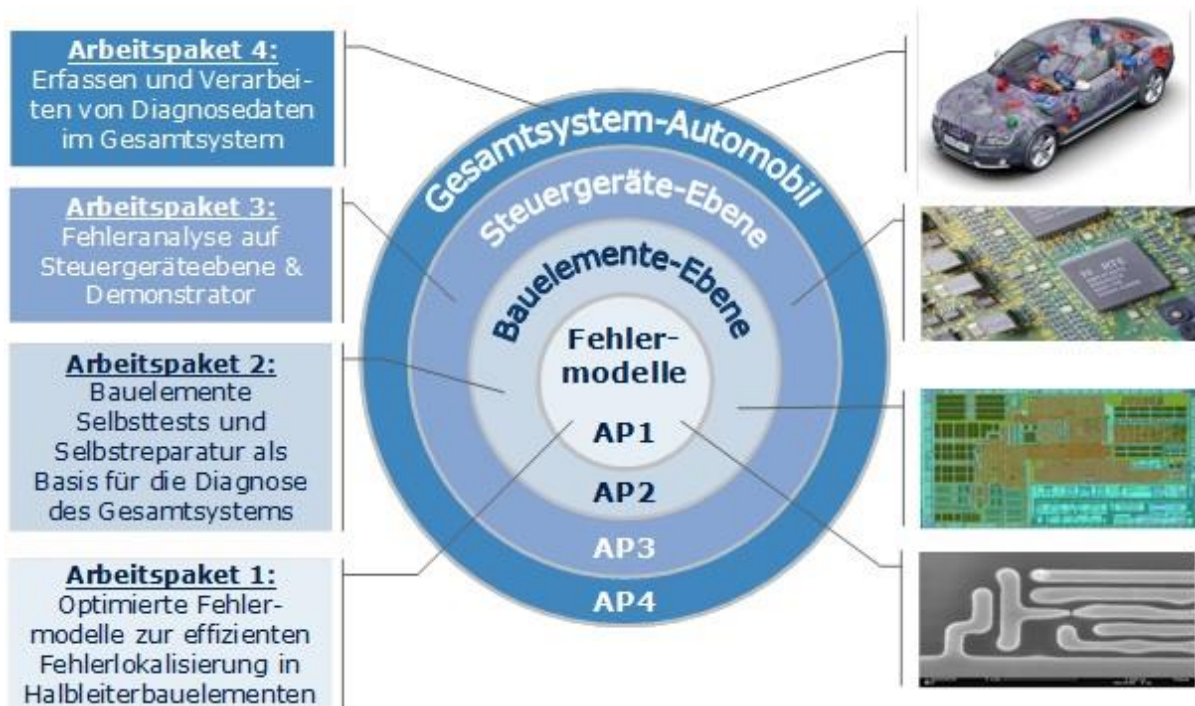


Abbildung III-1: Übersicht Arbeitspakete im Förderprojekt DIANA

#### III.1. Arbeitspaket 1: Optimierte Fehlermodelle zur effizienten Fehlerlokalisierung in Halbleiterbauelementen

Im ersten Arbeitspaket, „AP1: Optimierte Fehlermodelle zur effizienten Fehlerlokalisierung in Halbleiterbauelementen“, wurden Grundlagenarbeiten zur effizienten Vorbereitung der Fehlerdiagnose in Halbleiterbauelementen durchgeführt. Dazu wurden Fehlermodelle auf Transistorebene auf ihre Eignung zur Fehlerlokalisierung untersucht. Es wurden Verfahren zur Verfügung gestellt, mit denen bewertet werden kann, wie gut die in Arbeitspaket 2 erarbeiteten Verfahren angenommene Fehler erkennen bzw. lokalisieren können.

Das Arbeitspaket 1 wurde von den beiden Halbleiterherstellern im Projektkonsortium, Infineon und ZMDI, gemeinsam bearbeitet. Die Leitung oblag der ZMDI. Unterauftragnehmer für alle in Arbeitspaket 1 bearbeiteten Aufgaben war das Fraunhofer IIS/EAS Dresden. Eine Übersicht der Aufgaben in Arbeitspaket 1 findet sich in Abbildung III-2.


<b>Arbeitspaket 1:</b> Optimierte Fehlermodelle zur effizienten Fehlerlokalisierung in Halbleiterbauelementen		
	Partner	Unterauftragnehmer
<b>Aufgabe 1.1:</b> Fehlerlokalisierung in Analog-, Mixed-Signal- und Digitalkomponenten auf Transistorlevel unter Berücksichtigung von Parameterschwankungen	Infineon	Fraunhofer IIS/EAS Dresden
<b>Aufgabe 1.2:</b> Bewertung von Diagnoseprogrammen bezüglich der Fehlerlokalisierung und des Einflusses von Veränderungen im Herstellungsprozess	Infineon	Fraunhofer IIS/EAS Dresden
<b>Aufgabe 1.3:</b> Fehlerdiagnose für Analog- und Mixed-Signal Schaltungen	ZMD	Fraunhofer IIS/EAS Dresden
<b>Aufgabe 1.4:</b> Verbesserung der Testgüte von Digital-schaltungen auf Basis erweiterter Fehlermodelle	ZMD	Fraunhofer IIS/EAS Dresden

Abbildung III-2: Aufgaben in Arbeitspaket 1, Projektpartner und Unterauftragnehmer. Nachfolgend wird die Kurzfassung der Aufgabenstellung in AP1 aus der Vorhabensbeschreibung wiedergegeben.

**Aufgabe 1.1:** „Fehlerlokalisierung in Analog-, Mixed-Signal- und Digitalkomponenten auf Transistorlevel unter Berücksichtigung von Parameterschwankungen“

Anhand von Schaltungsdaten (Schaltplan, Layout und Modelle) und fehlerbedingten Ausfallsignaturen wird eine Diagnosestrategie entwickelt. Diese Diagnosestrategie berücksichtigt Parameter- und Layouteinflüsse. Sie listet die Schwachstellen der entworfenen Schaltung auf und zeigt die Auswirkungen der erkannten Schwächen auf die Ausbeute und Zuverlässigkeit des Produkts auf. Die Testgrenzen (Konfidenzbereiche), unter denen das Produkt während des Tests als funktionsfähig („pass“) klassifiziert wird, sind ebenfalls ein Ergebnis dieser Untersuchung. Die erstellten Verzeichnisse von Fehlerbildern bilden die Grundlage für die Fehlermodellierung, so dass eine Lokalisierung der Fehlerursache auch von übergeordneten Komponenten aus möglich wird.

**Aufgabe 1.2:** „Bewertung von Diagnoseprogrammen bezüglich der Fehlerlokalisierung und des Einflusses von Veränderungen im Herstellungsprozess“

Es wird an einer Beispielkomponente aus dem Automotive-Bereich eine verallgemeinerungsfähige Vorgehensweise erarbeitet, um aus den zur Verfügung stehenden Informationen zu applikationsspezifischen und herstellungsbedingten Parameterschwankungen die Auflösung von Diagnoseprogrammen bezüglich Fehlerort und Fehlerursache, insbesondere für sporadisch auftretende Fehler, zu bewerten.

### Aufgabe 1.3: „Fehlerdiagnose für Analog- und Mixed-Signal-Schaltungen“

Die Untersuchungen zur Fehlerdiagnose werden sich zunächst auf eingebettete Tri-Port RAMs als Analog- bzw. Mixed-Signal-Komponenten konzentrieren. RAMs haben an ihrem Rand bezüglich der Adresse sowie der ein- und auszulesenden Daten digitales Verhalten. Jedoch sind im Inneren die funktionsbestimmenden Komponenten, wie Pre-Charging, Schreib- und Leseverstärker, sowie die eigentlichen Speicherzellen, analog. Ziel ist es, für die Diagnose, auf Grund der großen Komplexität von RAMs, zu einer diagnosegerechten Schaltungsbeschreibung zu kommen. Wegen der strukturellen Regularität und der verwendeten Testverfahren (March-Tests) wird erwartet, dass eine diagnosegerechte Schaltungsbeschreibung auf die zu diagnostizierende Zelle, die sie umgebenden Zellen, die zugehörigen Schaltungsteile des Pre-Charging und der Schreib- und Leseverstärker beschränkt werden kann. Es müssen die elektrischen Lasten und Eigenschaften der Signalübertragung der „ausgesparten“ Schaltungsteile in geeigneter Weise berücksichtigt werden. Eine analoge Fehlermodellierung der RAM-spezifischen Defekte wird ausgeführt. Die Fehlermodellierung wird in Hinsicht auf die Fehlerinjektion Einfluss auf die diagnosegerechte Schaltungsbeschreibung haben.

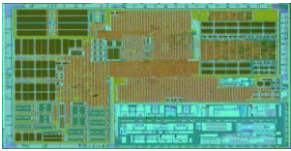
### Aufgabe 1.4: „Verbesserung der Testgüte für Digitalschaltungen auf Basis erweiterter Fehlermodelle“

Um die Wirkungen der in digitalen Schaltungen auftretenden Defekte zu analysieren, ist ihre Modellierung auf der Logik-Bit-Ebene völlig unzureichend. Deshalb wird sie in dieser Aufgabe auf der elektrischen Ebene erfolgen. Es wird davon ausgegangen, dass sich das Verhalten eines Defektes durch die elektrischen Größen Strom, Spannung, Magnetfluss und Ladung, also durch eine Zusammenschaltung elektrischer Netzwerkelemente, modellieren lässt. Aus diesem Grunde werden für die digitalen Schaltungen ebenfalls Netzwerkbeschreibungen auf der elektrischen Ebene verwendet. Das elektrische Fehlermodell wird an den mit dem Defektort korrespondierenden Knoten in die Schaltung injiziert. Eine bessere Hardwarenähe des elektrischen Schaltungsmodells wird durch die Einbeziehung von parasitären Schaltelementen erreicht. Wegen des höheren Simulationsaufwandes auf der elektrischen Ebene wird die digitale Schaltung in Teilschaltungen zerlegt. Dabei ist die Lage der Fehlerorte zu berücksichtigen. Jede dieser Teilschaltungen wird auf der elektrischen Ebene fehlersimuliert. Durch eine geeignete Modellierung der Schaltungsumgebung der jeweiligen Teilschaltungen werden die elektrischen Wirkungen des Fehlerverhaltens zu den digitalen primären Test-Out-Pins weitergeleitet.

## **III.2. Arbeitspaket 2: Bauelemente Selbsttests und Selbstreparatur als Basis für die Diagnose des Gesamtsystems**

Im zweiten Arbeitspaket, „AP2: Bauelemente Selbsttest und Selbstreparatur als Basis für die Diagnose des Gesamtsystems“, wurden teilweise schon bekannte und darüber hinaus auch im Rahmen von DIANA neu entwickelte Hard- und Softwarebasierte Selbsttest- und Selbstreparaturverfahren auf ihre Eignung zum Einsatz auf Systemebene untersucht. Für geeignete Verfahren wurden Lösungen zur Integration der Verfahren auf Systemebene entwickelt, zum Beispiel für die Verwendung von auf

**Arbeitspaket 2:**  
Bauelemente Selbsttests und Selbstreparatur als Basis für die Diagnose des Gesamtsystems



	Partner	Unterauftrag - nehmer
<p><b>Aufgabe 2.1:</b> Algorithmen zur analogen Messdaten - erfassung und -auswertung auf Systemen mit eingeschränkten Hardware Ressourcen; Integration in den Produktionstest</p>	<b>Infineon</b>	
<p><b>Aufgabe 2.2:</b> Online Speicher -Selbsttest und Selbst - reparatur in der Applikation</p>	<b>Infineon</b>	
<p><b>Aufgabe 2.3:</b> Test und Diagnose digitaler Logik in der Applikation unter Verwendung von On -Chip Produktionstestlogik und standard Hochgeschwindigkeits -Schnittstellen</p>	<b>Infineon</b>	<b>BTU Cottbus</b>
<p><b>Aufgabe 2.4:</b> HW und SW basierter Build -in-Self-Test für Prozesskerne</p>	<b>Infineon</b>	
<p><b>Aufgabe 2.5:</b> Selbstreparatur -Funktionen für Logik und Verbindungsstrukturen</p>	<b>Infineon</b>	<b>BTU Cottbus</b>

Abbildung III-3: Aufgaben in Arbeitspaket 2, Projektpartner und Unterauftragnehmer.

Systemebene vorhandenen Datenschnittstellen zum Austausch von Test- und Diagnosedaten. Andere Verfahren wurden so erweitert, dass ihr Einsatz auf Systemebene möglich ist, als Beispiel sei die Erweiterung von Testdatenkompressionsverfahren für den Einsatz als Selbsttest im Gesamtsystem genannt.

Das Arbeitspaket 2 wurde von der Infineon Technologies AG geleitet. Unterauftragnehmer für die Aufgaben 2.3 und 2.5 in AP2 war die Brandenburgische Technische Universität, Cottbus. Eine Übersicht der Aufgaben in Arbeitspaket 2 findet sich in Abbildung III-3.

Nachfolgend wird die Kurzfassung der Aufgabenstellung in AP2 aus der Vorhabensbeschreibung wiedergegeben.

Aufgabe 2.1: „Algorithmen zur analogen Messdatenerfassung und –auswertung auf Systemen mit eingeschränkten Hardware Ressourcen; Integration in den Produktionstest“

Bereitstellung geeigneter Algorithmen zur Messdatenauswertung auf Systemen mit eingeschränkten Hardwareressourcen. Ankoppelung der Algorithmen über geeignete funktionale Schnittstellen an das Gesamtsystem. Integration der entwickelten Selbst-

testverfahren in den Standardproduktionstest zur Korrelation von klassischen Produktionstestergebnissen und Selbsttestergebnissen.

Aufgabe 2.2: „Online Speicher-Selbsttest und Selbstreparatur in der Applikation“

Weiterentwicklung und Verbesserung von effizienten Selbsttest und Selbstreparaturmethoden für flüchtige Speicher (SRAM). Ziel ist die Erkennung und ggf. die Reparatur von Fehlern, die erst im Betrieb z.B. durch Alterung oder Stress entstehen. Die zu entwickelnden Methoden sollen daher nicht nur im Produktionstest sondern auch in der Betriebsphase einsetzbar sein, ohne dass ein signifikanter Zusatzaufwand von Seiten des Gesamtsystems erforderlich ist. Dazu ist es notwendig die bekannten Verfahren so zu erweitern, dass sie vom Gesamtsystem aufgerufen und gesteuert werden können.

Aufgabe 2.3: „Test und Diagnose digitaler Logik in der Applikation unter Verwendung von On-Chip Produktionstestlogik und Standard Hochgeschwindigkeitsschnittstellen“

Die auf komplexen Halbleiterbauelementen vorhandene Logik für den Produktionstest digitaler Standardkomponenten wird so erweitert, dass sie für den Selbsttest und die Selbstdiagnose im Endgerät genutzt werden kann. Zur Verbesserung der Testqualität wird die Möglichkeit geschaffen, Testdaten über in der Applikation vorhandene Hochgeschwindigkeits-Schnittstellen mit dem Gesamtsystem auszutauschen.

Aufgabe 2.4: „HW und SW basierter Built-In Self Test für Prozessorkerne“

Zur Verbesserung der Beobachtbarkeit von Fehlern im Prozessorkern von Microcontrollern für sicherheitskritische Anwendungen aus den Bereichen Automobil- und Automatisierungselektronik werden neue Selbsttest-Lösungen auf Basis funktioneller Software erforscht. Darüber hinaus werden Hardware basierte Selbsttests erforscht, die die Software basierten Selbsttest komplementieren. Zum Nachweis der Wirksamkeit dieser Selbsttest werden neue Verfahren entwickelt und validiert.

Aufgabe 2.5: „Selbstreparatur-Funktionen für digitale Standardlogik und Verbindungsstrukturen“

Auf der Basis wesentlicher Vorarbeiten des Unterauftragnehmers BTU Cottbus wird untersucht, ob und wie Verfahren der eingebauten Selbstreparatur für Standard-Logik und Verbindungsstrukturen in Verfahren des IC- und Systemtests integriert werden können. Die Basis-Verfahren für die (Selbst-) Reparatur von Logik werden vom Unterauftragnehmer im Rahmen eines getrennten Förderprojekts entwickelt. Dabei erfolgt die Implementierung von Logik auf der Basis Semi-regulärer Basisblöcke, die jeweils administrierte Redundanz enthalten. Diagnostische Test- und Selbstreparatur-Funktionen werden weitgehend integriert. In DIANA wird eine Verknüpfung der Verfahren mit übergeordneten Testfunktionen eines Systems untersucht, um zu zeigen, in wie weit diese Verfahren dazu beitragen können, den Gesamt-Fehlerzustand des Systems zu überwachen. Die in dieser Aufgabe behandelten Fragestellungen weisen über die in den anderen Aufgaben in Arbeitspaket 2 bearbei-



teten Inhalte hinaus. Eine möglichst exakte Selbstdiagnose ist die Voraussetzung für jede Art von Selbstreparatur. Die Selbstreparatur von Standardlogik wird ansonsten in DIANA nicht weiter betrachtet. In dieser Aufgabe wird jedoch geklärt, in wie weit die Ergebnisse aus den anderen Aufgaben in Arbeitspaket 2 schon heute als Grundlage für ein zukunftsweisendes Thema wie die Selbstreparatur von Standardlogik verwendet werden können.

### III.3. Arbeitspaket 3: Fehleranalyse auf Steuergeräteebene & Demonstrator

Im dritten Arbeitspaket, „AP3: Fehleranalyse auf Steuergeräteebene & Demonstrator“, wurden die Anforderungen aus Steuergerätesicht an die Diagnoseinfrastruktur auf der Halbleiterebene definiert. Die in AP2 entstandenen Verfahren wurden auf ihre Eignung für die Diagnose auf Steuergeräteebene bewertet und angepasst. In AP3 entstand im Wesentlichen ein Demonstrator auf Steuergeräteebene. Ziel war es zu zeigen, dass die Test- und Diagnoseverfahren einen Fehler innerhalb eines Halbleiters oder in der angeschlossenen Peripherie, wie zum Beispiel Sensoren, erkennen und die Fehlerinformation effizient an das Gesamtsystem weitermelden können.

AP3 wurde von der Continental AG geleitet. Die Aufgaben in AP3 haben die Partner Continental AG und Infineon Technologies AG bearbeitet. In Aufgabe 3.3 wurde die Infineon Technologies AG durch den Unterauftragnehmer von der Universität der Bundeswehr in München unterstützt. Abbildung III-4 stellt die Aufgaben in AP3 graphisch dar

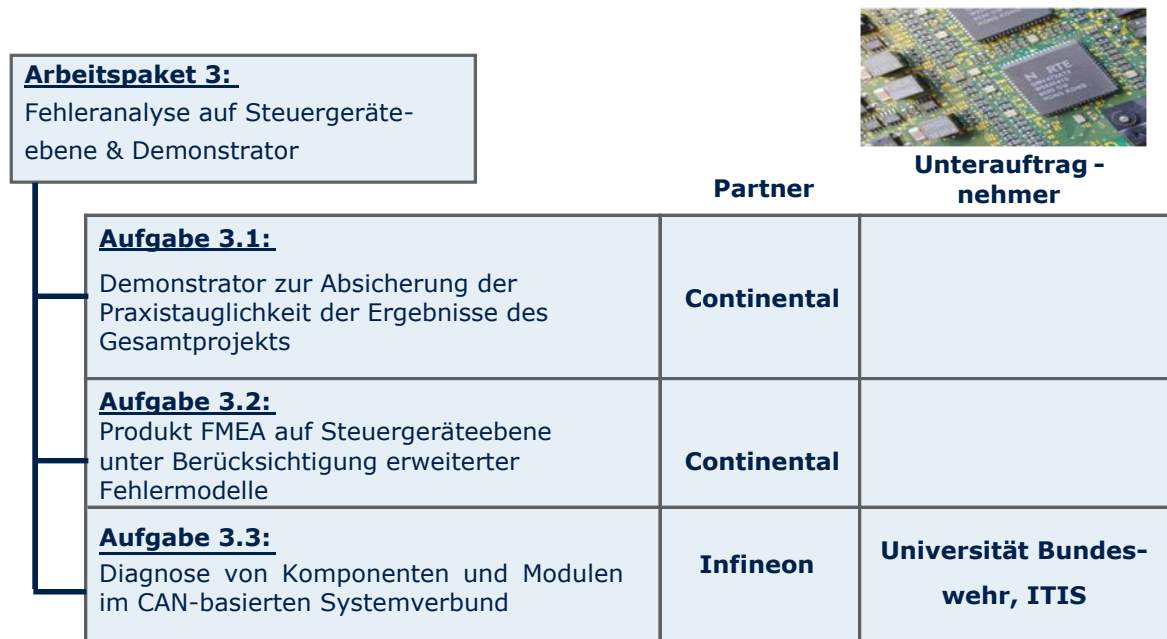


Abbildung III-4: Aufgaben in Arbeitspaket 3, Projektpartner und Unterauftragnehmer

Nachfolgend wird die Kurzfassung der Aufgabenstellung in AP3 aus der Vorhabensbeschreibung wiedergegeben.

### Aufgabe 3.1: „Demonstrator zur Absicherung der Praxistauglichkeit der Ergebnisse des Gesamtprojekts“

Aufbau eines fahrzeugtauglichen Demonstrators auf Steuergeräteebene zur Verifikation der Projektergebnisse des Gesamtprojekts. Dazu wird aus den bei der Continental Automotive Group verfügbaren Steuergeräten eine geeignete Architektur ausgewählt und so erweitert, dass Verfahren, die auf der Bauelemente- (AP2) und Systemebene (AP4) erforscht, spezifiziert und prototypisch implementiert werden, im Zusammenspiel auf ihre Praxistauglichkeit geprüft werden können. In Form von Analysen und Messungen am Demonstrator ist sicherzustellen, dass der Mehraufwand für die Diagnose, in Form von Speicherbedarf, Diagnoselaufzeit und allgemeiner Reaktionszeit, den Anforderungen des Gesamtsystems gerecht wird. Eine prototypische Integration des Demonstrators in das Elektronik-Gesamtsystem direkt beim Automobilhersteller wird angestrebt.

### Aufgabe 3.2: „Produkt FMEA<sup>1</sup> auf Steuergeräteebene unter Berücksichtigung erweiterter Fehlermodelle“


In DIANA werden erweiterte Fehlermodelle für komplexe Logikbausteine entwickelt, insbesondere um die Diagnoseauflösung zu verbessern. Existierende, wie neu zu entwickelnde Fehlermodelle werden im Rahmen von Aufgabe 3.2 auf ihre Eignung für die Verwendung im Rahmen einer Produkt-FMEA auf Steuergeräteebene geprüft. Es wird anhand des Demonstrators auf Steuergeräteebene gezeigt, in wie weit die in einer FMEA gewonnen Erkenntnisse dem realen Verhalten in der Fehlerwirkungskette entsprechen. Ergebnisse aus theoretischen Vorüberlegungen und Erkenntnisse aus den praktischen Erfahrungen am Demonstrator können zur weiteren Verbesserung der Fehlermodellierung und zur Optimierung der Diagnoseverfahren beitragen.

### Aufgabe 3.3: „Diagnose von Komponenten und Modulen im CAN-basierten Systemverbund“

Steigende Integrationsdichte und Kapselung von Modulen und Modulkomponenten sind als zunehmender Trend im Bereich der physikalischen Realisierung moderner Steuergeräte in der Automobil-Domäne zu beobachten. Dies gestaltet funktionale Tests und die Analyse der physikalischen Allokation auftretender Störungen zunehmend als Herausforderung. In dieser Aufgabe wird eine Diagnosemethode entwickelt, die es dem Anwender ermöglicht den spezifikationsgemäßen Zustand von Steuergeräten und deren Komponenten ohne physikalische Eingriffe zu evaluieren. Zur Veranschaulichung wird ein Demonstrator spezifiziert und prototypisch implementiert, welcher Kernkomponenten in verteilten Steuergeräten über einen vorhandenen CAN-Bus (Controller Area Network) anspricht, diese diagnostiziert und auftretende Fehler lokalisiert.

---

<sup>1</sup> FMEA: Failure Mode and Effect Analysis – Auswirkungsanalyse



<b>Arbeitspaket 4:</b> Erfassen und Verarbeiten von Diagnosedaten im Gesamtsystem	Partner	Unterauftrag- nehmer
<b>Aufgabe 4.1:</b> Datenerfassung und Diagnose in Automotive Systemen	Audi	FAU-Erlangen, LHS Uni Stuttgart, ITI
<b>Aufgabe 4.2:</b> Methoden zur optimalen Diagnoseauflösung auf Systemebene	Audi	FAU-Erlangen, LHS
<b>Aufgabe 4.3:</b> Methoden und Algorithmen zur optimalen Diagnoseauflösung für analoge und gemischt digital-analoge Baugruppen	Continental	FAU-Erlangen, LZS
<b>Aufgabe 4.4:</b> Software basierter Selbsttest für Test und Diagnose digitaler Logik in Automotive Systemen	Infineon	Universität Stuttgart, ITI

Abbildung III-5: Aufgaben in Arbeitspaket 4, Projektpartner und Unterauftragnehmer.

### III.4. Arbeitspaket 4: Erfassen und Verarbeiten von Diagnosedaten im Gesamtsystem

Das vierte Arbeitspaket, „AP4: Erfassen und Verarbeiten von Diagnosedaten im Gesamtsystem“, bearbeitete die Problemstellung aus Sicht des Gesamtsystems. Unter anderem wurden dabei folgende Fragestellungen bearbeitet: Welche Anforderung an Datenmenge, Diagnoselaufzeiten und Diagnoseauflösung gibt es aus Sicht des Gesamtsystems im Betrieb, damit ein reibungsloser Betrieb gewährleistet werden kann. Welche Randbedingungen herrschen bzgl. einer Diagnose in der Werkstatt. Wie wirken sich die diversen, für jeden Einsatzfall spezifischen Randbedingungen auf die Möglichkeiten der Diagnoseauflösung aus.

AP4 wurde von der Audi AG geleitet. Die Aufgaben in AP4 haben sich die Partner Audi AG, Continental AG und Infineon Technologies AG geteilt. Als Unterauftragnehmer waren die Universität Stuttgart und die Lehrstühle für Hardware-Software-Co-Design sowie für Zuverlässige Schaltungen und Systeme an der FAU Erlangen beteiligt. Eine graphische Übersicht der Aufgaben in Arbeitspaket 4, sowie die Zuordnung der Partner und Unterauftragnehmer findet sich in Abbildung III-5.

Nachfolgend wird die Kurzfassung der Aufgabenstellung in AP4 aus der Vorhabensbeschreibung wiedergegeben.

#### Aufgabe 4.1: „Datenerfassung und Diagnose in Automotive Systemen“

In einem ersten Arbeitsschritt ist zu untersuchen, inwieweit bestehende Test-Infrastruktur der Schaltung beispielsweise für den Selbsttest die Diagnose auf Systemebene unterstützen kann oder entsprechend zu erweitern ist. Dabei bieten sich standardisierte Strukturen an, wie JTAG-Ketten (IEEE 1149.1) und Ausstattungen

nach IEEE 1500 (Core Test Standard) wie schaltungsspezifische Selbstaussstattungen wie MISR<sup>2</sup>, Kompressions- und Dekompressionslogik oder Prüfketten für den eingebetteten oder externen Test. Auf der Basis einer geeigneten Auswahl und Ergänzung dieser Strukturen sind Methoden zu entwickeln, diese für eine autonome Diagnose zu verwenden. Bei einer Datenerfassung während des Betriebs darf die Funktion eines Chips, eines Steuergerätes und des Gesamtsystems nicht beeinträchtigt werden. Um das Datenvolumen gering zu halten, werden die zu erfassenden Daten geeignet ausgewählt und komprimiert. In diesem Zusammenhang wird untersucht, inwieweit höhere Schichten bei der Auswahl unterstützen können. Es sollen Verfahren, die auf Systemebene Daten entweder periodisch oder ereignisgetrieben erfassen, entwickelt und untersucht werden. Hierbei steht die Herausforderung, dass die Erfassung ohne Beeinträchtigung der Funktionalität des Systems erfolgt, im Vordergrund. Die weitgehende Wiederverwendung der offline Infrastruktur für die online Diagnose führt zu Vorteilen bezüglich einheitlichen Zugriffs, einheitlicher Steuerung und der Kosten.

#### Aufgabe 4.2: „Methoden zur optimalen Diagnoseauflösung auf Systemebene“

Die umfangreichen Datenmengen, die sowohl in der Offline-Diagnose zur Erkennung permanenter Fehler als auch Online für transiente und intermittierende Fehler anfallen, erfordern den Einsatz von Datenkompressionstechniken, wie sie auch im Produktionstest verwendet werden. Entsprechende Signaturen sind so auszuwählen, dass die diagnostische Auflösung in den weiteren Schritten möglichst wenig beeinträchtigt wird. Die Weitergabe der komprimierten Daten auf Chipebene basiert auf den in Aufgabe 4.1 zu entwickelnden Schnittstellen und Formaten. Dabei sind die Kompression im Gerät und externe Diagnosealgorithmen aufeinander abzustimmen. Es wird ein Kompaktierungsverfahren gewählt, das neben einem vertretbaren Hardwareaufwand vor allem eine weitgehende Erhaltung der diagnostischen Auflösung ermöglicht. Um dies zu zeigen, wird ein Diagnosealgorithmus angepasst, so dass dieser mit komprimierten Daten arbeitet und die erreichbare diagnostische Auflösung experimentell ermittelt.

#### Aufgabe 4.3: „Methoden und Algorithmen zur optimalen Diagnoseauflösung für analoge und gemischt digital-analoge Baugruppen“

Die Anforderungen an eine hohe Fehlererkennung und damit eine zweckmäßige und wirtschaftliche Diagnose von Fehlverhalten von elektronischen Bauteilen kumulieren in der Fähigkeit, die Diagnose und Analyse mit genau der noch geforderten Präzision durchführen zu können, welche dann zu dem optimalen und damit minimalen Ressourcenverbrauch führt.

Der Prüfling bestehend aus Sensor, Übertragungskanal, Analog-Digital-Wandler für die Digitalisierung und Kompression der analogen Mess- bzw. Testwerte muss so diagnostiziert werden können, dass mögliche Fehlerursachen später wieder aus diesen komprimierten Daten extrahiert werden können. Der Prüfling ist durch ein

---

<sup>2</sup> MISR: Multiple Input Signature Register – Signaturregister

geeignetes Messverfahren aus seiner Umgebung herauszuschneiden. Dazu sind geeignete Methoden und Algorithmen zu erforschen, welche aus der Signatur der digitalisierten Testantworten auf das gesuchte Fehlverhalten des Prüflings schließen können. Prüfling, Übertragungskanal, Digitizer und Kompressor (Bildung der digitalen Signatur) müssen somit einstellbar und reproduzierbar mit einer definierten Prüfschärfe aufgelöst werden können. Die messtechnischen Zugänge (Signalpfade) zu den internen Beobachtungs- und Steuerungsknoten sind damit hinsichtlich ihrer elektrischen Eigenschaften extrahiert und können berücksichtigt werden, so dass auch mit vergleichsweise einfachem Prüfinstrumentarium und Messprozeduren eine hohe Diagnoseschärfe, und damit genaue und möglichst eindeutige Fehlerlokalisierung erreicht werden kann.

#### Aufgabe 4.4: „Software basierter Selbsttest für Test und Diagnose digitaler Logik in Automotive Systemen“

Diagnose der Hardware eingebetteter Systeme hat strukturorientiert zu erfolgen, um aus fehlerhaften Verhalten auf defekte physikalische Einheiten schließen zu können. Demgegenüber steht auf der Systemebene lediglich das Verhalten als Hardware/Software-Schnittstelle zur Verfügung. In dieser Aufgabe werden Hardwareausstattungen spezifiziert und Methoden zur Generierung von Software-Templates entwickelt, die es erlauben, durch einen neuen Software-basierten Selbsttest Fehler der digitalen Hardware-Struktur mit einer hinreichend großen Auflösung und mit einer für die Systemebene handhabbaren Datenmenge zu diagnostizieren.

### III.5. Partner und Unterauftragnehmer

Tabelle III-1 zeigt eine Übersicht der vier Arbeitspakete und eine Zuordnung der Arbeitspaketleitung sowie der Mitarbeit der Projektpartner.

Tabelle III-1: Mitwirkung der Partner und Unterauftragnehmer in DIANA

Projektpartner davon Unterauftragnehmer	AP1	AP2	AP3	AP4
Audi				✓
LHS, FAU Erlangen				✓
ITI, Universität Stuttgart				✓
Continental			✓	✓
LZS, FAU Erlangen				✓
Infineon	✓	✓	✓	✓
LTI, BTU Cottbus		✓		
IIS/EAS Dresden	✓			
ITIS, Universität der Bundeswehr			✓	
ITI, Universität Stuttgart				✓
ZMDI	✓			
IIS/EAS Dresden	✓			

## IV. Verwertungsplan

Nach Abschluss des Projekts werden die erzielten Ergebnisse der Verwertung zugeführt. Konkret werden von den Partnern des Projekts DIANA folgende Verwertungsergebnisse angestrebt, siehe auch Tabelle II-1.

### IV.1.1. *Verbesserung der Diagnosequalität in der Automobilwerkstatt*

Die Projektpartner werden, basierend auf einem durchgängigen Diagnoseansatz, die Fähigkeiten zur Diagnose von Elektronikfehlern in der Automobilwerkstatt deutlich verbessern. Grundlage ist die Abstimmung einer durchgängigen Diagnosearchitektur vom Halbleiterbauelement zum Diagnosegerät. Unter Verwendung der Diagnosearchitektur werden sich die häufig iterativen Werkstattaufenthalte, die zur Fehlerbeseitigung nach Elektronikausfällen benötigt werden, langfristig um mindestens einen Werkstattaufenthalt verringert.

### IV.1.2. *Verbesserung der Lieferqualität an OEM und Endkunden*

Da die erforschten Diagnoseverfahren in den jeweiligen Produktionstest der Vorprodukte integriert werden, wird sich auch die Lieferqualität der Halbleiterbauelemente und Steuergeräte verbessern. Diese verbesserte Lieferqualität wird sich im Wesentlichen in einer Reduktion der frühzeitigen Geräteausfälle niederschlagen. Basierend auf den neuen Diagnoseverfahren wird mindestens eine Halbierung der frühzeitigen Geräteausfälle angestrebt.

### IV.1.3. *Verringerung der Entwicklungszeit für neue Fehlersuchverfahren*

Durch die im Projekt definierte Diagnoseinfrastruktur lassen sich Fehler im System bzw. dessen Komponenten schneller und exakter identifizieren. In der Praxis kann man durch Verwendung abgestimmter Soft- und Hardware-Schnittstellen damit schneller zwischen bekannten und nicht bekannten Fehlerursachen unterscheiden. Die Zeit für die Analyse von Fehlern noch unbekannter Herkunft wird sich durch die zusätzlich verfügbare Diagnoseauflösung verringern. Die Projektpartner streben hierfür eine Reduktion der Analysezeit um ein Viertel an, im Vergleich zur aktuellen Zeit für die Analyse noch nicht bekannter Fehler.

### IV.1.4. *Reduzierung der Zeit für die Diagnose und die Fehlerbeseitigung*

Bisher werden komplexe Analyse- und Diagnoseverfahren ausschließlich für eine geringe Anzahl von Schaltkreisen im Labor angewendet. Für den Einsatz beim Automobilhersteller bzw. in der Werkstatt sind die verfügbaren Verfahren hinsichtlich der Laufzeit, der erforderlichen Testausstattung und des qualifizierten Bedienpersonals viel zu aufwändig. Im Projekt wurden die Voraussetzungen geschaffen, Diagnoseverfahren im Gesamtsystem einsatzfähig zu machen. Dabei wurde als Zielgröße für die Diagnose des Gesamtsystems maximal eine doppelte Laufzeit gegenüber der Laufzeit der bei Projektstart üblichen Gut/Schlecht-Tests für das Gesamtsystems angestrebt. Basierend auf den neuen Diagnoseergebnissen ist im Gegenzug langfristig die Halbierung der Zeit bis zur vollständigen Fehlerbeseitigung im Gesamtsystem eines der Ziele des Projekts.

#### **IV.1.5. Integration der Diagnoseverfahren in den Halbleiter-Produktionstest**

Die im Projekt entwickelten Diagnoseverfahren werden auch in den Produktionstest der Halbleiterbauelemente integriert. Dies ist notwendig, um zu verhindern, dass Diagnoseläufe im Gesamtsystem Fehler detektierten, die schon im Produktionstest hätten gefunden werden müssen. D.h., der Produktionstest muss eine echte Obermenge auch der im Rahmen der Diagnose verwendeten Tests sein. Es wird angestrebt, dass die neuen Diagnoseverfahren die Produktionstestkosten bei ihrer produktiven Einführung um maximal 5% erhöhen. Langfristig wird angestrebt, optimierte Verfahren zu verwenden, deren Einsatz die Kosten nicht erhöht. Es wird erwartet, dass dieses Ziel durch intelligente Integration der neuen Verfahren in bereits existierende Testverfahren erreicht werden kann.

## V. Ergebnisse

Im Projektverlauf ergab sich eine Präzisierung der Ziele und Randbedingungen:

- Pauschale Integration eines Diagnoseverfahrens für das Gesamtsystem hält einer Wirtschaftlichkeitsbetrachtung nicht Stand
- Konkretes Ziel der Diagnose ist abhängig vom Anwendungsfall (Ort der Diagnose), dessen Randbedingungen müssen definiert sein
- Anwendungsfälle beinhalten jeweils spezifische Herausforderungen, denen die technische Umsetzung begegnen muss
  - Beispiel: Systemverfügbarkeit bei periodischer Diagnose im Betrieb (typisches Verfahren Software-basierter Selbsttest (SBST)) ist zu gewährleisten
- Kombination der Randbedingungen ist Grundlage für die technische Umsetzung
  - Notwendige diagnostische Auflösung (Identifikation eines defekten Steuergeräts, Halbleiters, Gatters oder Transistors)  
Beispiel: Bei der Diagnose in der Werkstatt ist die Identifikation eines defekten Halbleiters nicht sinnvoll, da nur die kleinste in der Werkstatt tauschbare Einheit (in der Regel das Steuergerät), ersetzt werden kann. In der Fehleranalyse dagegen, will man die Fehlerursache verstehen → Identifikation des Fehlerorts (Gatter oder Transistor)
  - Ressourcen (Beispiel: wird im System Speicher für Testdaten/-antworten benötigt, muss der Microcontroller leistungsfähiger sein, als für die Applikation alleine nötig, um SBST ausführen zu können)
  - Zeit (als wichtigste Ressource): für die Diagnose beim Hochlauf/Nachlauf eines Steuergeräts hat man deutlich weniger Zeit, als für die Diagnose in der Werkstatt, bzw. in der Fehleranalyse.
- Ergebnisse aus DIANA orientieren sich an den Anwendungsfällen und halten die jeweiligen Randbedingungen ein
- In DIANA entwickelte Entwurfsmethoden (Aufgabe 4.2) ermöglichen gezielten Systementwurf unter Berücksichtigung der Randbedingungen des Anwendungsfalls.



	Ort	Zeit	Auflösung
 Systementwurf	Entwicklung	Monate	nicht zutreffend
 Diagnose periodisch im Betrieb	Fahrzeug	Abhängig vom Zielsystem	Bauelemente-/Blockebene
 Diagnose im Hochlauf/Nachlauf	Fahrzeug	Hochlauf: 10ms Nachlauf: 5s	Bauelemente-/Blockebene
 Diagnose während Werkstattaufenthalt	Fahrzeug	Sekunden	ECU-Ebene
 Diag. in Fehleranalyse im System	OEM/TIER1 ECU-Ebene	Stunden	Gatter-/Transistorebene
 Diag. in Fehleranalyse auf Bauelementebene	Halbleiterhersteller, IC-Ebene	Stunden	Gatter-/Transistorebene

Abbildung V-1: Übersicht über die Anwendungsfälle und deren Eigenschaften

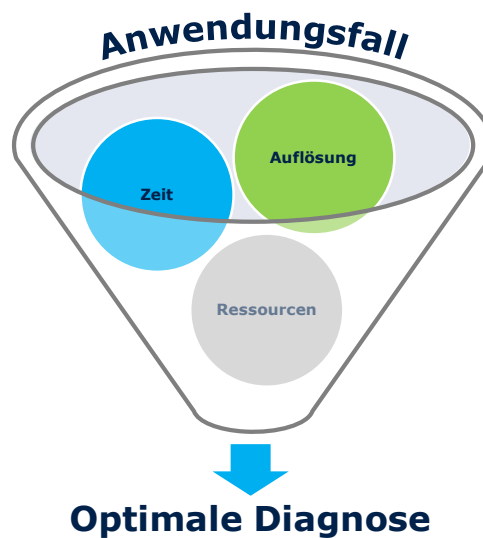


Abbildung V-2: Vom Anwendungsfall zur optimalen Diagnose

Die im Projekt erarbeiteten Technologien sind diesen Randbedingungen unterworfen und bieten eine Lösung für die Projektziele (siehe Tabelle II-1). Die folgende Übersicht zeigt, wie die Erreichten Ergebnisse die zu Beginn des Projekts definierten Herausforderungen adressieren.

### **V.1. Verbesserung der Diagnosequalität in der Automobilwerkstatt**

Erstmalige Bereitstellung technischer Lösungen für die strukturbasierte Diagnose auf Halbleiter- und Steuergeräteebene im Gesamtfahrzeug

- Eingebaute Selbsttests für digitale Logik und Speicher, sowie softwarebasierte Selbsttests<sup>3</sup> (A2.2, A2.3, A4.4)
- Diagnosezugang über Automotive-Standardschnittstellen (A2.3, A3.3)
- Selbsttests für Filterbeschaltungen (A4.3)

### **V.2. Verbesserung der Lieferqualität an OEM und Endkunden**

Neue Diagnoseverfahren erhöhen die Prüfschärfe für Analog- und Mixed-Signal-Schaltungen

- Diagnosemethodik für Analog- und Mixed-Signal-Schaltungen (A1.3)
- Selbsttests für Analog-Digital-Wandler und chipinterne Stimuligenerierung (A2.1)

### **V.3. Verringerung der Entwicklungszeit für neue Fehlersuchverfahren**

Neue Methoden beschleunigen Entwicklung und Bewertung von Diagnoseverfahren

- Analytische Methode zur Modellierung aller zellinternen Fehler, Kompatibilität zu etablierten Fehlermodellen (A1.4)
- Automatisierte Bewertung der Diagnostic Coverage von softwarebasierten Selbsttests (A2.4)

### **V.4. Reduzierung der Zeit für die Diagnose und die Fehlerbeseitigung**

Erstmalige, optimierte Integration von Diagnosemaßnahmen auf Halbleiterebene in das Gesamtfahrzeug

- Bewertung von Diagnosemaßnahmen im Systemkontext (A4.1)
- Gezielte Integration von Diagnoseverfahren in bestehende und zukünftige Systeme (A4.2)

Innovatives Schaltungskonzept für zusätzliche Robustheit in der Applikation

- Selbstreparatur digitaler Logik in Verbindung mit On-Line-Test und Fehlerkorrektur (A2.5)

### **V.5. Integration der Diagnoseverfahren in den Produktionstest von Halbleiterbauelementen**

Bewertung diagnostischer Tests ermöglicht Abgleich mit dem Halbleiter-Produktionstest

---

<sup>3</sup> A<n>.<n> steht hier für Aufgabe<n>.<n>

- Entwicklung der Grundlagen für einen hierarchischen Diagnosezugang und Erstellung von Fehlerbildverzeichnissen auf elektrischer Netzwerkebene (A1.1)
- Hierarchischer Diagnosezugang und Bewertung von Diagnoseprogrammen auf Verhaltensebene (A1.2)

## VI. Ausblick

### VI.1. Potentiale für die Automobilindustrie

- Kürzere Fehlersuchzeiten in der Werkstatt
  - Arbeitszeit ist wichtiger Kostenfaktor für den Kunden (und ggf. auch den Hersteller: Garantiefall, Produktionstest)
- Erhöhte Analysetiefe bereits im Fahrzeug beschleunigt Fehlerabstellprozess
  - Wenn sich Fehler bereits im System beobachten lassen, spart das Zeit (kein Ausbau möglicher fehlerverursachender Komponenten nötig)
  - Erhöht Wahrscheinlichkeit, dass Ursache für den Fehler gefunden werden kann (manche Fehler „verschwinden“ wenn man das System zerlegt), verringert Rate von „kein Fehler feststellbar“
- Verbesserte Echtzeit Diagnose im Betrieb bei verbesserter Prüfschärfe während Steuergerät Hoch-/Nachlauf erhöht die Systemverfügbarkeit
  - Im Automobil geht Sicherheit immer vor Systemverfügbarkeit
  - DIANA bringt bessere Möglichkeiten zur Differenzierung des sicheren Zustands. Diagnosemethoden erlauben eine bessere Unterscheidbarkeit von Fehlern, die einen totalen Ausfall eines Systems bedeuten, von solchen die nur Teile des Systems betreffen. In letzterem Fall kann eine (eingeschränkte) Systemverfügbarkeit gegeben sein: z.B. bei einem Doppelkupplungsgetriebe wird nur ein Getriebeteil abgeschaltet → nur ein Teil der Gänge nutzbar (Fahrt zur Werkstatt möglich)
  - Fehler, die nur unter ganz bestimmten, flüchtigen Umgebungsparametern (Feuchtigkeit, Temperatur, etc.) auftreten, lassen sich leichter eingrenzen
- Bessere Tests im Hochlauf reduzieren die Wahrscheinlichkeit von Steuergerät-Fehlfunktionen während des Fahrbetriebs
  - Sicherheit geht vor
- Erhöhte Prüfschärfe gibt detaillierteres Feedback zum Steuergerätehersteller / OEM
  - Zusätzlich zu den bereits bestehenden Datenquellen (z.B. Daten aus Werkstätten), ist durch DIANA eine neue (detailliertere) Datenquelle über Relevanz von Elektronikfehlern erschließbar.
  - Ferndiagnose über mobile Netze erlaubt Online-Diagnose

### VI.2. Zukünftige Handlungsfelder

- Intelligente Diagnose bildet eine Symbiose zwischen Sicherheit und Systemverfügbarkeit
  - Bessere Differenzierung sicherer Zustände
  - Synergien mit Techniken, die für die Gewährleistung der funktionalen Sicherheit (ISO 26262) notwendig sind
- Evolution industrieller Prozesse
  - Arbeiten im „Spannungsdreieck“ öffnet Horizonte, hebt Synergien und aktiviert Potentiale

- Bisher ist Fehleranalyse ein zeitintensiver Prozess, der seriell die Kette der Zulieferer (OEM→TIER1→weitere Zulieferer) durchläuft.
- Ziel: Partnerschaft; Vernetzung erleichtert Austausch und das Verständnis der Kundenapplikation



Abbildung VI-1: „Spannungsdreieck“ zwischen OEM, TIER1 und Halbleiterherstellern

## Detaillierte Darstellung der Ergebnisse

### VII. Ergebnisse zum Arbeitspaket 1 „Optimierte Fehlermodelle zur effizienten Fehlerlokalisierung“

Ein Ziel der Diagnose besteht darin, durch die Auswertung von fehlerhaft ausgeführten Funktionen, Schlussfolgerungen auf die Art und den Ort eines Defektes auf dem Wafer abzuleiten. Um dieses Ziel zu erreichen, ist es notwendig, den Defekten die geeigneten Testoperationen zuzuordnen. Stochastische Defekte auf dem Wafer, Parameterschwankungen im Herstellungsprozess und die Gestaltung des Layout können Verursacher eines Fehlverhaltens von Schaltkreisen sein. Die Erkennbarkeit eines Fehlverhaltens wird beeinflusst durch die Operationsbedingungen des Schaltkreises, welche wesentlich durch die Betriebsparameter bestimmt werden, und die Genauigkeit der Messgeräte. Die Untersuchung dieser Zusammenhänge ist Gegenstand des Arbeitspaketes 1. Um die Auswirkungen von Defekten auf dem Wafer zu simulieren, wurden Fehler in die Netzlisten injiziert. Unter Verwendung der analogen Fehlersimulation wurde die Eignung von Testsignalen für die Fehlerlokalisierung und die Leistungsfähigkeit von Testverfahren bewertet. Gegenstand der Arbeiten waren sowohl die Durchführung und Auswertung von Messungen im Diagnoselabor, als auch Untersuchungen in den Schaltkreisbeschreibungen Layout, Transistornetzliste mit parasitären Elementen, logische Gatternetzliste und Verhaltensbeschreibung.

#### VII.1. Ergebnisse zur Aufgabe 1.1 „Fehlerlokalisierung in Analog-, Mixed-Signal- und Digitalkomponenten auf Transistorlevel unter Berücksichtigung von Parameterschwankungen“

Im Rahmen der Aufgabe 1.1 wurde anhand von Schaltungsdaten und fehlerbedingten Ausfallsignaturen eine Diagnosestrategie entwickelt, die Parameter- und Layouteinflüsse berücksichtigt. Ziel der Arbeiten ist die Ausarbeitung einer Vorgehensweise, die eine Fehlerlokalisierung auch von übergeordneten Komponenten aus ermöglicht.

Hierzu wurden für repräsentative Schaltungsstrukturen, unter Einbeziehung eines ausgewählten Messverfahrens, Fehlerbild- bzw. Fehlersignaturverzeichnisse erstellt, die wiederum die Grundlage für die in Aufgabe 1.2 durchzuführenden Arbeiten zur Bewertung von Fehlerdiagnoseprogrammen bilden. Bei der Analyse der Fehlerwirkungen werden Prozeßparametereinflüsse berücksichtigt, um Technologiegrenzen sowie Layout bedingte Schwächen erkennbar zu machen und die Ermittlung von Konfidenzbereichen für Testparameter zu ermöglichen.

Für rechen-technische Untersuchungen stand ein analoger Fehlersimulator zu Verfügung.

##### VII.1.1. Auswahl repräsentativer Schaltungsstrukturen

Analog-Digital-Wandler und Digital-Analog-Wandler sind elementare Bestandteile nahezu aller im Automobil eingesetzten Steuergeräte und der darin verbauten integrierten Halbleiterkomponenten. Einige besonders verbreitete Architekturen dieser im Folgenden kurz als AD- bzw. DA-Wandler bezeichneten Komponenten

wurden daher als repräsentative Schaltungsstrukturen für die Untersuchungen zur effizienten Fehlerlokalisierung in Halbleiterbauelementen ausgewählt.

### VII.1.1.1. R2R-DA-Wandler

Bei dieser häufig eingesetzten Form eines DA-Wandlers erfolgt die Umsetzung eines digitalen Wertes in ein analoges Signal mittels eines sogenannten R2R-Widerstandsnetzwerkes. Untersucht wurden Wandler mit Auflösungen von 3 und 5 Bit. Abbildung VII-1 zeigt den schematischen Aufbau eines 5-Bit-R2R-DAC.

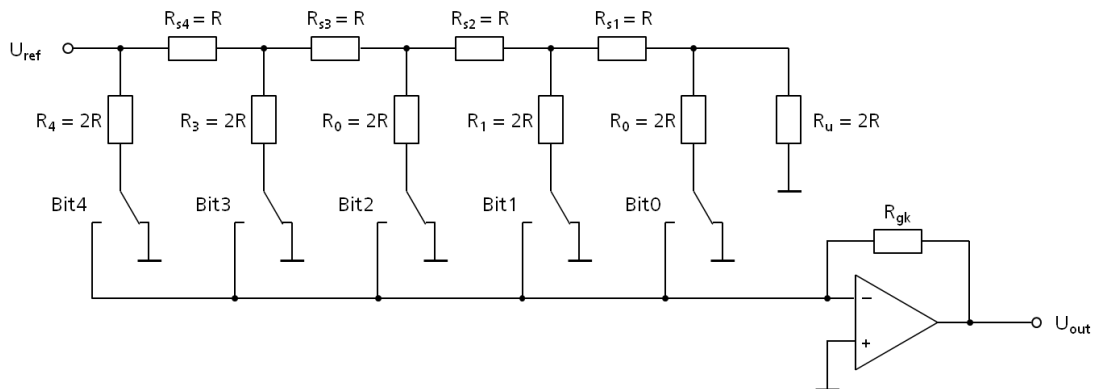


Abbildung VII-1: Schematischer Aufbau eines 5-Bit-R2R-DAC bestehend aus Widerstandsnetzwerk und Operationsverstärker

Als Fehler wurden sowohl resistive Kurzschluss- und Unterbrechungsfehler als auch parametrische Fehler an den Widerständen des R2R-Netzwerkes und des Rückführungswiderstandes des Operationsverstärkers sowie an den Widerständen  $R_1$  und  $R_L$  als auch der Kapazität  $CL$  im verwendeten Operationsverstärker [1] injiziert. Berücksichtigt wurden außerdem resistive Kurzschluss- und Unterbrechungsfehler an den Transistoren des Operationsverstärkers. Insgesamt ergaben sich so 395 Einzelfehler, die mit dem analogen Fehlersimulator [2] und verschiedenen Rampenfunktionen als Stimuli simuliert wurden.

Die berechneten Kennlinien zeigen beispielsweise folgende Fehlerbilder (siehe Abbildung VII-2):

- Anstiegs- (Verstärkungs-) und Offset-Fehler infolge resistiver Kurzschlüsse an  $R_1$  bzw.  $CL$  im Operationsverstärker und parametrischer Fehler an  $R_1$
- Verletzung der Kennlinienmonotonie (Fehler in der Stufung) infolge parametrischer Fehler an Widerständen in der Widerstandskette
- Ungenaue Analogwerte als stufenunabhängige Offsets ohne Monotonieverletzung mit gleichen Höhen der Stufen infolge parametrischer Fehler an  $R_1$  im Operationsverstärker

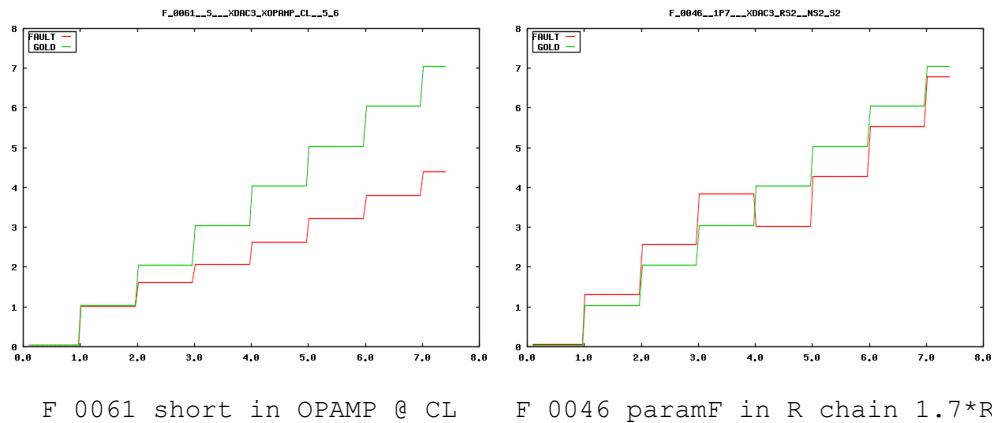


Abbildung VII-2: Beispiele von Kennlinienveränderungen an einem 3-Bit-R2R-DAC

### VII.1.1.2. Flash-AD-Wandler

Eine der wichtigsten Schaltungsarchitekturen zur Umsetzung analoger Signale in digitale Daten ist der sogenannte Flash-Wandler. Hierbei wird jeder mögliche digitale Ausgangswert des Wandlers über einen separat implementierten Komparator detektiert und angezeigt. Untersucht wurden auch hier Wandler mit Auflösungen von 3 und 5 Bit. Abbildung VII-3 zeigt den schematischen Aufbau eines Flash-ADC.

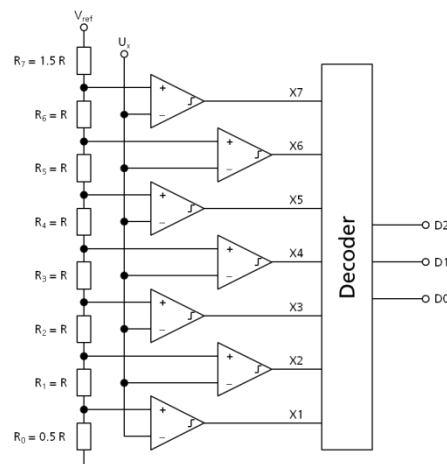


Abbildung VII-3: Schematische Darstellung eines Flash-ADC (hier 3 Bit Auflösung) bestehend aus Widerstandskette, Komparatoren und kombinatorischer Logik

Bei einem 5-Bit-Flash-AD-Wandler wurden als Fehler sowohl resistive Kurzschluss- und Unterbrechungsfehler als auch parametrische Fehler an den Widerständen  $R_0 \dots R_{31}$  der Widerstandskette sowie den Widerständen  $R_1$  und  $R_L$  als auch den Kapazitäten  $CL$  der verwendeten Operationsverstärker [1] injiziert. Resistive Kurzschluss- und Unterbrechungsfehler wurden außerdem an den Komparatorausgängen  $x_1 \dots x_{31}$  und an den Transistoren der Operationsverstärker injiziert. Insgesamt ergaben sich so 9197 Einzelfehler, die mit dem analogen Fehlersimulator simuliert wurden.



Die berechneten Kennlinien zeigen beispielsweise folgende Fehlerbilder (siehe Abbildung VII-4):

- Einzelne fehlende Ausgangswerte (missing codes)
- Mehrere fehlende Ausgangswerte (s. Fehler F\_114 in Abbildung VII-4)
- Anstiegsfehler (s. Fehler F\_131 in Abbildung VII-4)

Die Operationsverstärker sind gegenüber den hier betrachteten parametrischen Änderungen von R1, RL und CL sehr unempfindlich, wohingegen die Widerstandskette des AD-Wandlers auf parametrische Änderungen empfindlich und differenziert reagiert.

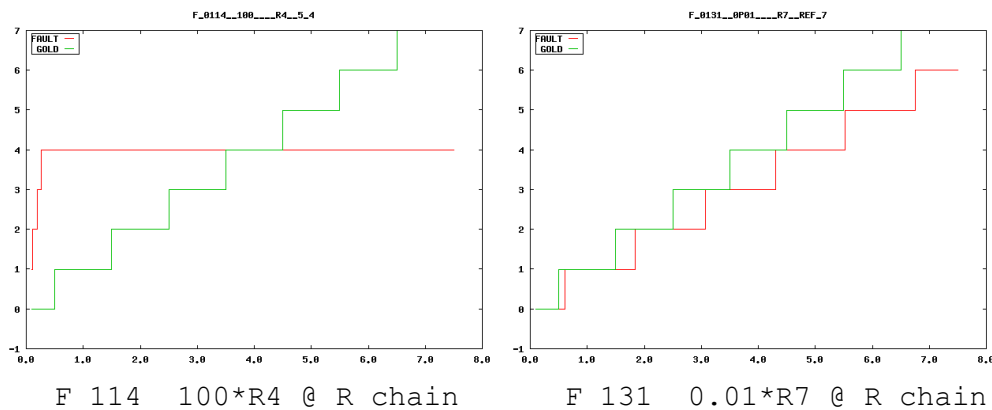


Abbildung VII-4: Beispiele von Kennlinienveränderungen an einem 3-Bit-Flash-ADC

### VII.1.1.3. $\Sigma\Delta$ -AD-Wandler

Auch in der Automobiltechnik werden aufgrund technischer und wirtschaftlicher Vorteile vermehrt  $\Sigma\Delta$ -Wandler eingesetzt. In seiner einfachsten Form besteht dieser Wandlertyp aus einem  $\Sigma\Delta$ -Modulator erster Ordnung mit nachgeschaltetem Digitalfilter. Die Untersuchungen wurden anhand des in Abbildung VII-5 dargestellten  $\Sigma\Delta$  Bit Wandlers [3] durchgeführt.

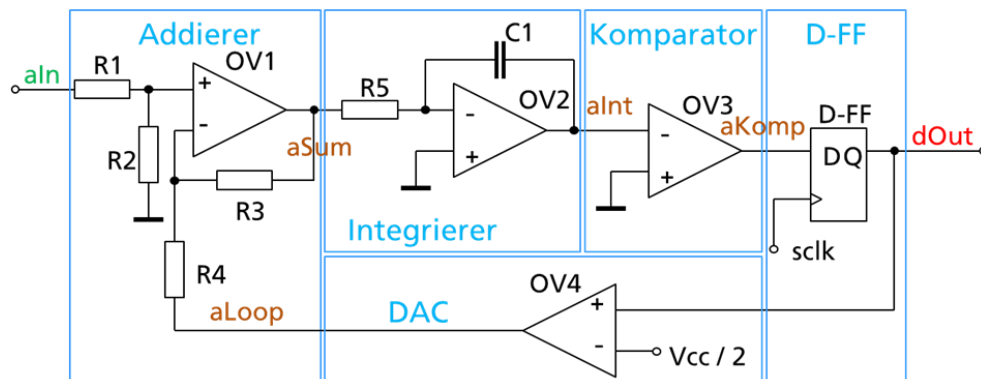


Abbildung VII-5: Blockschaltbild eines  $\Delta\Sigma$ -AD-Wandlers erster Ordnung [3] mit markierten Funktionseinheiten

#### VII.1.1.4. Addierer

Die Funktionseinheit des Addierers wird durch einen mit entsprechenden Widerständen beschalteten Operationsverstärker realisiert.

Als Fehler wurden innerhalb und außerhalb des Operationsverstärkers resistive Kurzschluss- und Unterbrechungsfehler injiziert. Weiterhin sind als parametrische Fehler Abweichungen zu den Nominalwiderständen bzw. –kapazitäten mit den Faktoren von 0,01, 0,2, 0,5, 0,7, 1,3, 1,7, 5, 100, d. h. als Abweichungen zu den Nominalwerten der Widerstände und Kapazitäten von –99%, –80%, –50%, –30%, +30%, +70%, +400%, +9900% modelliert worden.

Insgesamt ergaben sich so 559 Einzelfehler, die mit dem analogen Fehlersimulator simuliert wurden. Die berechneten Fehlerbilder wurden für die Fehlerdiagnose in Fehlerverzeichnissen systematisch erfasst.

Zur Illustration sind in Abbildung VII-6 Stimuli und Ausgangssignale des Addierers für den fehlerfreien Fall verglichen mit drei injizierten Fehlern dargestellt.

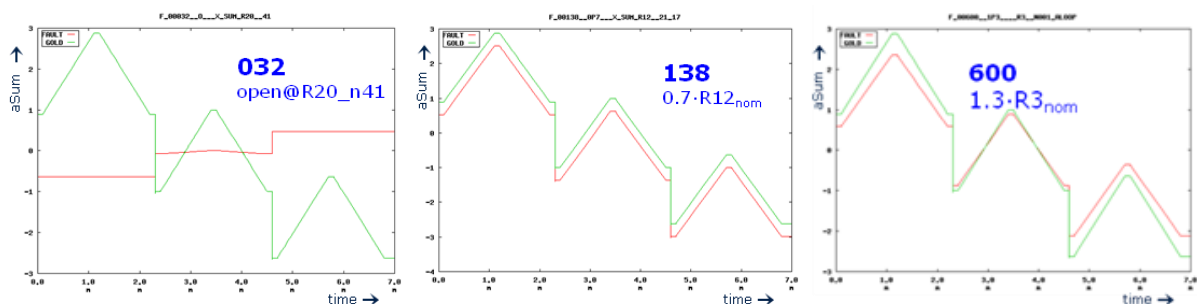


Abbildung VII-6: Ausgangssignalverläufe für drei injizierte Fehler (grün – fehlerfrei, rot – fehlerbehaftet)

#### VII.1.1.5. Integrierer, DAC, Komparator

Auch alle anderen im Blockschaltbild dargestellten Funktionseinheiten werden durch entsprechend beschaltete Operationsverstärker realisiert.

Als Fehler wurden auch hier resistive Kurzschluss- und Unterbrechungsfehler an den Widerständen und an den Kapazitäten sowie an den Transistorterminals injiziert. Als Fehler in der äußeren Beschaltung wurden resistive Kurzschluss- und Unterbrechungsfehler eingefügt.

Für den Integrierer ergaben sich 567, für den DAC 583 und für den Komparator 551 Einzelfehler, die mit dem analogen Fehlersimulator simuliert wurden und deren Fehlerbilder in Fehlerverzeichnissen katalogisiert wurden.

### VII.1.2. Modell für ein Loop-Back-Struktur-gestütztes Meßverfahren

Für die Fehlerdiagnose und -lokalisierung in integrierten Anlogschaltungen werden unterschiedliche, zum Teil sehr aufwendige Techniken, wie Strom-, Spannungs- und Lichtemissions-Meßverfahren eingesetzt. Im Rahmen dieser Arbeit wurde die Anwendbarkeit des für den gemeinsamen DA- / AD-Wandler Test (s. Aufgabe 1.2) entwickelten Loop-Back-Verfahren (s. Abbildung VII-7) als Meßverfahren untersucht. Die Nutzung der Loop-Back-Struktur ermöglicht den Testzugang für eingebettete Anlogschaltungen mit systemeigenen Komponenten ohne zusätzliche analoge Meßtechnik durch meßtechnische Analyse eines DA- / AD-Wandlerpaares.

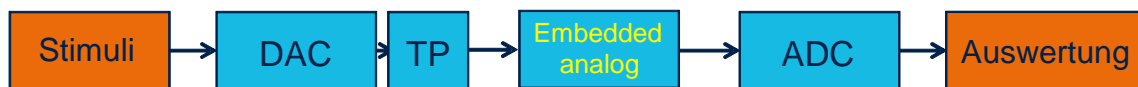


Abbildung VII-7: Loop-Back-Anordnung mit DA- / AD-Wandler zur Analyse eingebetteter Anlogschaltungen

Die dargestellte Loop-Back-Anordnung wurde auf elektrischer Netzwerkebene modelliert und simuliert. Das für das verwendete Meßverfahren als fehlerfrei anzunehmende Übertragungsverhalten der DA- / AD-Wandler wurde anhand der in VII.1.1 ermittelten fehlerfreien Kennlinien abgebildet. Die Funktionsfähigkeit des Ansatzes zur Fehlerdetektion und zum Fehlerverzeichnisaufbau als Grundlage für Fehlerlokalisierungen der eingebetteten Anlogschaltung konnte belegt werden.

### VII.1.3. Hierarchischer Diagnosezugang

Ziel der Diagnosestrategie ist es, anhand von Ausfallsignaturen eine Lokalisierung von Fehlern auch von übergeordneten Komponenten aus zu ermöglichen und das erforderliche Zusammenwirken mit der Aufgabe 1.2 zu gewährleisten.

Aufgrund der bei den ausgewählten, hochkomplexen Schaltungsdesigns zu erwartenden Gegebenheiten und des Simulationsaufwandes ist bei der Fehlerdiagnose zwingend ein hierarchisches, komponentenweises Vorgehen erforderlich. Notwendig ist deshalb auch bei der Fehlerinjektion und Parametervariationen innerhalb der Schaltung die Beschränkung auf eine bzw. wenige Komponenten. Hierarchisch bedeutet hier sowohl unterschiedliche Detaillierungstiefen innerhalb der elektrischen Beschreibungsebene (extrahierte Netzlisten nur für die Komponente, in der eine

Diagnose ausgeführt werden soll, sonst Verwendung von Entwurfsnetzlisten) als auch unterschiedliche Beschreibungsebenen (Abbildung von Fehlerwirkungen von der elektrischen Beschreibungsebene auf die Verhaltensbeschreibungsebene mit MATLAB und ausgehend von Fehlerbildern auf der Verhaltensebene als Repräsentanten für Fehlerbilder einer übergeordneten Komponente Auswahl von Kandidaten für Fehlerursachen auf elektrischer Ebene).

Basierend auf den ausgewählten Schaltungsdesigns und aufgrund der Komplexität wurde wie folgt vorgegangen:

- Fehlersimulationen auf elektrischer Ebene für einzelne Komponenten
- Abbildung der Fehlerauswirkungen als Änderungen in den Kennlinien oder anderen Charakteristika der Komponenten wie veränderte Spezifikationskenngrößen
- Aufbau entsprechender Fehlerlexika, die die Beziehung zwischen Fehlerursache auf elektrischer Ebene und Änderungen von charakteristischen Parametern von MATLAB-Modellen auf Verhaltensbeschreibungsebene dokumentieren
- Fehlersimulationen auf Verhaltensbeschreibungsebene mit MATLAB
- Merkmalsauswahl und Merkmalsberechnung
- Aufbau von Fehlerlexika bez. ausgewählter Merkmale
- Auswahl von Kandidaten für Fehlerursachen bez. MATLAB-Simulationen
- Rückschluss auf / Auswahl von Kandidaten für Fehlerursachen bez. der auf elektrischer Ebene simulierten Komponenten durch Ähnlichkeitsvergleiche von Fehlerbildern

#### **VII.1.4. Ermittlung von Konfidenzbereichen für Testparameter**

Der Zusammenhang zwischen dem Konfidenzintervall einer Testspezifikation, einem Guardband, dem Konfidenzintervall der Ausgangskenngröße der fehlerbehafteten Schaltung und der Detektierbarkeit von Fehlern ist im Rahmen dieser Arbeit herausgearbeitet worden.

Führt ein injizierter Fehler bei nominalen Prozessparametern zu einer größeren Abweichung vom Erwartungswert  $\mu_g$  des Testparameters (z. B.  $> |6 \cdot \sigma_g|$ ), so wird der injizierte Fehler als „detektiert“ deklariert.

Für das Beispiel in Abbildung VII-8 sind das alle Fehler mit Werten, die außerhalb der Toleranzintervalle  $[LTL, UTL] = [t_{\min\_3\sigma_g}, t_{\max\_3\sigma_g}] = [0.7, 1.3] = [1.0 \pm 0.3]$  liegen.

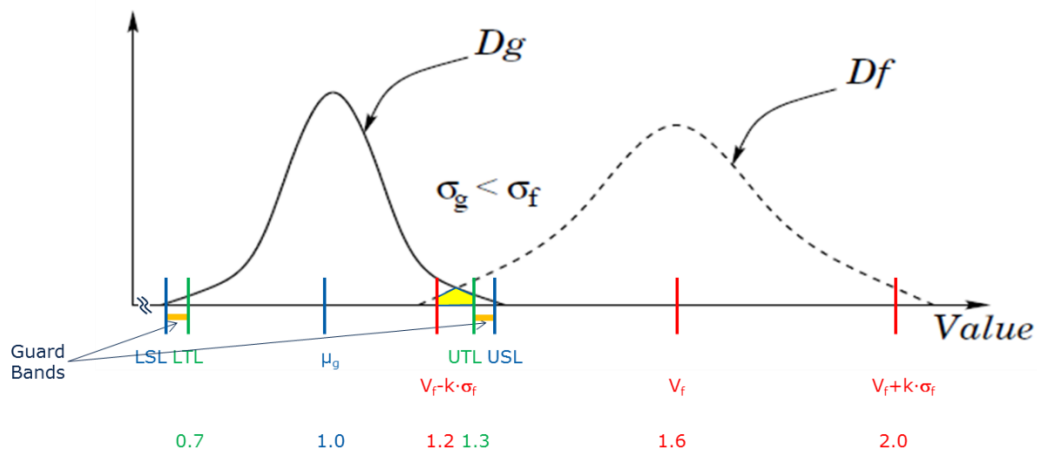


Abbildung VII-8: Darstellung zu den Zusammenhängen zwischen Spezifikationsgrenzen, Toleranzgrenzen und Guardband (Begriffserklärungen s. Abbildung VII-10)

Für Fehler, die zu einer kleineren Abweichung (z. B.  $< |3 \cdot \sigma_g|$ ) führen, kann anhand der Toleranzintervalle für den fehlerfreien und den jeweiligen Fehler-Fall als Maß für seine Detektierbarkeit (DM) eine Wahrscheinlichkeit für die jeweilige Fehlerdetektierbarkeit, (probability of detection of fault – POD [6]) oder ein Detektierbarkeitsgrad (detectability degree – DD [7]) bestimmt werden.

So würde ein Fehler, der eine Abweichung von  $+3\sigma_g$  vom erwarteten Erwartungswert  $\mu_g$  hat (also genau auf dem upper-tolerance-limit UTL von  $\mu_g + 3\sigma_g$  liegt) für ein Konfidenzintervall von  $(-3\sigma_g, +3\sigma_g)$  ( $\cong 93.3\%$  yield) einen Detektierbarkeitsgrad von 50% haben (50% des Konfidenzintervalls der Ausgangskenngröße der fehlerbehafteten Schaltung liegen außerhalb des Konfidenzintervalls, das durch den Toleranzbereich aufgespannt wird – Überlappung im Bereich von  $[\mu_g, \mu_g + 3\sigma_g]$  (s. Abbildung VII-9)). Für ein Konfidenzintervall von  $(-6\sigma_g, +6\sigma_g)$  ( $\cong 99.99966\%$  yield) hätte dieser Fehler einen Detektierbarkeitsgrad von 25% (25% des Konfidenzintervalls der Ausgangskenngröße der fehlerbehafteten Schaltung liegen außerhalb des Konfidenzintervalls, das durch den Toleranzbereich aufgespannt wird – Überlappung im Bereich von  $[\mu_g - 3\sigma_g, \mu_g + 6\sigma_g]$  (s. ebenfalls Abbildung VII-9)).

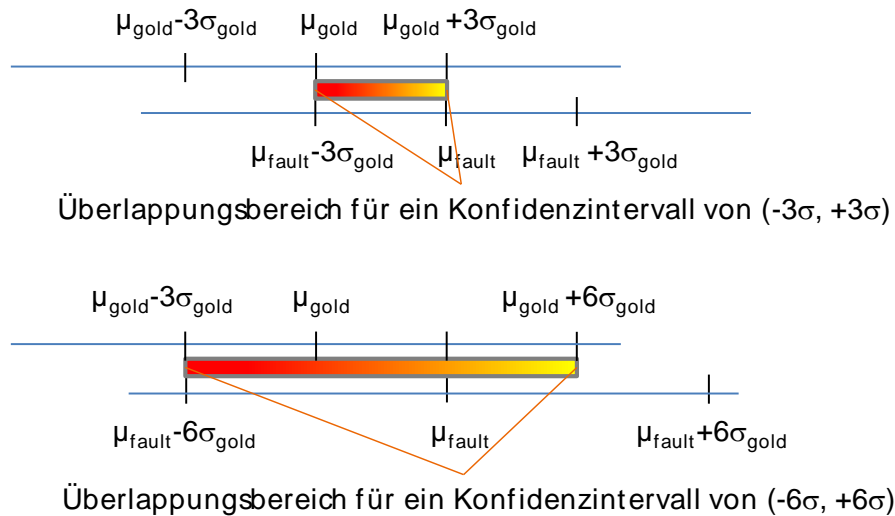


Abbildung VII-9: Beispiele für Überlappungsbereiche bei unterschiedlichen Konfidenzintervallen

Damit ist ein Zusammenhang zwischen dem Konfidenzintervall der Spezifikation, der Lage des Konfidenzintervalls der Ausgangskenngröße der fehlerbehafteten Schaltung und somit dem Detektierbarkeitsgrad eines Fehlers hergestellt –  $DM_f = f(\mu_g, \sigma_g, \mu_f, \sigma_f, k, \varepsilon)$ .

In Abbildung VII-10 ist der Gesamtablauf zur Bestimmung von Detektierbarkeitsmaßen (Überlappungsbereichen) und ihr Zusammenhang zur Guardband-Veränderung und damit der Änderung der Toleranzintervalle und somit auch der Konfidenzbereiche dargestellt.

Dabei bedeuten:

$V_g$  – konkreter (simulierter oder gemessener) Wert des Ausgangs der fehlerfreien Schaltung

$V_f$  – konkreter (simulierter oder gemessener) Wert des Ausgangs der mit dem Fehler  $f$  behafteten Schaltung

$\mu_g$  – Erwartungswert des Ausgangs der fehlerfreien Schaltung

$\sigma_g$  – Standardabweichung des Ausgangs der fehlerfreien Schaltung

$\mu_f$  – Erwartungswert des Ausgangs der mit dem Fehler  $f$  behafteten Schaltung

$\sigma_f$  – Standardabweichung des Ausgangs der mit dem Fehler  $f$  behafteten Schaltung

$k$  – „Effektivitätsfaktor“ eines Tests ( $k=2 \rightarrow 2\sigma_g$ ;  $k=5 \rightarrow 5\sigma_g$ )

$k$  niedrig: Wahrscheinlichkeit, dass eine fehlerfreie Schaltung als fehlerhaft deklariert wird, ist hoch

$k$  hoch: Wahrscheinlichkeit, dass eine fehlerfreie Schaltung als fehlerhaft deklariert wird, ist gering

$\varepsilon$  – Guard Band – Sicherheitsbereich zur Verschärfung eines Pass/Fail-Limits; Bereich um den ein „specification limit“ reduziert wird, so, dass Pass/Fail-Entscheidungen mit höherer Konfidenz getroffen werden

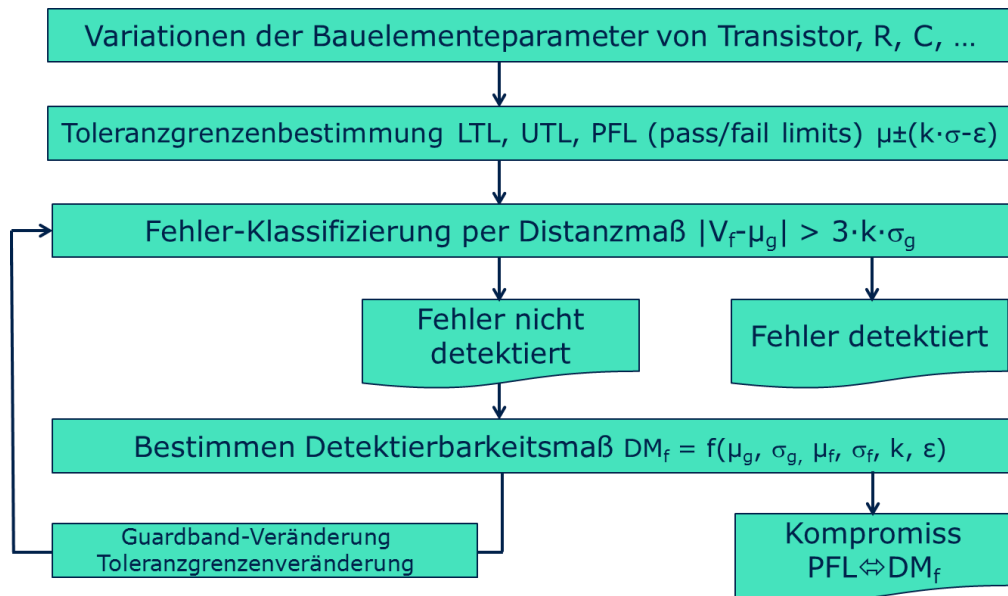


Abbildung VII-10: Gesamtablauf zur Bestimmung von Detektierbarkeitsmaßen (Überlappungsbereichen) und ihr Zusammenhang zur Guardband-Veränderung und damit der Änderung der Toleranzintervalle und somit auch der Konfidenzbereiche

#### VII.1.4.1. Berechnungen für ein Schaltungsbeispiel

Als Beispielschaltung ist ein Spannungsteiler mit nachgeschaltetem Spannungsfollower ausgewählt worden. Die Schaltung wurde einer Monte-Carlo-Simulation unterzogen, wobei als auszuwertende Kenngröße (Testparameter) die Slew-Rate des Operationsverstärkers ausgewählt wurde.

Abbildung VII-11 zeigt den Zusammenhang zwischen dem Konfidenzintervall der Spezifikation, der Lage des Konfidenzintervalls der Ausgangskenngröße der fehlerbehafteten Schaltung und somit dem Detektierbarkeitsgrad eines Fehlers. Am Beispiel wird deutlich, wie für diesen relativ nahe zum Nominalfall liegenden Fehler eine gute Fehlerdetektierbarkeit nur unter Inkaufnahme von hohen Ausbeuteverlusten erfolgen kann.

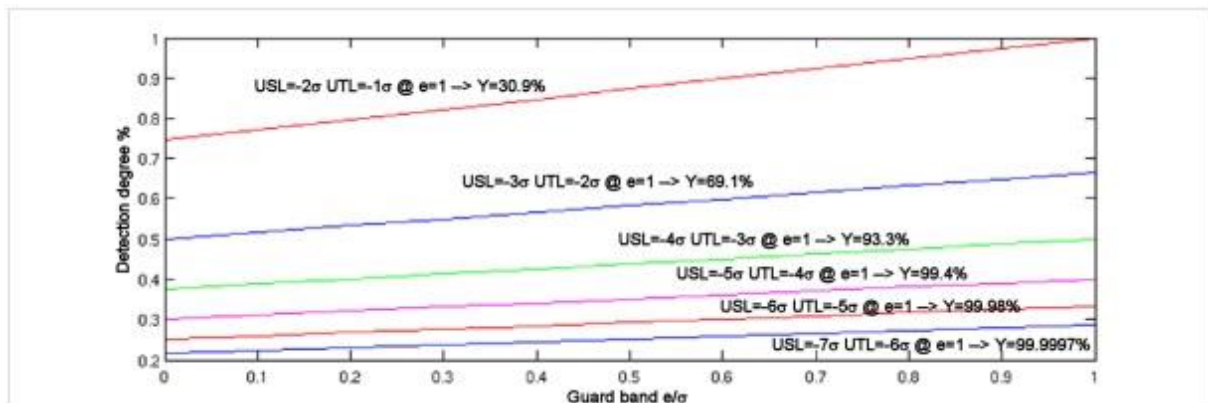


Abbildung VII-11: Zusammenhang zwischen dem Konfidenzintervall der Spezifikation, der Lage des Konfidenzintervalls der Ausgangskenngröße der fehlerbehafteten Schaltung und somit dem Detektierbarkeitsgrad eines Fehlers für das Schaltungsbeispiel

Als Beispiel sei der grüne Kurvenverlauf erörtert. Er zeigt, dass bei einem USL von  $USL = \mu_g - 4\sigma_g$  je nach Guardband ( $0 \dots 1\sigma_g$ ) nur ein Detektierbarkeitsgrad von 38% bis 50% erreicht werden kann, d. h. nur 38% bis 50% des Konfidenzintervalls der Ausgangskenngröße der fehlerbehafteten Schaltung liegen außerhalb des Konfidenzintervalls, das durch den Toleranzbereich von  $LTL = [\mu_g - 3\sigma_g, UTL = \mu + 3\sigma_g]$  aufgespannt wird.

### VII.1.5. Erfassung von *technologie- und layoutbedingten Schwächen*

Ein bedeutender Bestandteil der Fehlerdiagnose ist die Erfassung technologie- und layoutbedingter Schwächen. Hierzu zählen beispielsweise:

- Lange parallele Leitungen (kritische Leitungsparallelitäten)
- Einzelkontakte
- Lange Poly-Leiterzüge, die einen wesentlich höheren Widerstand haben als Metal-Leitungen
- Matching-Probleme
- Großflächige Kreuzungen von Leiterbahnen
- Ungenügende Angaben zur Stromtragfähigkeit, und damit ungenügende Abschätzbarkeit von Elektromigrationsauswirkungen

Im Rahmen der Aufgabe 1.1 wurden Untersuchungen zur Stromtragfähigkeit von Bauelementeanschlüssen und zur Bestimmung kritischer Leitungsparallelitäten durchgeführt.

#### VII.1.5.1. *Abschätzung der zu erwartenden Stromtragfähigkeit von Bauelementeanschlüssen*

Die Elektromigrationsanfälligkeit von Schaltungen kann durch eine Abschätzung der zu erwartenden Stromtragfähigkeit von Bauelementeanschlüssen beurteilt werden.



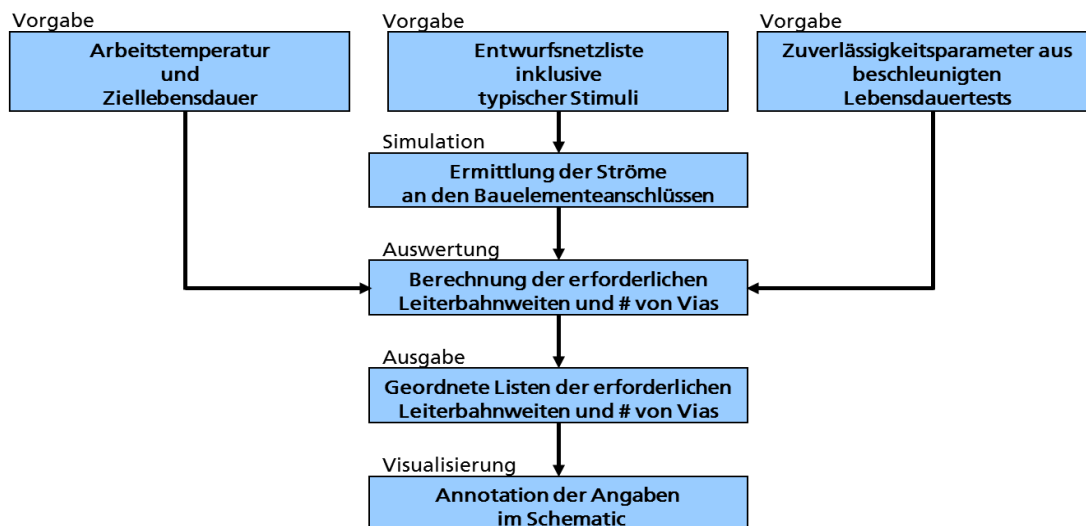
### VII.1.5.2. Prinzipielle Vorgehensweise

Unter Zuhilfenahme von Informationen aus beschleunigten Stresstests und in Abhängigkeit von simulierten typischen Stromverläufen für Anschlüsse von Bauelementen können für vorgebbare Zeiträume und Temperaturen layerbezogen vor einer Layoutgenerierung die Layout-Constraints „Minimale Leiterbahnweiten“ und „Minimale Anzahl von Vias/Contacts durch die Berechnung der erforderlichen Leiterbahnbreiten bzw. Via/Contact-Dimensionen bestimmt werden.

Neben dem Feststellen von Limitverletzungen kann damit auch ein systematischer Zugang für eine Layoutoptimierung unterstützt werden.

Die prinzipielle Vorgehensweise ist in Abbildung VII-12 grafisch dargestellt. Eine ausführlichere Darstellung der Vorgehensweise zu einer in einem anderen Projekt entwickelten Pre-Layout–Elektromigrations-Analyse ist in [8] enthalten.

#### Vorgehensweise bei der EAS\*-Pre-Layout-Elektromigrations-Analyse



\* Fraunhofer-Institut für Integrierte Schaltungen  
Institutsteil Entwurfsautomatisierung

Abbildung VII-12: Ablaufplan zur prinzipiellen Vorgehensweise der Prä-Layout–Elektromigrations-Analyse

### VII.1.5.3. Beispiel

Eine Prä-Layout-Elektromigrations-Analyse wurde auf eine Testbench (s. Abbildung VII-13) zur Bestimmung der Slew Rate eines Operationsverstärkers angewendet, um erforderliche minimale Leiterbahnweiten sowie die notwendige minimale Anzahl von Vias bzw. Kontakten zu bestimmen.

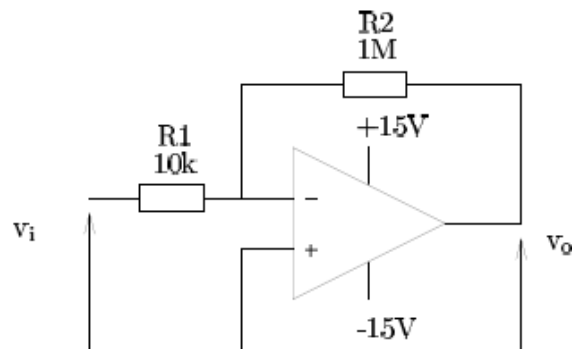


Abbildung VII-13: Testbench-Struktur zur Slew Rate eines Operationsverstärkers nach [3]

Als Resultat konnten für die einzelnen Layer die entsprechenden Limits für Leiterbahnweiten bzw. Vias bestimmt werden. Tabelle VII-1 zeigt eine Auswahl aus den Limits für Leiterbahnweiten.

Tabelle VII-1: Auswahl aus den berechneten Limits für Leiterbahnweiten, die auf der Basis einer Prä-Layout-Elektromigrations-Analyse berechnet wurden

OpAmp				
Structure Metal_1   Oper.Temp. 125 °C   Life Time 100.000 h				
Rank	Terminal	Label 1	Maximum current / A	Minimum linewidth / m
1	/V_in/MINUS	peak	-3.2773e-04	4.3536e-07
5	/I57/in	peak	-2.9094e-04	3.8649e-07
19	/I57/vdd	peak	5.3266e-05	7.0759e-08

#### VII.1.5.4. Beurteilung

Mit einer *Prä-Layout*-Elektromigrations-Analyse können Verletzungen der Layout-Constraints „Minimale Leiterbahnweiten“ und „Minimale Anzahl von Vias/Contacts“ in einer frühen Phase des Design-Flows aufgedeckt werden.

Eine Erweiterung der Anwendung des vorgestellten Zugangs auf den *Post-Layout*-Fall, bei dem parasitäre Widerstände als die Bauelemente betrachtet werden, deren Anschlüsse bezgl. der Stromtragfähigkeit beurteilt werden, würde einen Zwischenschritt zu einer direkten layout-basierten Elektromigrations-Analyse, z. B. nach [5], darstellen.

Nach einer solchen Erweiterung können dann Entwurfsschritte effektiviert werden und die Robustheit einer Schaltung gegen Elektromigration noch gezielter beurteilt werden. Damit sind elektromigrationsbezogenen Schwachstellen in einer Schaltung anhand der Lage der parasitären Widerstände lokalisierbar, ohne eine sehr aufwendige direkte layout-basierte Elektromigrations-Analyse durchführen zu müssen.

### **VII.1.5.5. Netzwerksimulationen zur Bestimmung relevanter Schaltungsparameter**

Relevante Schaltungsparameter, die auf technologie- und layoutbedingte Schwächen hindeuten, können auch mit Hilfe von Netzwerksimulatoren bestimmt werden. Solche Möglichkeiten sind z. B. Monte-Carlo-Simulationen oder Sensitivitätsanalysen.

Die Resultate hängen jedoch von den vorgegebenen Stimuli und den auszuwertenden Kenngrößen (Testparameter) ab. Im Fehlerfall ist die Parameterrelevanz zudem von dem jeweiligen betrachteten Fehler abhängig.

### **VII.1.5.6. Bestimmung kritischer Leitungsparallelitäten**

Durch die Auswertung von Informationen zur Lage der aus einem Layout extrahierten Koppelkapazitäten und ihren Kapazitätswerten können unter Nutzung eines im Rahmen der Aufgabe 1.3 des Arbeitspaketes 1 entwickelten Zugangs zur Ermittlung langer paralleler Leitungen Schwachstellen bez. kritischer Leitungsparallelitäten bestimmt werden.

Im Rahmen des Softwarepaketes zur analogen Fehlersimulation wird dazu zunächst eine geordnete Liste von Koppelkapazitäten erstellt. Auf deren Basis können dann entsprechende Visualisierungen im Cadence Layout-Editor erfolgen.

Tabelle VII-2 zeigt ein Beispiel für eine geordnete Liste von Koppelkapazitäten für einen Operationsverstärker eines analogen Test-Busses.

Tabelle VII-2: Beispiel für eine geordnete Liste von Koppelkapazitäten

OpAmp			
coupling capacities			
ident	value	node 1	node 2
c55	1.456E-15	en_i	net134
C116	1.364E-15	net61	Net134
c128	1.233E-17	net61	net132
c24	9.142E-19	ib1u25i	enb_i

Aus der Bestimmung kritischer Koppelkapazitäten, die Anhaltspunkte auf lange parallele Leitungen geben, lassen sich dann Hinweise zur Entwurfsverbesserung hinsichtlich einer Zuverlässigkeitserhöhung von Schaltungen ableiten.

### **VII.1.6. Demonstrator-Board DA/AD-Wandler**

Die Zielerreichung der Aufgaben 1.1 und 1.2 konnte mit einem Demonstrator-Board DA/AD-Wandler nachgewiesen werden. Der Demonstrator besteht aus den in VII.1.1 beschriebenen DA- / AD-Wandlern, die einzeln oder in einer Loop-Back-Anordnung betrieben werden können (siehe Abbildung VII-14)

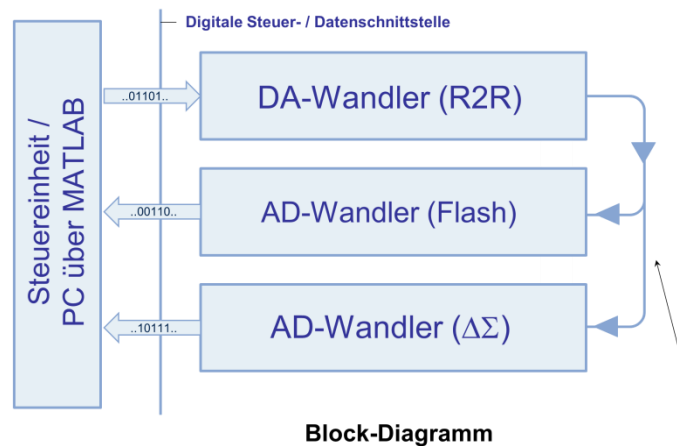


Abbildung VII-14: Schematische Darstellung des Board-Demonstrators

Die Komponenten des Demonstrator-Boards DA/AD-Wandler wurden auf elektrischer Netzwerkebene modelliert. MATLAB- und LTSpice Modelle wurden für die Gesamtschaltung entwickelt. Entsprechend der Aufgabe 1.1 wurden die Einzelkomponenten auf Netzwerkebene simuliert und mit dem analogen Fehlersimulator einer Fehlersimulation unterworfen. Als Fehler wurden Kurzschlüsse und Unterbrechungen sowie parametrische Fehler betrachtet. Entsprechende Fehlerbildverzeichnisse wurden erstellt. Die Ergebnisse der Netzwerksimulationen werden als Referenz für die MATLAB-Simulationen entsprechend der Aufgabe 1.2 verwendet. Sie ermöglichen die Berechnung der Übertragungsfunktionen für die Modellkomponenten in der MATLAB-Umgebung und bilden so die Grundlage für den hierarchischen Diagnosezugang (siehe VII.1.3).

### VII.1.7. Zusammenfassung

Im Rahmen der Aufgabe 1.1 wurden Fehlersimulationen für die Komponenten ausgewählter repräsentativer Schaltungsstrukturen, Arbeiten zur Einbeziehung von Modellen für Messverfahren in das Diagnoseverfahren, die Ermittlung von Konfidenzbereichen für Testparameter und Analysen zu technologie- und layoutbedingten Schwächen durchgeführt. Die Zielerreichung wurde mit einem Demonstrator-Board DA/AD-Wandler nachgewiesen.

Für die Komponenten der ausgewählten Schaltungsdesigns wurden umfangreiche Fehlersimulationen durchgeführt. Basierend auf den Entwurfsnetzlisten wurde so die Erstellung von Fehlerbild- und Fehlersignaturverzeichnissen für die zu diagnostizierenden Komponenten an R2R-, Flash- und Delta-Sigma-Wandlern erster Ordnung durchgeführt. Dies geschah auf der Basis von Fehlersimulationen für die Komponenten separat und innerhalb einer Loop-Back-Anordnung unter Berücksichtigung der durch die Schwächenanalyse ermittelten Fehlerarten. Dabei ergaben sich bei Fehlersimulationen auf der elektrischen Beschreibungsebene und bei MATLAB-Fehlersimulationen adäquate Resultate.

Als Messverfahren ist eine ursprünglich für den gemeinsamen DA- und AD-Wandler Test ausgelegte Loop-Back-Struktur zur Analyse des Verhaltens eingebetteter Analogschaltungen untersucht worden. Anhand einer Beispielschaltung konnte mit Hilfe von Simulationen das untersuchte Messverfahren hinsichtlich seiner Leistungsfähigkeit einer ersten Einschätzung unterzogen werden und entsprechende Aussagen abgeleitet werden.

Der Zusammenhang zwischen dem Konfidenzintervall einer Spezifikation, der Lage des Konfidenzintervalls der Ausgangskenngroße der fehlerbehafteten Schaltung und Detektierbarkeitsgraden von Fehlern ist im Rahmen der Untersuchungen zu Konfidenzbereichen für Testparameter herausgearbeitet worden. Anhand einer Beispielschaltung konnten diese Zusammenhänge validiert und quantifiziert werden.

Analysen zu technologie- und layoutbedingten Schwächen beinhalteten die Abschätzung der Stromtragfähigkeit von Bauelementeanschlüssen innerhalb einer Schaltung, die Ermittlung besonders relevanter Schaltungsparameter mittels Netzwerksimulationen und die Bestimmung kritischer Leitungsparallelitäten. Die prinzipiellen Vorgehensweisen für die obigen Analysen sind dargelegt worden. Konkrete Schwachstellen konnten anhand von Beispielen exemplarisch benannt werden.

Die erste Stufe des durch Aufgabe 1.1 und 1.2 gemeinsam getragenen hierarchischen Diagnosezugangs, die aus Modellierungen und Fehlersimulationen von Komponenten auf elektrischer Netzwerkebene besteht, ist exemplarisch für einen im Rahmen des Arbeitspaketes 1 erstellten Demonstrator-Board DA/AD-Wandler und dessen Komponenten realisiert worden. Dabei sind als Fehler Kurzschlüsse und Unterbrechungen sowie parametrische Fehler betrachtet worden und es wurden entsprechende Fehlerbildverzeichnisse erstellt. Es konnte gezeigt werden, dass diese Ergebnisse als Referenz für die MATLAB-Simulationen entsprechend der Aufgabe 1.2 geeignet sind und so für den hierarchischen Diagnosezugang grundlegende Bedeutung haben.

### VII.1.8. Literatur

[1] Kaminska, B., Arabi, K., Bell, I., Goteti, P., Huertas, J.L., Kim, B., Rueda, A., Soma, M., Analog and Mixed-Signal Benchmark Circuits - First Release, ITC, pp.183, International Test Conference 1997 (ITC'97), 1997.

[2] Hopsch, F.; Lindig, M.; Straube, B.; Vermeiren, W.: "Characterization of digital cells for statistical test," Proc. IEEE 14th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS), April 2011, pp. 255 -260.

[3] Desouky, A.O.; Khairy, M.S; Abdelhalim, M.B.; Amer, H.H., Low-cost test of the first-order Delta-Sigma converter, Computer Engineering & Systems 2009, pp. 100-105

[4] <http://users.skynet.be/hugocoolens/spice/ua741/ua741.htm>

[5] [J. Lienig, G. Jerke: Current-Driven Wire Planning for Electromigration Avoidance in Analog Circuits. Proceedings of the 8th Asia and South Pacific Design Automation Conference \(ASP-DAC\), 2003, pp. 783-788](#)

- [6] S.J.Spinks, C.D.Chalk, I.M.Bell, M.Zwolinski: Generation and Verification of Tests for Analogue Circuits Subject to Process Parameter Deviations. Generation and Verification of Tests for Analog Circuits Subject to Process Parameter Deviations, *IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, Paris, France, Oct. 1997 ([DFT 1997](#)): pp. 100-109, *Journal of Electronic Testing Theory and Applications*, 20(1), pp. 11-23
- [7] A. Khouas, [A. Derieux](#): Fault Simulation for Analog Circuits Under Parameter Variations. *J. Electronic Testing* 16 (3): 269-278 (2000)
- [8] Verbundprojekt A18HVMOS (Projektträger Sächsischen Aufbaubank); 2007

## VII.2. Ergebnisse zur Aufgabe 1.2 „Bewertung von Diagnoseprogrammen bezüglich der Fehlerlokalisierung und des Einflusses von Veränderungen im Herstellungsprozess“

Das Arbeitspaket 1 des Projektes DIANA mit dem Titel „Optimierte Fehlermodelle zur effizienten Fehlerlokalisierung in Halbleiterbauelementen“ beinhaltet Arbeiten zur Vorbereitung der Fehlerdiagnose in Halbleiterbauelementen. Dabei werden die Auswirkungen von Produktionsdefekten und Parameterschwankungen auf das Verhalten der zu diagnostizierenden Schaltungen simuliert. Zu diesem Zweck werden Fehlermodelle und Diagnoseprogramme auf ihre Eignung zur Defektlokalisierung untersucht.

Die Aufgaben 1.1 und 1.2 sind stark aufeinander bezogen. Während Aufgabe 1.1 die Fehlerlokalisierung in elektrischen Netzwerken von Analog- und Mixed-Signal-Komponenten unter Berücksichtigung von Parameterschwankungen untersucht, befasst sich Aufgabe 1.2 mit der Bewertung von Diagnoseprogrammen bezüglich der Fehlerlokalisierung und des Einflusses von Veränderungen im Herstellungsprozess.

Dabei bilden die Kennlinien von ADC und DAC die Schnittstellen. Sie entstehen als Ergebnis der elektrischen Netzwerksimulation unter dem Einfluss von Fehlern und Parameterschwankungen und stellen eine Eingangsinformation für den Loop-Back-Test dar. Im Rahmen der Einbeziehung von Delta-Sigma-ADCs ( $\Delta\Sigma$ -ADCs) wurde eine hierarchische Herangehensweise entwickelt, bei der Komponenten des  $\Delta\Sigma$ -ADCs der Netzwerksimulation unterworfen werden und die Ergebnisse direkt in den Loop-Back-Test einfließen. Dadurch werden die Fehlersimulationen größtenteils auf die MATLAB-Ebene verlagert und die Berechnung von Fehlersignaturen [5] effektiver. Diese Fehlersignaturen bilden die Grundlage einer wirksamen Fehlerdiagnose.

Als repräsentatives Beispiel wurde das Verfahren des Tests von Digital-Analog-Wandlern (DAC) und Analog-Digital-Wandlern (ADC) in einer Loop-Back-Struktur ausgewählt. In Erweiterung der Aufgabenstellung werden die Untersuchungen auf  $\Delta\Sigma$ -ADCs ausgedehnt.

Für die Fehlerdiagnose wurde eine hierarchische Herangehensweise umgesetzt. Flash-ADCs und R2R-DACs können auf Netzwerkebene als Komponente simuliert und fehlersimuliert werden. Die Ergebnisse werden in Form von Kennlinien

übernommen, die Simulationen in der Loop-Back-Struktur und die Fehlerdiagnose auf der Verhaltensebene ausgeführt und die Diagnoseergebnisse in die Netzwerkebene zurücktransformiert. Da  $\Delta\Sigma$ -ADCs aufwändige Netzwerksimulationen erfordern, wurde die hierarchische Herangehensweise verfeinert. Auf Netzwerkebene werden Komponenten wie Addierer, Integrierer und Komparatoren bearbeitet, die Simulationsergebnisse in Parameter für die Funktionen der Komponenten transformiert und schließlich auf Verhaltensebene als Funktionen der Komponenten in den Simulationen ausgeführt. Auf Verhaltensebene sind dadurch dynamische Tests möglich, wie sie zum Test von  $\Delta\Sigma$ -ADCs üblich sind [9].

## VII.2.1. Hierarchische Diagnosestrategie

### VII.2.1.1. Konzeption der Diagnosestrategie

Elektrischen Netzlisten beschreiben elektrische Schaltungen mit einem hohen Detaillierungsgrad. Dadurch ist eine größere Nähe zum Layout möglich und Defekte können als Fehler abgebildet werden. Diese Fehler sind wiederum Grundlage für Fehlersimulation und Fehlerdiagnose. Ebenso können Parametervariationen auf Netzwerkebene gut realisiert werden.

Der hohe Detaillierungsgrad bewirkt jedoch einen großen Aufwand für die Simulation auf Netzwerkebene. Umfangreiche Fehlersimulationen und Simulationen mit unterschiedlichen Parametern sind deshalb auf dieser Ebene nicht möglich.

Einen Ausweg bietet ein hierarchisches Vorgehen, in dem Netzwerk- und Verhaltensebene einbezogen werden:

- Die zu untersuchende Schaltung wird in Komponenten zerlegt, die zum einen eine auf Netzwerkebene gut handhabbare Größe und zum anderen gut definierbare Schnittstellen besitzen. Diese Schnittstellen müssen so beschaffen sein, dass die Werkzeuge der verschiedenen Ebenen über diese gut miteinander kommunizieren können.
- Für jede Komponente werden auf Netzwerkebene die Fehler in Form von Fehlerlisten mit Fehlernummer, Fehlerort und Fehlerwert definiert und die Fehlersimulationen sowie Simulationen mit unterschiedlichen Parametern ausgeführt. Ergebnis dieser Simulationen sind Dateien, die das Ein-/Ausgabeverhalten jeder Komponente beschreiben.
- Auf Verhaltensebene wird die Gesamtsimulation ausgeführt. Zunächst werden die umfangreichen Ergebnisse der Netzwerksimulationen aller Komponenten in Dateien konvertiert, die nur noch die für die Simulation auf Verhaltensebene wesentlichen Informationen enthalten, d. h., die Komponenten werden durch Kennlinien oder Funktionen charakterisiert. Dieses Vorgehen bewirkt eine erhebliche Reduzierung der Datenmenge, ohne die Genauigkeit der Simulationen deutlich zu verschlechtern. Durch die reduzierte Datenmenge ist ein Vergleich der Kennlinien oder Funktionen möglich, in dessen Ergebnis äquivalente Fehler erkannt und entfernt werden können.
- Durch dieses Vorgehen werden die Ergebnisse der Netzwerksimulationen aller Komponenten in das Gesamtsimulationsmodell auf Verhaltensebene eingebunden. Außerdem werden Referenzen zu den Fehlerlisten hergestellt.

Die Gesamtsimulation berücksichtigt damit Fehler und Parameterschwankungen, die auf Netzwerkebene beschrieben sind.

- Ergebnis jeder Simulation der Gesamtschaltung ist das Ein-/Ausgabeverhalten der Schaltung. Da die Daten, die dieses Verhalten beschreiben, auch auf Verhaltensebene sehr groß sind, werden aus ihnen Kenngrößen berechnet und zu einer Fehlersignatur zusammengefasst. Die Fehlersignatur soll den Fehler möglichst eindeutig charakterisieren, aber auch Parametervariationen abbilden.

### VII.2.1.2. Umsetzung der Diagnosestrategie für eine Loop-Back-Struktur aus Digital-*Analog- und Analog-Digital-Wandlern*

Korrespondierend zu repräsentativem Beispiel und dem zu untersuchenden Diagnoseprogramm wurde die Diagnosestrategie für die Fehlerdiagnose von Digital-Analog- und Analog-Digital-Wandlern in einer Loop-Back-Struktur, Abbildung VII-15 umgesetzt.

Bei der Partitionierung werden der R2R-DAC und der Flash-ADC als Komponenten behandelt. Beide Wandler können auf Netzwerkebene simuliert werden. Zunächst wurde der Loop-Back-Test für 3-Bit-Versionen des Flash-ADCs, des R2R-DACs sowie des Stromquellen-DACs realisiert. Die Kennlinien wurden auf Verhaltensebene mit MATLAB erzeugt. Als Fehler wurden sowohl strukturelle wie Haft-, Brücken-, Unterbrechungs- und Parameterfehler als auch funktionelle Fehler wie fehlende Codes untersucht. Insgesamt wurden 98 Fehler des Flash-ADCs, 86 Fehler des R2R-DACs und 32 Fehler des Stromquellen-DACs betrachtet.

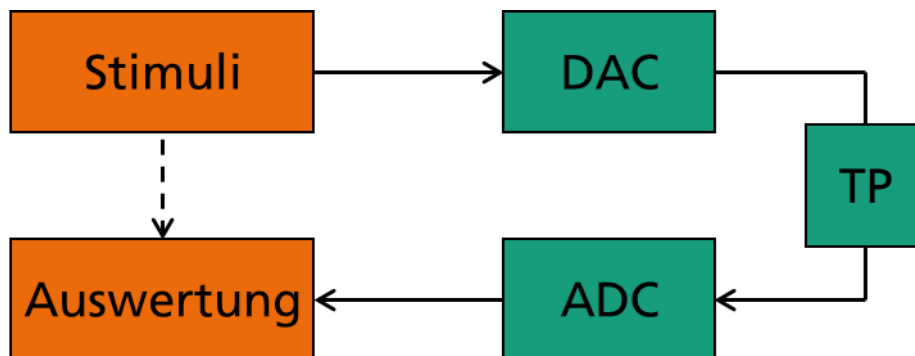


Abbildung VII-15: Loop-Back-Struktur aus DAC und ADC

Im nächsten Schritt wurden auf Netzwerkebene 5-Bit-Versionen des Flash-ADCs und des R2R-DACs simuliert (s. Aufgabe 1.1). Die Simulationen stehen in engem Zusammenhang mit dem Continental-Demonstrator, da auf dem Demonstrator-Board die beiden Wandler in Hardware realisiert sind und als Loop-Back-Struktur geschaltet werden können. Insgesamt wurden 9.197 Fehler im Flash-ADC und 430 Fehler im R2R-DAC untersucht. Die Simulationsergebnisse liegen sowohl als Ergebnisdatei als auch als Bild der Kennlinie vor.

Simulation und Fehlersimulation auf Netzwerkebene erfolgen so, dass Kennlinien der Wandler daraus abgeleitet werden können. Der ADC wird mit einer Rampenfunktion und der DAC mit einer Treppenfunktion stimuliert. Die zeitlichen Parameter wie



Anstieg der Rampenfunktion und Taktung der Treppenfunktion korrespondieren dabei mit den Vorgaben der Loop-Back-Struktur.

Bei der weitgehend automatischen Konvertierung der Simulationsergebnisse werden beim ADC die Eingangswerte zu den Flanken bestimmt, beim DAC die Ausgabewerte zu den Eingangswerten. Aus allen Ergebnisdateien wird unter Einbeziehung der Fehlerliste eine Datei erzeugt, in der alle Kennlinien mit der jeweiligen Fehlerinformation stehen. Da die Kennlinien von ADC und DAC auch analoge Werte haben, ist die Genauigkeit von Bedeutung. Nach der Konvertierung der Kennlinien wurde untersucht, welche Kennlinien zueinander äquivalent sind, da äquivalente Kennlinien auch äquivalente Fehler bedeuten. Im Ergebnis dieses Kennlinienvergleichs brauchen nur 2.286 Fehler (25%) des ADCs und 185 Fehler (47%) des DACs weiter betrachtet werden.

Die Simulation der Gesamtschaltung wird auf Verhaltensebene mit MATLAB in der Loop-Back-Struktur ausgeführt. Für die Loop-Back-Struktur wird ein synchroner Takt verwendet, mit dem der Stimulus-Generator bzw. das Eingangsregister des DACs und das Ausgangsregister des ADCs arbeiten.

Zur Charakterisierung von Flash-ADC und R2R-DAC sind statische Kenngrößen wie Verstärkungsfehler (engl. gain error), Offsetfehler, INL (engl. integrated non-linearity), DNL (engl. differential non-linearity) oder TUE (engl. total unadjusted error) üblich [1]. Entsprechend erfolgt die Stimulierung des DACs mit einer Treppenfunktion. Der ADC kann nur durch den Ausgang des DACs stimuliert werden. Deshalb wird zwischen beide Wandler ein Tiefpass geschaltet, der die Treppenfunktion am Ausgang des DACs glättet.

Die Ausgabe des ADCs der Loop-Back-Struktur und damit auch jedes Simulationsergebnis ist zeit- und wertediskret. Zur Reduzierung der Datenmenge werden Kenngrößen verwendet, mit denen die Simulationsergebnisse charakterisiert und Unterscheidungen insbesondere hinsichtlich von Fehlern möglich werden. Zunächst werden die o. g. Kenngrößen zur Charakterisierung von ADCs und DACs auch für die Fehlerdiagnose genutzt, später aber weitere Kenngrößen verwendet.

Die bei der Loop-Back-Simulation berechneten Kenngrößen werden als Merkmalsvektor zu einer Fehlersignatur zusammengefasst und in einem Fehlerverzeichnis gespeichert. Für die sich anschließende Fehlerdiagnose wird nur auf dieses Fehlerverzeichnis zurückgegriffen.

### ***Einbeziehen von Parameterschwankungen in den Diagnoseprozess***

Parameter, z. B. Kennwerte von Widerständen, Transistoren, Bauelementen, Zellen, bestimmen die Kennlinien der untersuchten Wandler. Schwankungen dieser Parameter können im Herstellungsprozess auftreten. Das Einbeziehen von Parameterschwankungen in den Diagnoseprozess erfolgt mit unterschiedlichen Zielstellungen:

- a. Fehlerdiagnose unter Berücksichtigung der Parameterschwankungen
- b. Parameterschwankungen selbst sind Ziel der Diagnose. Es steht die Frage, inwieweit bestimmte Parameter schwanken.

Im Folgenden wird vor allem auf die erste Fragestellung Bezug genommen. Durch Einbeziehen von Parameterschwankungen in den Diagnoseprozess wird die Fehlerdiagnose besser quantifiziert.

Für die Untersuchungen von Parameterschwankungen wurden an einem 5-Bit-Flash-ADC die Widerstände des Spannungsteilers und die Schwellwerte der Komparatoren, an einem 5-Bit-R2R-DAC die Widerstände des R2R-Netzwerks sowie Offset und Verstärkung des Ausgangsverstärkers variiert. Als Nominalwert wurden der Idealwert und als Standardabweichung 1% verwendet. Für den fehlerfreien Fall und jeden Fehler wurde zusätzlich zur variationsfreien Simulation eine Monte-Carlo-Simulation mit 10.000 pseudozufällig variierten Parametern durchgeführt. Im Einzelnen wurden folgende Fehler untersucht:

- 5-Bit-Flash-ADC: 256 Widerstands-, 310 Schwellwert-, 72 Haftfehler
- 5-Bit-R2R-DAC: 100 Widerstands-, 40 Offset-, 24 Verstärkungsfehler

Die Loop-Back-Struktur wurde mit diesen Kennlinienscharen simuliert. Aus den Ergebnissen jeder Monte-Carlo-Simulation wurden Fehlersignaturen berechnet. Jede Signatur besteht aus mehreren Kenngrößen, für die jeweils ein Nominalwert und ein Histogramm berechnet werden. Diese Fehlersignaturen werden abgespeichert und für die Fehlerdiagnose verwendet.

### ***Fehlerdiagnose***

Bei der Fehlerdiagnose besteht die Aufgabe, von einem unbekanntem Fehler Fehlerort, Fehlerart und Fehlerwert zu bestimmen. Die Fehlerdiagnose kann basierend auf einem Abstandsmaß oder basierend auf Abstandsquantifizierung durch Parametervariationen erfolgen.

Anhand von Abbildung VII-16 wird die zweite Herangehensweise für zwei Kenngrößen erläutert. Die Kenngrößen der Monte-Carlo-Simulationen liegen um den Nominalwert jeden Fehlers innerhalb eines Rechtecks, hier als Fehlerintervall bezeichnet. Liegt eine gemessene und zu untersuchende Fehlersignatur innerhalb eines solchen Fehlerintervalls, so ist der dazugehörige Fehler Kandidat für die gemessene Fehlersignatur, vgl. [6], [7].

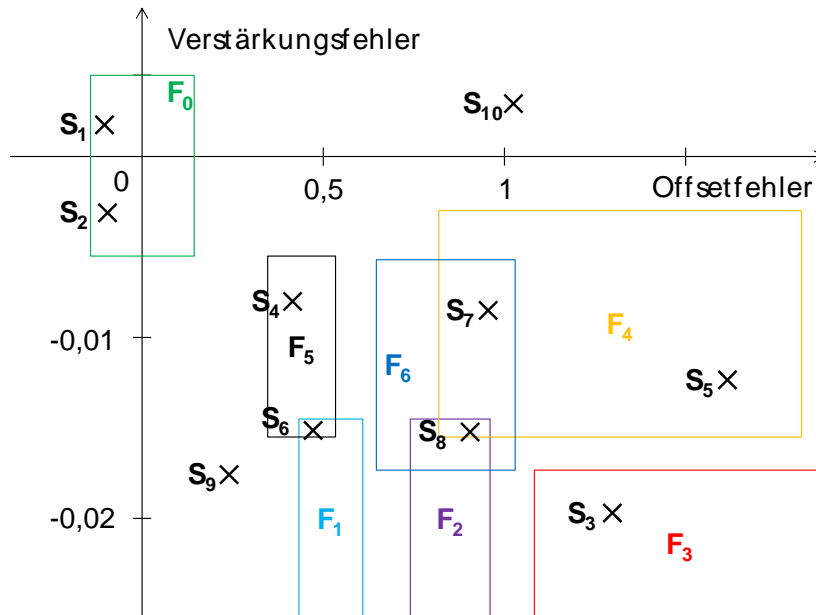


Abbildung VII-16: Fehlerintervalle  $F_i$  der Kenngrößen „Verstärkungsfehler“ und „Offsetfehler“ für verschiedene Fehler und gemessene Fehlersignaturen  $S_i$

Für den Wirksamkeitsnachweis wurden per Zufallsgenerator 10.000 Fehler ausgewählt und die Fehlerdiagnose durchgeführt. Bei ca. einem Viertel der Fehler war eine eindeutige Fehlerlokalisierung möglich. Bei den anderen Fehlern wurden mehrere Fehlerorte ermittelt. Bei über 22% der Fehler sind über 40 Fehlerorte möglich.

### Fehleranalyse

Mit Hilfe der Fehleranalyse kann eine Schaltung bezüglich der betrachteten Fehler, der verwendeten Kenngrößen der Fehlersignatur und der Stimuli dahingehend beurteilt werden, wie gut die Fehlerdiagnose ausfällt. Abhängig von der Herangehensweise bei der Fehlerdiagnose bezieht sich auch die Fehleranalyse auf die Abstandsmaße oder die Abstandsquantifizierung durch Parametervariationen. Die im Folgenden beschriebene Fehleranalyse nimmt auf die Abstandsmaße Bezug.

Bei der Fehleranalyse werden die Abstände aller Fehler miteinander verglichen.

- Ein Fehler ist gut von allen anderen zu unterscheiden und damit gut diagnostizierbar, wenn der Abstand zum nächsten Fehler möglichst groß ist.

Wird die Abstandsquantifizierung durch Parametervariationen realisiert, so werden die Histogramme der Kenngrößen in die Beurteilung einbezogen. Anhand der Fehlerintervalle in Abbildung VII-16 kann die Güte der Fehlerdiagnose beurteilt werden, d. h., wie gut die Fehler lokalisierbar sind und welchen Beitrag die einzelnen Kenngrößen dabei leisten. Der Einfachheit halber werden im Folgenden Fehler an unterschiedlichen Fehlerorten betrachtet.

- Ein Fehler ist dann eindeutig lokalisierbar, wenn er sich von jedem anderen Fehler in mindestens einer Kenngröße unterscheidet.

In Abbildung VII-16 ist außer dem fehlerfreien Fall  $F_0$  nur der Fehler  $F_3$  eindeutig lokalisierbar. Von den Fehlern  $F_1$  und  $F_2$  unterscheidet sich  $F_3$  im Offsetfehler und von  $F_4$  im Verstärkungsfehler, von den anderen Fehlern in beiden Kenngrößen. Von den anderen Fehlern haben wenigstens zwei (z. B.  $F_1$  und  $F_5$ ) eine gemeinsame Schnittmenge.

Um die Fehlerlokalisierbarkeit zu verbessern, müssen weitere Kenngrößen hinzugenommen werden. Dabei wird untersucht, wie viel jede Kenngröße zur Unterscheidung beiträgt. Aus der Kenntnis der Kennlinienverläufe bisher nicht unterscheidbarer Fehler können gezielt zusätzliche, auch alternative Kenngrößen (vgl. [3], [10]) entwickelt werden, die zu deren Unterscheidung beitragen.

### VII.2.1.3. Umsetzung der Diagnosestrategie für einen Delta-Sigma-Analog-Digital-Wandler erster Ordnung

Die Diagnosestrategie wurde an einem weiteren Anwendungsbeispiel, dem Test eines  $\Delta\Sigma$ -ADCs erster Ordnung [4] demonstriert. Simulationen von  $\Delta\Sigma$ -Wandlern auf Netzwerkebene erfordern einen sehr hohen Simulationsaufwand. Schon Wandler erster Ordnung besitzen vier Operationsverstärker, dazu kommt noch ein getaktetes D-Flip-Flop als digitales Bauelement. Dabei sind Übergänge von der analogen zur digitalen Domäne und durch die Rückkopplung zurück in die analoge notwendig. Diese Übergänge erschweren eine geschlossene Simulation der Gesamtschaltung. Zudem sind viele Takte notwendig, bis die Wandlung ausgeführt ist.

Im Gegensatz zur Loop-Back-Schaltung aus DAC und ADC, wo die Partitionierung vorgegeben ist, die Stimuli festliegen und die Einbindung der Simulationsergebnisse in Form von Kernlinien erfolgt, liegt beim Beispiel  $\Delta\Sigma$ -ADC das Vorgehen bei der hierarchischen Diagnosestrategie nicht so auf der Hand.

Im Rahmen dieses Projektes wurden zunächst Untersuchungen an  $\Delta\Sigma$ -ADCs 1. und 2. Ordnung auf Verhaltensebene durchgeführt. Im MATLAB-Modell des  $\Delta\Sigma$ -ADCs wurden für die Blöcke Funktionen verwendet, die den gängigen Funktionen beschalteter Operationsverstärker entsprechen. Für die Fehlersimulation wurden Verstärkungs-, Offset- und Schwellenfehler an den Operationsverstärkern untersucht und gezeigt, dass diese Fehler diagnostiziert werden können.

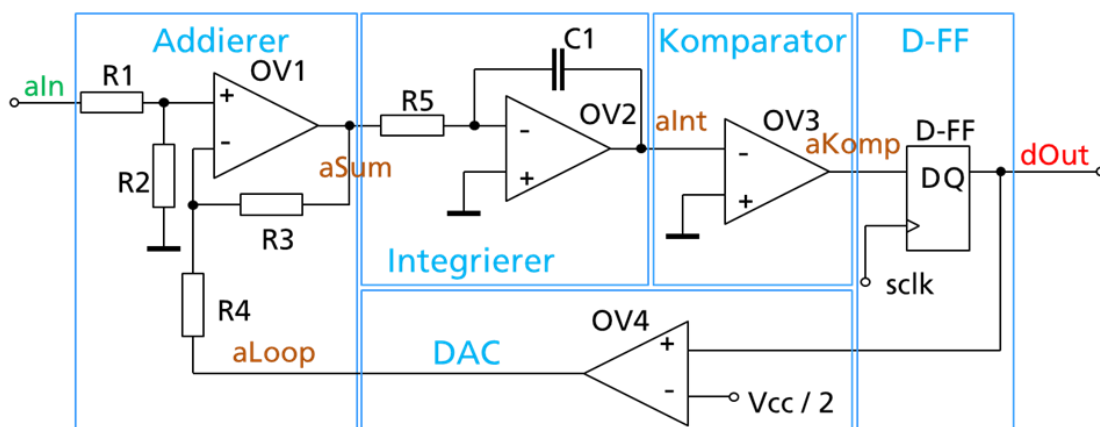


Abbildung VII-17: Partitionierung des  $\Delta\Sigma$ -ADCs

Abbildung VII-17 zeigt die Partitionierung für das hierarchische Vorgehen. Dabei entstehen Komponenten mit Funktionen, die im Verhaltensmodell verwendet werden. Nach der Partitionierung können auf Netzwerkebene die Gutsimulation und die Fehlersimulationen für jede Komponente ausgeführt werden. Aus den Simulationsergebnissen werden dann die Parameter für die Funktionen der Komponenten berechnet. Die Funktionen selbst werden durch Funktionsapproximation (engl. fitting) gewonnen.

Bei der Simulation des  $\Delta\Sigma$ -ADCs werden die Komponenten wiederholt durchlaufen. Da das Ausgangssignal einer Komponente das Eingangssignal der nächsten bildet, besteht die Gefahr, dass sich kleine Abweichungen summieren und zu falschen Ergebnissen führen. Die Gegenkopplung beim  $\Delta\Sigma$ -ADC wirkt dem entgegen. Bei der Verhaltenssimulation werden weitgehend die gleichen Eingangswerte der Komponenten verwendet wie bei der Netzwerksimulation. Zwischenwerte werden durch Interpolation gewonnen. Zusätzlich wird eine maximale Abweichung berechnet, um die Einhaltung der Genauigkeit zu beurteilen.

Durch LTSpice-Simulationen der fehlerfreien Gesamtschaltung wurden die Funktionen der Komponenten und sinnvolle Stimuli für die elektrische Netzwerksimulation festgelegt. Da der Addierer zwei Eingänge besitzt, müssen die Stimuli weitgehend die auftretenden Eingangssignalkombinationen abdecken, vgl. Abbildung VII-18. Die Stimulierung des Integrierers ist insofern kompliziert, weil er eine interne Rückkopplung besitzt. Abbildung VII-18 zeigt den gefundenen Stimulus. Da 1-Bit-Digital-Analog-Wandler und Komparator jeweils nur einen Eingang besitzen, ist deren Stimulierung vergleichsweise einfach. Für den 1-Bit-DAC werden Pulse unterschiedlicher Dauer, wie sie am Ausgang des D-FFs entstehen und für den Komparator eine ansteigende und abfallende Rampe verwendet.

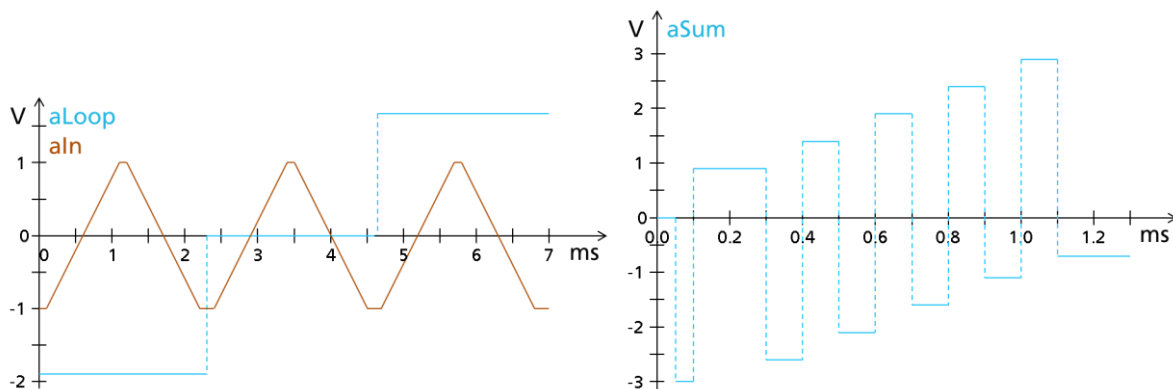


Abbildung VII-18: Stimuli für Addierer (links) und Integrierer (rechts)

Alle Komponenten haben einen Operationsverstärker, der unterschiedlich beschaltet ist. Jede der Komponenten wurde fehlerfrei und mit 528 Fehlern simuliert. Bei der Konvertierung konnten für alle Komponenten Funktionsparameter berechnet werden. Beim anschließenden Vergleich der Funktionsparameter wurden äquivalente Funktionsparameter-Tupel der jeweiligen Komponente entfernt, so dass beim Addierer 514, beim Integrierer 517, beim 1-Bit-DAC 527 und beim Komparator 514 unterschiedliche Funktionen zu berücksichtigen sind.

Mit den parametrisierten Funktionen wird die Gesamtsimulation des  $\Delta\Sigma$ -ADCs auf der Verhaltensebene durchgeführt. Nach Konvertierung der Ergebnisse der Fehlersimulationen auf der elektrischen Netzwerkebene stehen die Funktionsparameter für jede Komponente in einer Datei zur Verfügung. Im MATLAB-Modell des  $\Delta\Sigma$ -ADCs wurden die entsprechenden Funktionen implementiert. Bei der Gesamtsimulation des  $\Delta\Sigma$ -ADCs auf der Verhaltensebene werden diese Dateien eingelesen und jede Komponente korrespondierend mit dem aktuell bearbeiteten Fehler parametrisiert. Insgesamt werden so 2.072 verschiedene Fehler simuliert. Nach der Fehlersimulation wird ein Vergleich der Fehlersignaturen durchgeführt, um Fehler zu finden, die in Bezug auf die Simulationsergebnisse äquivalent sind. Die berechneten Fehlersignaturen werden in der Ergebnisdatei abgespeichert und stehen für die Fehlerdiagnose zur Verfügung.

#### VII.2.1.4. Untersuchungen an einem $\Delta\Sigma$ -Analog-Digital-Wandler zweiter Ordnung

Der untersuchte  $\Delta\Sigma$ -Analog-Digital-Wandler 2. Ordnung (Messkanal) mit Auflösung von 12 Bit und Dezimierung durch ein Gazsi-Butterworth-Filter findet sich weitgehend auf einem repräsentativen Design wieder, vgl. Abbildung VII-19. Zur Stimulierung wird ein Sinussignal verwendet. Die Untersuchung der dynamischen Kenngrößen SNR und THD [9] erfolgt für verschiedene Amplituden.

Als Fehler werden Abweichungen der internen Verstärkungen, der Offsets und der Schwellen betrachtet. Dabei wird der Wertebereich der einzelnen Parameter zunächst logarithmisch abgedeckt. Zur Erhöhung der Genauigkeit ändern sich die Werte für den Bereich um den Gutwert dagegen linear.

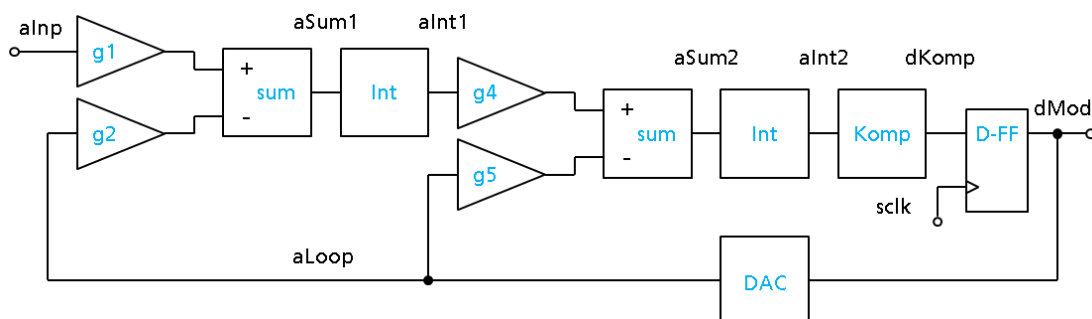


Abbildung VII-19: MATLAB-Modell des  $\Delta\Sigma$ -ADCs 2. Ordnung

Mit diesen Fehlern wurden die Simulationen in der MATLAB-Umgebung ausgeführt. Dabei wurde der  $\Delta\Sigma$ -ADC 2. Ordnung in die die Loop-Back-Struktur eingebunden und hier die Fehler injiziert. Als Stimulus wurde ein Sinus verwendet, dessen Amplitude von -20 dB bis 0 dB variiert wurde. Die Ergebnisse zeigen, dass beim  $\Delta\Sigma$ -ADC 2. Ordnung ähnlich wie beim  $\Delta\Sigma$ -ADC 1. Ordnung Offsetfehler verschiedener OPVs einerseits und Verstärkungsfehler verschiedener OPVs andererseits in gewissen Bereichen gleiche Wirkung zeigen. Damit ist keine eindeutige Zuordnung des Fehlers zu einem bestimmten OPV möglich. Wohl aber können Offset- und Verstärkungsfehler voneinander unterschieden werden.

### **VII.2.1.5. Programme zur Simulation und Fehlerdiagnose von Digital-Analog- und Analog-Digital-Wandlern in einer Loop-Back-Struktur**

Im Rahmen des Projektes wurde ein MATLAB-Programmpaket zur Simulation des Loop-Back-Tests und zur Fehlerdiagnose entwickelt. Neben statischen wurden dynamische Tests einbezogen. Bei den statischen Tests werden Fehler und Parameterschwankungen durch Kennlinien von DAC und ADC berücksichtigt. Bei diesem hierarchischen Vorgehen können diese Kennlinien entweder als Ergebnis von Aufgabe 1.1 durch elektrische Netzwerksimulationen oder durch ein separates MATLAB-Programm auf Verhaltensebene erzeugt werden. Für den dynamischen Test von  $\Delta\Sigma$ -ADCs ist ein komponentenbezogenes hierarchisches Vorgehen umgesetzt worden.

Die im Rahmen des Projektes entstandenen Programme bilden keine geschlossene Software. Vielmehr werden die anstehenden Aufgaben Schritt für Schritt gelöst und so eine Gesamtlösung erreicht. Die im Folgenden aufgeführten Programme bauen aufeinander auf und sind über die Ergebnisdateien miteinander vernetzt.

- **Konvertieren der Simulationsergebnisse der elektrischen Netzwerkebene in die Verhaltensebene**

*ConvertFaultList* konvertiert die Fehlerliste, die mit dem analogen Fehlersimulator aFSIM [8] automatisch erzeugt wird, in eine Tabelle mit Fehlernummer, Fehlerort, –art, und –wert als Referenz für die Fehlerdiagnose.

*ConvertaFSIMFiles* konvertiert die Ergebnisdateien. Diese enthalten den Zeitpunkt und die Simulationsergebnisse im fehlerfreien Fall und für den konkreten Fehler. Beim ADC werden die Zeitpunkte der Flanken berechnet und in die Eingangswerte umgerechnet. Beim DAC werden die Ausgangswerte nach jeder Eingangswertänderung bestimmt. Durch einen Vergleich der konvertierten Kennlinien können äquivalente Kennlinien eliminiert und so Simulationen erspart werden.

- **Erzeugen und Anzeigen von ADC-Kennlinien**

*CreateWriteADCchar* erzeugt ADC-Kennlinien auf Verhaltensebene und legt diese in einer Datei ab. Die Fehler sind auf dieser Ebene beschrieben:

- Decoder-Haftfehler
- Komparator-Haftfehler
- parametrische Fehler der Widerstandskette
- missing Codes

*ShowADCChar* zeigt **ausgewählte** Kennlinien als MATLAB-Plot an.

*CreateWriteADCMC* generiert eine Schar von ADC-Kennlinien mit Monte-Carlo-Simulation.

- **Erzeugen und Anzeigen von DAC-Kennlinien**

*CreateWriteDACchar* erzeugt DAC-Kennlinien auf Verhaltensebene und legt diese in einer Datei ab. Die Fehler auf Verhaltensebene sind:

- parametrische Fehler der Stromquellen

- parametrische Fehler der Widerstände des R2R-Netzwerks
- Offsetfehler des Ausgangsverstärkers
- Verstärkungsfehler des Ausgangsverstärkers
- Einzelfehler (Veränderung der Kennlinie an einer Stelle)
- zufällige Veränderungen der Kennlinie

**ShowDACChar** zeigt ausgewählte Kennlinien als MATLAB-Plot an.

- Erzeugen und Anzeigen von Kennlinien für  $\Delta\Sigma$ -ADCs

**CreateWriteADCcharDS** erzeugt Kennlinien für einen  $\Delta\Sigma$ -ADC auf Verhaltensebene und legt diese in einer Datei ab. Fehler sind dabei:

- parametrische Fehler der Widerstände
- parametrische Fehler des Kondensators
- Offsetfehler
- Schwellwertfehler
- Verstärkungsfehler

**ShowADCdsChar** zeigt ausgewählte Kennlinien als MATLAB-Plot an

- Loop-Back-Test

*TestLoopBack* führt den Loop-Back-Test für gegebene Kennlinien von DAC und ADC auf Verhaltensebene aus. Die entstehenden Dateien mit Fehlersignaturen sind Grundlage der Fehlerdiagnose und Fehleranalyse.

Kenngößen sind Verstärkungsfehler, Offsetfehler, TUE, fehlende Codes, minimale und maximale Werte von DNL und INL.

- Loop-Back-Test für ADC-Monte-Carlo-Simulationen

*GenADCMCHist* erzeugt Fehlersignaturen mit Histogrammen für ADCs, *GenDACMCHist* Fehlersignaturen mit Histogrammen für DACs.

- Fehlerdiagnose

*FaultDiagnosis* führt die Fehlerdiagnose aus.

*TestFaultDiagnosis* testet *FaultDiagnosis* mit Zufallswerten und erstellt eine Statistik. Das Programm beinhaltet folgende Schritte:

- Definieren der Histogramm-Datei-Namen und Öffnen dieser Dateien
- Einlesen der Histogramm-Dateien (Monte-Carlo-Simulation)
- Bestimmen einer zufälligen Fehlersignatur für den zu untersuchenden Fehler
- Ausführen der Fehlerdiagnose
- Schreiben der Ergebnisse in Ergebnisdatei
- Bilden einer Statistik

- Fehleranalyse

*ClassifyFaults* klassifiziert Fehlersignaturen als Ergebnisse der Loop-Back-Simulation ohne Parametervariationen mit Hilfe eines Abstandsmaßes.

**TestClassifyFaults** stellt für *ClassifyFaults* die Aufrufparameter bereit und ermöglicht so einen Test des Programms.



- **Hierarchischer Test des  $\Delta\Sigma$ -ADCs 1. Ordnung**

*ConvertLTSpiceFiles* konvertiert die Ergebnisse der LTSpice-Simulation in Ergebnisse zum Taktwechsel.

*ConvCurve2FctClk* konvertiert die Ergebnisse der LTSpice-Simulation in Funktionen für Komponenten des  $\Delta\Sigma$ -ADCs.

*ConvertCurve2Function* konvertiert die Ergebnisse der aFSIM-Simulation der Komponenten des  $\Delta\Sigma$ -ADCs in **Parameter der Komponentenfunktionen**.

*TestDSADC1st* simuliert den  $\Delta\Sigma$ -ADC **1. Ordnung auf Verhaltensebene unter Verwendung der Funktionsparameter**.

#### **VII.2.1.6. Demonstrator für die Fehlerdiagnose**

Die Erfüllung der Aufgabenstellungen von 1.1 und 1.2 wird mit einem Demonstrator-Board DA/AD-Wandler nachgewiesen. Er enthält einen 5-Bit-Flash-ADC, einen 5-Bit-R2R-DAC und einen  $\Delta\Sigma$ -ADC **1. Ordnung**. Die Wandler können einzeln oder als Loop-Back-Struktur konfiguriert werden. Die Fehlerinjektion wird mit sog. Jumpfern realisiert. Das sind Brücken, die entweder einen Kurzschluss darstellen oder mit Widerständen bestückt werden. Dabei werden Widerstände oder Dioden überbrückt oder Anschlüsse auf ein festes Potential gelegt. Dadurch können parametrische Widerstandsfehler, Schwellwertfehler, aber auch Haftfehler nachgebildet werden. Zur Anzeige der Fehler auf dem Board werden LEDs vorgesehen. Die in Abbildung VII-20 dargestellten Blöcke entsprechen im Wesentlichen den Schaltungen von ADC und DAC, an denen die Fehlerdiagnosestrategie in den Abschnitten VII.2.1.2 und VII.2.1.3 beschrieben wurde.

Die vorgesehenen Wandler wurden entsprechend der Konzeption so in Hardware implementiert, dass eine sinnvolle Fehlerinjektion möglich wurde:

- Widerstände des R2R-Netzwerkes des DACs: diskret
- Widerstände der Widerstandskette des Flash-ADCs: diskret
- Komparatoren des Flash-ADCs: Schaltkreise
- Operationsverstärker des DS-ADCs: Schaltkreise
- Operationsverstärker des DACs: Schaltkreise
- Beschaltung der Operationsverstärker: diskret

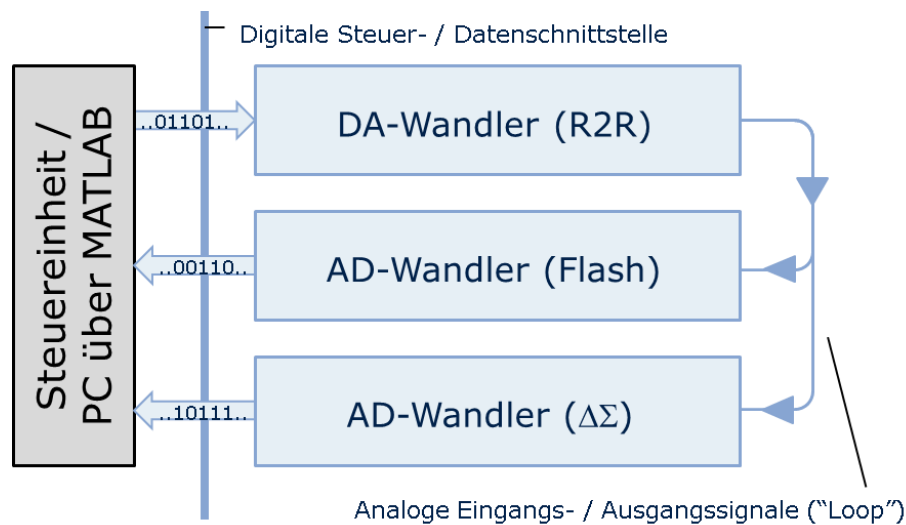


Abbildung VII-20: Schematische Darstellung des Demonstrator-Boards

Als Schnittstelle PC – Demonstrator-Board wurde eine programmierbare Schnittstelle mit dem USB-IO24-Kabel [2] realisiert. Zur Bedienung der Schnittstelle und damit zum Steuern des Demonstrators gibt es Lese- und Schreibbefehle. Die gewählte Schnittstelle hat den Nachteil, dass die Ausführungszeit der Lese- und Schreibbefehle stark variiert und von Bedingungen abhängt, die sich dem Einfluss des PCs entziehen. Zur Stimulierung wurde die vorhandene Breite des Datenports genutzt und der DAC auf 8 Bit erweitert. Durch die geringe Datenrate und den ungleichmäßigen Datenstrom konnte der Tiefpass nicht verwendet werden. Deshalb wirken sich selbst kleine Störungen ungünstig aus. Unter solchen Bedingungen war eine besonders robuste Fehlerdiagnose notwendig. Die gewählte Fehlerdiagnose orientiert sich an Kenngrößen, die die Kennlinien vor allem qualitativ erfassen. Folgende Kenngrößen wurden ausgewählt:

- Breite und Kode der längsten Stufe der Kennlinie
- Breite und Kode der kürzesten Stufe der Kennlinie
- Unterschiede der Breiten der verbliebenen Stufen
- Zahl der fehlenden Codes und fehlende Codes
- Zahl der Richtungsänderungen, erste Richtungsänderung

Es wurden mehrere Programme zur Fehlerdiagnose am Demonstrator-Board implementiert. Nach Setzen der Jumper wird mit *SimBoardCharCurves* das Board stimuliert, die Kennlinie aufgezeichnet und gespeichert. *DemoFaultDiagnosis* löst die Aufgabe der Fehlerdiagnose durch Aufruf verschiedener Programme:

- *ReadCharCurvesCreateFaultSigs* liest Kennlinien ein und berechnet daraus Fehlersignaturen.
- *StimReadDemoBoardCompFsig* stimuliert das Demo-Board, liest die Kennlinie des zu untersuchenden Fehlers ein und berechnet dessen Fehlersignatur.
- *DiagnoseFault* diagnostiziert Fehler durch Vergleich der Fehlersignaturen des zu untersuchenden Fehlers mit den simulierten Fehlersignaturen.
- *ShowFault* zeigt diagnostizierten Fehler durch eine LED an.

Die gewählten Kenngrößen der Fehlersignaturen besitzen in Verbindung mit der Prioritätslogik am Flash-ADC ein hohes Unterscheidungspotenzial. Das zeigte sich an den Ergebnisse der Fehlerdiagnose: Von den 40 Fehlern wurden 18 korrekt und 10 mehrheitlich diagnostiziert, bei 8 der korrekte Fehlerort, bei zwei Fehlern der korrekte Fehler mit 50% bzw. 42%. Nur bei zwei Fehlern konnte nicht festgestellt werden, welcher Fehler vorliegt.

### **VII.2.2. Zusammenfassung**

In Aufgabe 1.2 werden Diagnoseprogramme bezüglich der Fehlerlokalisierung und des Einflusses von Veränderungen im Herstellungsprozess bewertet. Als repräsentatives Beispiel wurde das Verfahren des Tests von Digital-Analog-Wandlern (DAC) und Analog-Digital-Wandlern (ADC) in einer Loop-Back-Struktur ausgewählt. Die betrachteten DACs arbeiten parallel mit R2R-Netzwerk oder mit geschalteten Stromquellen. Neben Flash-ADCs werden zusätzlich Delta-Sigma-ADCs erster und zweiter Ordnung in die Untersuchungen einbezogen.

Ein Schwerpunkt dieses Meilensteinberichtes ist die hierarchische Diagnosestrategie. Sie besteht darin, dass die Fehler auf der elektrischen Netzwerkebene definiert und Fehlersimulationen auf dieser Ebene mit der dabei nötigen Genauigkeit ausgeführt werden. Die Simulationsergebnisse werden auf die Verhaltensebene transformiert, ohne dass dabei Genauigkeitsverluste auftreten und der Bezug zu den Fehlern verloren geht. Die sich anschließenden Untersuchungen auf Verhaltensebene ermöglichen umfangreiche Simulationen und Berechnungen und münden schließlich in eine Fehlerdiagnose, deren Ergebnis auf die Fehler der elektrischen Netzwerkebene abgebildet werden kann.

Diese Diagnosestrategie wurde auf eine Loop-Back-Struktur bestehend aus R2R-DAC und Flash-ADC sowie einen Delta-Sigma-ADC umgesetzt. In der ersten Anwendung wurden die kompletten Wandler in der elektrischen Netzwerkebene bearbeitet, in der zweiten Anwendung Komponenten wie Addierer, Integrierer, Komparator und 1-Bit-DAC.

Im Rahmen des Projektes wurde ein MATLAB-Programmpaket zur Simulation des Loop-Back-Tests und zur Fehlerdiagnose entwickelt. Das Programm bezieht ADCs und DACs über ihre Kennlinien ein. Diese Kennlinien sind Ergebnis von Fehlersimulationen, die entweder in der MATLAB-Umgebung oder auf der elektrischen Netzwerkebene ausgeführt werden. Bei Wahl einer Rampe als Stimulus werden statische Kenngrößen ermittelt und daraus eine Fehlersignatur gebildet, die den jeweiligen injizierten Fehler charakterisiert. Basierend auf den Fehlersignaturen wird ein Abstandsmaß definiert. Dieses ist wiederum Grundlage einer Analyse, in dessen Ergebnis die Diagnoseauflösung für den DAC und den ADC der Loop-Back-Struktur ermittelt wird. Eine erweiterte, ebenfalls realisierte Herangehensweise bezieht Parametervariationen, die im Herstellungsprozess entstehen, in die Fehlerdiagnose ein. Mittels Monte-Carlo-Simulation entsteht für jeden Fehler eine Kennlinienschar. Nach dem Loop-Back-Test wird schließlich für jede zu einem Fehler gehörende Kenngröße ein Histogramm gebildet. Die für jeden Fehler berechnete Fehlersignatur enthält damit neben den Kenngrößen selbst Histogramme derselben. Bei der Fehlerdiagnose wird die Signatur eines zu diagnostizierenden Fehlers mit den im Voraus berechneten und in einem Fehlerverzeichnis abgespeicherten

Fehlersignaturen verglichen. Die Histogramme ermöglichen dabei eine Abgrenzung verschiedener Fehler voneinander.

Das Demonstrator-Board DA/AD-Wandler besteht aus einem 8-Bit-R2R-DAC, einem 5-Bit-Flash-ADC und einem  $\Delta\Sigma$ -ADC erster Ordnung. Die einzelnen Blöcke werden als Beispiele für die hierarchische Diagnosestrategie im Bericht beschrieben. Alle Blöcke wurden auf elektrischer Netzwerkebene und auf der Verhaltensebene simuliert. In der Hardware sind Fehlerinjektionen möglich. Diese Fehler werden anschließend diagnostiziert.

### VII.2.3. Ausblick

Mögliche nachfolgende Aktivitäten nach Beendigung der direkten Projektarbeiten innerhalb der Verwertungsphase sollten die gemeinsame Integration der beiden Bestandteile des hierarchischen Diagnosezugangs<sup>4</sup> in Form eines einzigen Diagnosewerkzeuges sein. Dadurch werden Durchgängigkeit und automatisierter Ablauf des Diagnosevorgehens erreicht, z. B. innerhalb eines gängigen EDA-Tools.

Damit sind u. a. erzielbar:

- Beschleunigung der Analyse und Diagnosearbeiten
- Erhöhung der Ablaufstabilität durch Herabsetzen der Anfälligkeit möglicher Fehler durch Bediener infolge automatisierter Abläufe
- Visualisierungsmöglichkeiten von diagnostizierten Fehlern bezüglich ihrer Auftrittorte in Schematic bzw. Layout
- direkter Anschluss zu anderen Features des EDA-Tools
- Unterstützung von Protokollierung und Dokumentation
- Entlastung von möglichen Schnittstellenproblemen

Indem man die zu untersuchenden Schaltungen als Verhaltensmodelle in MATLAB-Notation formuliert, wird auch ein integrierter Zugang zu modellbasiertem Test und modellbasierter Vorgehensweise bei der Diagnose ermöglicht.

Weiterhin kann der hierarchische Fehlerdiagnosezugang auf die in Aufgabe 4.3<sup>5</sup> des Arbeitspaketes 4<sup>6</sup> verwendete Schaltung inklusive der dort angewendeten Diagnosestruktur erweitert werden. Mit deren Bearbeitern gab es bereits innerhalb von Fachdiskussionen einen intensiven Gedankenaustausch. Damit kann der Anschluss an ein vergleichsweise einfaches Prüfinstrumentarium und an Messprozeduren erreicht werden, mit denen aus Sicht des Gesamtsystems trotzdem eine hohe Diagnoseschärfe und somit weitgehend eindeutige Fehlerlokalisierungen ermöglicht werden.

---

<sup>4</sup> Modellierungen und Fehlersimulationen von Komponenten auf elektrischer Netzwerkebene und auf Verhaltensebene verbunden mit der Fehlerdiagnose in einer Loop-Back-Struktur

<sup>5</sup> Methoden und Algorithmen zur optimalen Diagnoseauflösung für analoge und gemischt digital-analoge Baugruppen

<sup>6</sup> Erfassen und Verarbeiten von Diagnosedaten im Gesamtsystem

#### VII.2.4. Literatur

- [1] „ADC and DAC Glossary“, MAXIM Application notes, Jul 22, 2012. <http://www.maximintegrated.com/appnotes/index.mvp/id/641>
- [2] Altenburg, U.: „USB-IO24-Kabel. Einfach messen, steuern und regeln mit dem PC“, [www.elektor.de/zeitschrift](http://www.elektor.de/zeitschrift), Dezember 2012
- [3] Chakrabarti, S.; Cherubal, S.; Chatterjee, A.: “Fault Diagnosis for Mixed-Signal Electronic Systems”, Aerospace Conference, 1999, pp. 169 – 179
- [4] Desouky, A.O.; Khairy, M.S; Abdelhalim, M.B.; Amer, H.H.: “Low-cost test of the first-order Delta-Sigma converter”, Computer Engineering & Systems 2009, pp. 100-105
- [5] Duhamel, P.; Rault, J.: “Automatic Test Generation Techniques for Analog Circuits and Systems: A review”, IEEE Trans. on Circuits and Systems, vol. CAS-26, no. 7, pp. 411-440, July 1979
- [6] Gulbins, M., Vermeiren, V., Redlich, St.: „Fehlerdiagnose für AD- und DA-Wandler in einer Loop-Back-Struktur“, Dresdner Arbeitstagung Schaltungs- und Systementwurf (DASS 2012), Dresden, 3./4. Mai 2012
- [7] Gulbins, M., Vermeiren, V., Redlich, St.: „Fehlerdiagnose für ADCs und DACs in einer Loop-Back-Struktur unter Einbeziehung von Parametervariationen“, Testmethoden und Zuverlässigkeit von Schaltungen und Systemen 25. GI/GMM/ITG-Workshop Dresden, 24. - 26. Februar 2013
- [8] Hopsch, F.; Lindig, M.; Straube, B.; Vermeiren, W.: “Characterization of digital cells for statistical test,” Proc. IEEE 14th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS), April 2011, pp. 255 -260.
- [9] Kester, W., “Understand SINAD, ENOB, SNR, THD, THD + N, and SFDR so you don't get lost in the noise floor”, Analog Devices, Tutorial MT-003, October 2008
- [10] Kook, S. H.: “Low-Cost Testing of High-Precision Analog-to-Digital Converters”, PhD Thesis. School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, August 2011

### VII.3. Ergebnisse zur Aufgabe 1.3 „Fehlerdiagnose für Analog- und Mixed-Signal-Schaltungen“

Die Testsignale, die beim Test analoger Schaltungen verwendet werden, sind sehr häufig aus der Spezifikation abgeleitet. Zum Zweck der Fehlerdiagnose ist es notwendig, einen aufgetretenen Fehler zunächst einer bestimmten Schaltung und dann einer bestimmten Struktur auf dem Wafer zuzuordnen. Testsignale, die aus einer Funktionsbeschreibung oder einer Spezifikation abgeleitet werden, geben keinen spezifischen Hinweis auf den Ort, an dem der Fehler auftritt.

Es ist deshalb notwendig, Testsignale so auszuwählen, dass eine Zuordnung des Fehlerbildes zu einem Schaltungsteil möglich wird. Für die rechen-technische Bearbeitung von Fehlern in analogen Schaltungen steht ein analoger Fehlersimulator zu Verfügung. Daraus ergeben sich Anforderungen an die Schaltungsbeschreibung, an die Modellierung und an die Arbeitsweise, sowohl im Design als auch bei der Simulation. Zur Vorbereitung analytischer Verfahren wurden in dieser Aufgabe Untersuchungen an speziellen Schaltungen vorgenommen, bei denen ein Fehler zu diagnostizieren war. Gegenstand der Bearbeitung waren ein Operationsverstärker und ein Tri-Port-RAM, dessen Schaltung nur mit Werkzeugen für Analogschaltungen einer Fehlerdiagnose unterzogen werden kann. Die Schaltungsgröße des Tri-Port-RAM erfordert insbesondere die Entwicklung einer Arbeitsweise bei der Modellierung der Schaltung und bei der Durchführung der Simulation. Das Ziel bestand darin, am Beispiel von typischen eingebetteten Komponenten eine Methodik zur Fehlerdiagnose für Mixed-Signal-Schaltungen zu erarbeiten.

Ein weiterer methodischer Schwerpunkt lag darin, Analyseergebnisse, die bei der analogen Fehlersimulation entstehen, für weitere Test- und Diagnosemethoden von Mixed-Signal-Schaltungen bereitzustellen, die Robustheit von Mixed-Signal-Schaltungen, und die Leistungsfähigkeit von ATPG-Tools für digitale Schaltungen zu verbessern.

Die Nutzerfreundlichkeit des analogen Fehlersimulators wurde durch die Einbindung in das Analog Environment von Cadence Virtuoso IC6 verbessert.

#### VII.3.1. Untersuchungen zur Fehlererkennbarkeit am Tri-Port-RAM

Folgende Arbeiten und Untersuchungen wurden durchgeführt:

- Generieren des Designs eines für die Untersuchungen zugeschnittenen RAM verringerter Komplexität
- Generieren von Netzlisten, die es gestatten, den Simulationsaufwand möglichst gering zu halten
- Erstellen der Testbench für Simulationen
- Ausführung von Fehlersimulationen zur Analyse der Erkennbarkeit von Fehlern im Tri-Port-RAM
  - Analyse der Erkennbarkeit von in die Speicherzellen injizierte resistive Kurzschluss- und Unterbrechungsfehler
  - Analyse der Erkennbarkeit eines konkreten Fehlers unter Variationen von Versorgungsspannungen und der Temperatur

### VII.3.1.1. Generieren des Designs eines für die Untersuchungen zugeschnittenen RAM verringerter Komplexität

Das Originaldesign des RAM entspricht einem 32x17 Bitarray und der dazugehörigen Steuerlogik. Die Wortbreite des RAM beträgt 17 Bit und es können 32 Wörter im RAM gespeichert werden. Die Adressbreite beträgt 5 Bit. Durch den regulären Aufbau des RAM ist die Bearbeitung eines Ausschnittes des Bitarray für aussagekräftige Ergebnisse hinreichend. Das vorliegende Design erlaubte die Konzentration der Untersuchungen auf eine 4-Bit-Zelle. Zu diesem Zweck wurde eine 4-Bit-Zelle und die dazugehörige Steuerlogik aus dem Design des RAM herausgelöst. Die Speicherzellen, die auf die durchzuführenden Untersuchungen keinen Einfluss haben, wurden als Dummy-Zellen modelliert. Auf diese Weise konnte die Komplexität der zu untersuchenden Schaltung drastisch reduziert werden, so dass sowohl die Simulationszeit als auch die anfallenden Datenmengen akzeptabel waren.

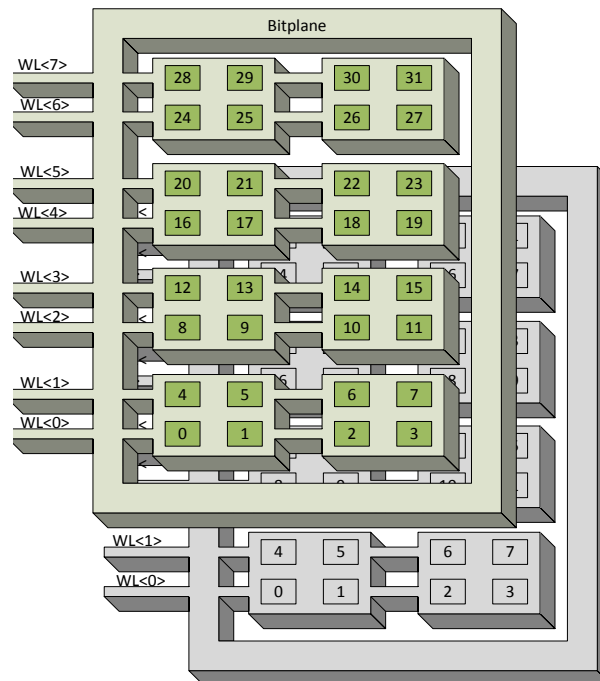


Abbildung VII-21: Schematische Darstellung des Bitarray

### VII.3.1.2. Erstellen der Testbench für Simulationen

Eine Halbierung der Simulationszeit wurde erreicht, indem auch die Steuerlogik des RAM auf die Signalpfade, die zur Ansteuerung der herausgelösten 4-Bit-Zelle benötigt werden, reduziert wurde.

Eine weitere Beschleunigung der Simulation wurde dadurch erreicht, dass die RAM-Simulation einmalig mit den gegebenen Stimuli ausgeführt wird, um die Signale am Control-Interface aufzuzeichnen. Die Simulation der Bit-Plane wird dann unter Verwendung der aufgezeichneten Signale ausgeführt (Abbildung VII-22). Hierbei ist

es wichtig, dass die Aufzeichnung der Signale mit der jeweils verwendeten Netzliste erfolgt, d.h. eine Wiederverwendung der aufgezeichneten Signale von der Design-Netzliste bei der Simulation der rückerkannten Netzliste ist nicht möglich. Diese weitere Reduktion ist durch die Tatsache möglich, dass jede Bit-Plane und jede 4-Bit-Zelle ihre eigene Steuerlogik beinhaltet, welche die nötigen Treiberstärken für Lesen und Schreiben erzeugt.

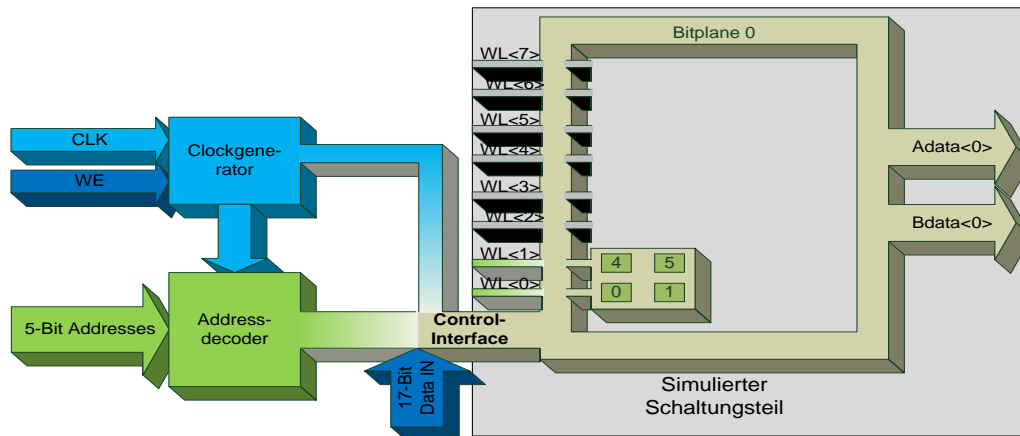


Abbildung VII-22: Reduzierte Testbench einer Bit-Plane

Die typische Simulationszeit des analogen Fehlersimulators mit dieser Testbench betrug etwa 10 Minuten.

### VII.3.1.3. Fehlersimulationen und Fehlerlisten

Ziel der auszuführenden analogen Fehlersimulationen war es, die Fehlererkennung zu bewerten, die bei Einsatz verschiedener Testalgorithmen erzielt wird. Darüber hinaus sollten Bedingungen untersucht werden, die die Erkennbarkeit von Fehlern beeinflussen. In den vier Speicherzellen der 4-Bit-Zelle wurden resistive Unterbrechungsfehler mit Widerstandswerten von  $10\text{ M}\Omega$  an den in der nachstehenden Abbildung VII-23 gekennzeichneten Orten injiziert. Resistive Kurzschlussfehler mit Widerstandswerten von  $10\ \Omega$  wurden in den vier Speicherzellen der 4-Bit-Zelle an jedem Transistor zwischen den jeweiligen Anschlüssen Gate-Drain, Gate-Source, Source-Drain und Drain-Bulk injiziert. Äquivalente Fehler werden dabei nur einmal injiziert.



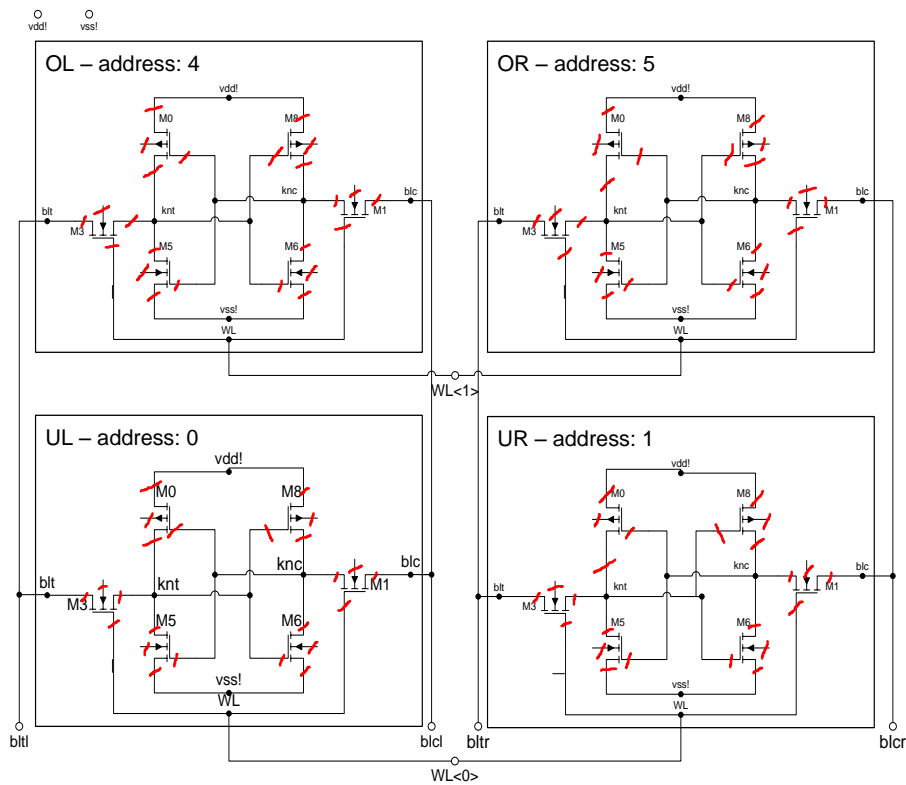


Abbildung VII-23: Schematische Darstellung für injizierte resistive Unterbrechungen mit Widerstandswerten von  $10\text{ M}\Omega$

#### VII.3.1.4. Verwendete Testfolgen

Für die Untersuchungen wurden verschiedene Marching-Tests in Signalfolgen für die Testbench umgesetzt. Der March-C+1W-Test wurde direkt aus dem Testszenario für den RAM übernommen, ein mehrfaches Lesen der Bitzellen wurde im March-Y+-Test realisiert. Eine Umsetzung des Checkerboard-Tests wurde als zusätzlicher Test für Untersuchungen bezüglich möglicher Auswirkungen von benachbarten Bit-Zellen implementiert. Bei der Umsetzung der Testfolgen aus den Marching-Tests wurden diese auf die vier Adressen 0, 1, 4 und 5 reduziert und absteigende bzw. aufsteigende Adressfolgen wurden mittels Gray-Code umgesetzt, d.h. bei einer Adressänderung wird immer nur ein Signal geändert. Somit ergibt sich eine aufsteigende Adressreihenfolge von 0, 1, 5, 4 und absteigend 4, 5, 1, 0. Die Gray-Code Umsetzung reduziert die auftretenden Simulationsevents und somit auch die Komplexität der transienten Simulation. Der Umstand, dass der RAM ein Dual Port RAM ist, wurde dahingehend ausgenutzt, dass Port A immer aufsteigend und Port B immer absteigend zum lesen adressiert wird.

### ***VII.3.1.5. Analyse der Erkennbarkeit von resistiven Kurzschluss- und Unterbrechungsfehlern in den Speicherzellen der 4-Bit-Zelle***

Von den 104 injizierten resistiven Unterbrechungsfehlern (Open) wurden 55 Fehler detektiert, von den 50 injizierten resistiven Kurzschlussfehlern (Short) wurden 48 Fehler detektiert. Nicht detektierte Open-Fehler sind 12 Fehler pro Bit-Zelle. Außerdem ist der FanInOpen an der Versorgungsspannung vdd vor seiner Auffächerung zu den Source-Terminals der pmos-Transistoren aller 4 Zellen nicht detektiert worden. In den Bit-Zellen OL und UR wurden 2 Kurzschlussfehler nicht detektiert.

Die Ursache für die Nichterkennbarkeit der Fehler besteht in der robusten Schaltung der Speicherzelle, die eine Fehlertoleranz bewirkt.

### ***VII.3.1.6. Analyse der Erkennbarkeit eines für einen Fehler kausalen Defektes unter veränderten Betriebsbedingungen***

Die Beschreibung des Fehlers [1] beinhaltete die gewonnenen Erkenntnisse zu einem Defekt an einem Kontakt einer Bit-Select-Read-Leitung, welcher nur bei niedrigen Temperaturen auftrat und nur dann anhand seiner elektrischen Auswirkung detektierbar war. Es standen weiterhin Defektbilder aus dem Layout und mikroskopische Aufnahmen des defekten Kontaktes sowie Beschreibungen über die Auswirkungen des Defektes zur Verfügung. Ziel der durchzuführenden Untersuchungen war es, aufzuklären, inwieweit der Fehler bei Raumtemperatur getestet werden kann. Dazu wurde die Erkennbarkeit des Fehlers unter Variationen von Versorgungsspannung und Temperatur mit Hilfe der analogen Fehlersimulation analysiert.

Der zu untersuchende Defekt auf dem Wafer wurde als resistive Unterbrechung modelliert und der Netzliste der Zelle UL der 4-Bit-Zelle am Select-Transistor M1 injiziert (Abbildung VII-24).

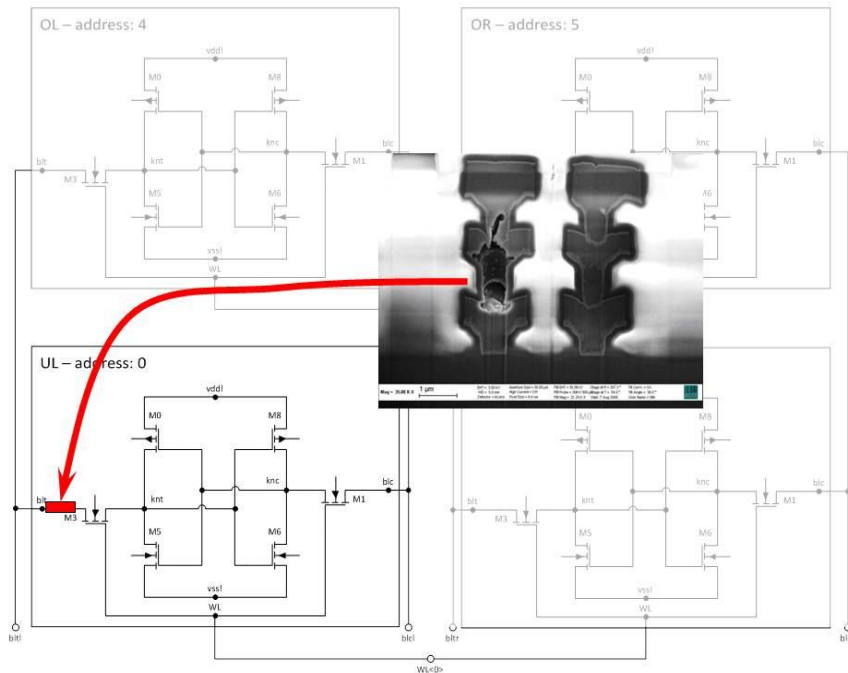


Abbildung VII-24: Injektion des zu untersuchenden und als resistive Unterbrechung modellierten Defektes in die Zelle UL einer Vier-Bit-Zelle

Für diese resistive Unterbrechung wurden analoge Fehlersimulationen mit verschiedenen Widerstandswerten und unterschiedlichen Temperaturen für Versorgungsspannungen von 5.5 V und 2 V durchgeführt. Abbildung VII-25 zeigt die Abhängigkeit der Erkennbarkeit von Widerstandswerten einer resistiven Unterbrechung bei diesen unterschiedlichen Betriebsbedingungen. Zu erkennen ist, dass bei einer Versorgungsspannung von 2 V nur resistive Unterbrechungen mit Widerstandswerten von etwa 30 k $\Omega$  detektiert werden können, wohingegen bei einer höheren Versorgungsspannung von 5.5 V eine Detektierbarkeit von bis zu einem Wert von ca. 10 k $\Omega$  erreicht wird. Das bedeutet, dass die Erkennbarkeit von Unterbrechungen bei Messungen mit höherer Betriebsspannung besser ist.

Die Untersuchungen zur Erkennbarkeit des Defektes führen zur Schlussfolgerung, dass ein resistiver Unterbrechungsfehler mit kleinstmöglichem Widerstandswert nur bei herabgesetzter Temperatur, jedoch nicht bei Raumtemperatur detektiert werden kann.

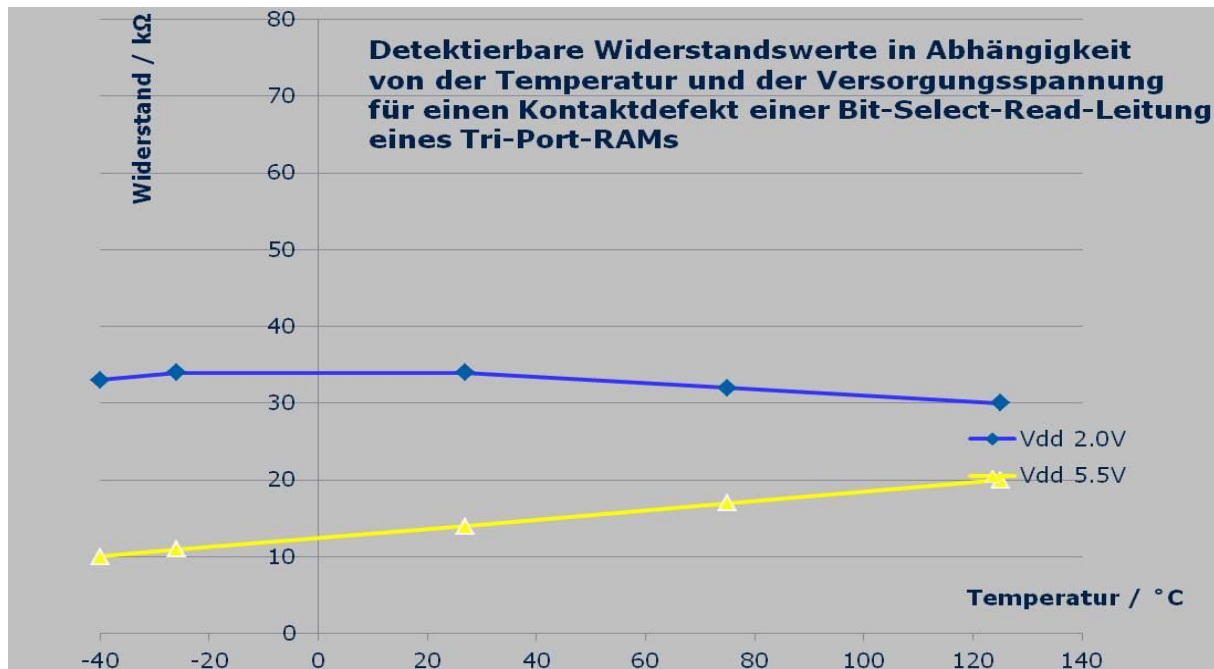


Abbildung VII-25: Abhängigkeit detektierbarer Widerstandswerte einer resistiven Unterbrechung bei unterschiedlichen Betriebsbedingungen

### **VII.3.1.7. Konfigurieren einer für Diagnoseuntersuchungen geeigneten Schaltungsbeschreibung**

Die Beschreibung der Schaltung für eine Diagnose muss alle parasitären Elemente, Vias und Kontakte der zu untersuchenden Komponente enthalten. In der Umgebung dieser Komponente reichen für ihre Ansteuerung und die Weiterleitung der Ausgangssignale mit LVS überprüfte Schematic-Netzlisten aus. Es entsteht eine hierarchische Schaltungsbeschreibung, die es ermöglicht, die Korrespondenz zwischen den in den rückerkannten Netzlisten enthaltenen parasitären Bauelementen und den Koordinaten im Layout herzustellen. Dadurch wird die Lokalisierung eines Defektes auf dem Wafer unterstützt werden.

### **VII.3.1.8. Vorschläge für Maßnahmen zur Verbesserung der Testbarkeit von Fehlern im RAM**

Eine Möglichkeit zur Verbesserung der Fehlererkennung besteht in der Variation von Testabläufen wie die Variation von Taktzyklen und die Erhöhung der Anzahl aufeinanderfolgender Lesezyklen.

Der Stress einer Schaltung führt generell zur Erhöhung der Anzahl der erkannten Fehler.

Durch Maßnahmen des Design for Test kann in Testmodi die Robustheit und damit die Empfindlichkeit gegenüber verschiedenen Fehlern von Schaltungsstrukturen beeinflusst werden. Dabei ist aber immer ein Kompromiss zwischen der erforderlichen Fläche und dem erreichbaren Qualitätsgewinn erforderlich.

### VII.3.2. Untersuchungen zur Fehlererkennbarkeit an einem Operationsverstärker

Für einen ausgewählten Operationsverstärker wurden Arbeitspunktanalysen, Analysen im Frequenzbereich, Transientenanalysen, Stabilitätsanalysen und Transfer-Funktions-Analysen ausgeführt.

Dieser Verstärker ist ein voll differentieller Verstärker mit Class-AB-Ausgangsstufe und hochohmigem Eingang. Die Verstärkungs-Rückkopplung ist bereits integriert. Abbildung VII-26 zeigt einen vergleichbaren Operationsverstärker.

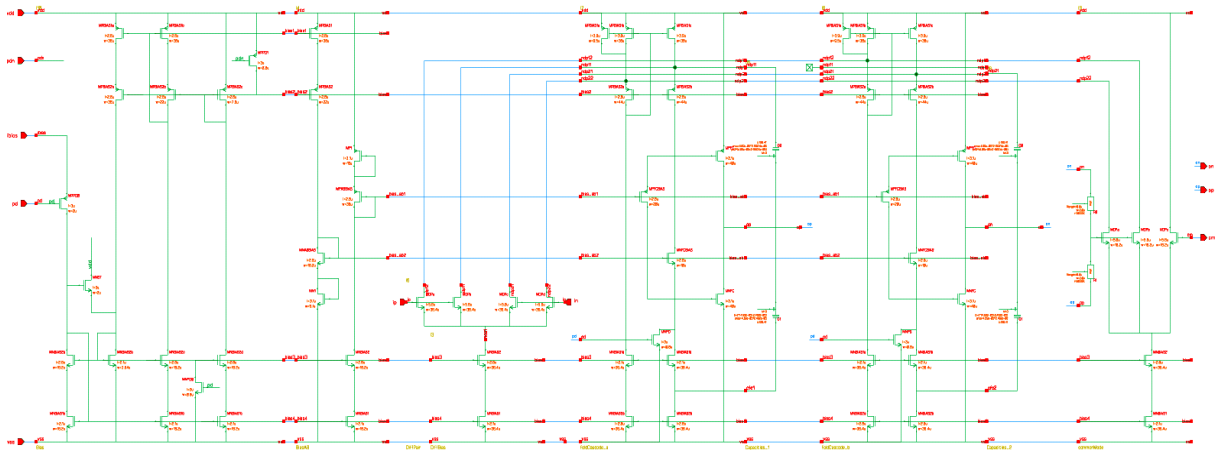


Abbildung VII-26: Operationsverstärker

Die aus dem Layout rückerkannte Spectre-Netzliste besteht aus 3419 Zeilen und enthält 3164 parasitäre Widerstände und Kapazitäten.

#### VII.3.2.1. Bestimmung der zu untersuchenden analogen Defekte

Die Untersuchungen wurden damit begonnen, Open- und Short-Fehler direkt in die extrahierten Netzlisten durch Modifizieren der Ursprungsnetzliste zu injizieren und den Operationsverstärker mit diesen Fehlern zu simulieren.

Mit diesem Zugang ergab sich eine hohe Anzahl nicht erkannter Fehler. Es war zu klären, ob es sich um Fehler handelt, die aufgrund nicht ausreichender Teststimuli nicht detektiert werden konnten, oder die aus strukturellen Gründen grundsätzlich nicht detektierbar sind.

Fehler, die wegen der Schaltungsstruktur nicht erkennbar sind, sollen in eine Klasse nicht erkennbarer Fehler eingeordnet werden und von der Fehlersimulation ausgeschlossen werden.

Ursachen für nicht erkannte Fehler am Verstärker:

- Widerstände in Serie zu einer Stromquelle können prinzipiell nur dann erkannt werden, wenn der Spannungsabfall am Widerstand so groß ist, dass der

Compliance-Bereich der Stromquelle verlassen wird oder der Widerstand den Frequenzgang der Stromquelle ausreichend stark beeinträchtigt.

- Biasing an Kaskode
- Eine Gate-Drain-Verbindung ist vorhanden, wodurch ein Kurzschluss nicht erkannt werden kann.
- Einige Fehler sind mittels Transientenanalyse nicht oder schwer zu erkennen.

Strukturen, bei denen Fehler schwer erkennbar sind:

- Dummy-Devices
- Stromspiegel (hier ist ein Fehler nur bei dynamischer Belastung erkennbar),
- Gate-Drain-Verbindung
- Transistoren, die als Diode geschaltet sind (zwei Transistoren mit Drain-Gate-Verbindung, die in Reihe geschaltet sind und für eine Bias Spannungserzeugung sorgen)
- Substratkontakt von Transistoren, Wanne, Widerstandsschleife

In Auswertung der Ergebnisse wurde ein graphentheoretischer Algorithmus entwickelt werden, mit dem möglichst realitätsnahe Defektpositionen als potentielle Fehlerorte ermittelt werden können. Weiterhin wurde ein Werkzeug zur automatischen Erkennung von strukturell nicht erkennbaren Fehlern entwickelt. Damit werden Fehler, die bedingt durch die Schaltungsstruktur, nicht erkennbar sind, von der Bearbeitung mit dem analogen Fehlersimulator ausgeschlossen.

### ***VII.3.2.2. Ausgeführte Untersuchungen am Verstärker als repräsentatives analoges Schaltungsdesign***

Die Untersuchungen sind an einer aus dem Layout extrahierten Netzliste ausgeführt worden. Für Fehlersimulationsläufe wurden insgesamt sind 2020 Fehler injiziert:

- resistive Unterbrechungen an 418 parasitären Widerständen (je 2 Widerstandswerte von 10 k $\Omega$  und 100 k $\Omega$ )
- resistive Kurzschlüsse an 308 parasitären Kapazitäten (je 3 Widerstandswerte von 10  $\Omega$ , 500  $\Omega$  und 1000  $\Omega$ )
- resistive Kurzschlüsse an 50 Transistoren (Widerstandswerte von 10  $\Omega$ )
- in einer zweiten Iteration für noch nicht erkannte Fehler
  - resistive Kurzschlüsse 0.1  $\Omega$
  - resistive Unterbrechungen bis zu 1000  $\Omega$

Als Auswertungskriterium zur Fehlerdetektion ist jeweils eine  $\pm 10\%$ -Abweichung von den Nominalwerten der einzelnen Kennwerte verwendet worden. Mit den verwendeten Widerstandswerten ist die Fehlererkennungsrate von Kurzschlussfehlern höher als die Fehlererkennungsrate von Unterbrechungsfehlern. Die Erkennbarkeit der Fehler wird maßgeblich durch die Widerstandswerte der resistiven Fehler beeinflusst. Die Auswertung der einzelnen Kennwerte der Schaltung zeigt den unterschiedlichen Beitrag zur Fehlererkennung. Durch die Auswertung einzelner Kennwerte wird eine sehr große Zahl von Fehlern erkannt. Es gibt auch Kennwerte, durch deren Auswertung nur eine kleine Anzahl von Fehlern erkannt wird. In Auswertung der Zusammenhänge zwischen Analysearten, den verwendeten Inputstimuli, überprüften Kennwerten sowie Parametern der injizierten

Fehler kann eine Optimierung der Testabläufe hinsichtlich der Ziele Optimierung der Testaufwandes oder der möglichst genauen Vorhersage eines Fehlerortes erfolgen. Mit Hilfe der Fehlersimulationen konnte dokumentiert werden, dass alle injizierten Fehler einen konkreten Bezug zu Fehlerorten im Layout hatten und so bei einer nachgelagerten Diagnose aus vorgegebenen Fehlerbildern eine Defektlokalisierung erfolgen kann.

In einer zweiten Iteration wurde die Fehlersimulation für die bisher nicht erkannten Fehler wiederholt, um nicht erkennbare Fehler zu identifizieren. Dabei wurden die Widerstandswerte für resistive Kurzschlüsse bis zu  $0.1 \Omega$  verringert und die Widerstandswerte für resistive Unterbrechungen bis zu  $100 \text{ M}\Omega$  erhöht.

#### ***VII.3.2.3. Nicht entdeckte Kurzschluss-Fehler***

An den parasitären Kapazitäten erwiesen sich 3 Kurzschlussfehler mit Widerstandswerten von  $0.1 \Omega$  als nicht detektiert, wenn als Auswertungskriterium zur Fehlerdetektion jeweils eine  $\pm 10\%$ -Abweichung von den Nominalwerten der einzelnen Kennwerte verwendet wird. Einer dieser Kurzschlussfehler an einer parasitären Kapazität ist auch dann nicht detektierbar, wenn als Auswertungskriterium zur Fehlerdetektion eine  $\pm 0\%$ -Abweichung von den Nominalwerten der einzelnen Kennwerte verwendet wird.

Zwischen Transistoranschlüssen erwiesen sich 12 Kurzschlussfehler mit Widerstandswerten von  $0.1 \Omega$  als nicht detektiert, wenn als Auswertungskriterium zur Fehlerdetektion jeweils eine  $\pm 10\%$ -Abweichung von den Nominalwerten der einzelnen Kennwerte verwendet wird. Zwei dieser Kurzschlussfehler zwischen Transistoranschlüssen sind auch dann nicht detektierbar, wenn als Auswertungskriterium zur Fehlerdetektion eine  $\pm 0\%$ -Abweichung von den Nominalwerten der einzelnen Kennwerte verwendet wird.

Charakteristisch für nicht erkennbare Fehler war die Lage der Kurzschlüsse zwischen Knoten mit annähernd gleichen Potentialen.

#### ***VII.3.2.4. Nicht entdeckte Unterbrechungsfehler***

An den parasitären Widerständen erwiesen sich 40 Fehler mit Widerstandswerten von  $100 \text{ M}\Omega$  als nicht detektiert, wenn als Auswertungskriterium zur Fehlerdetektion jeweils eine  $\pm 10\%$ -Abweichung von den Nominalwerten der einzelnen Kennwerte verwendet wird.

Zwei dieser Fehler sind auch dann nicht detektierbar, wenn als Auswertungskriterium zur Fehlerdetektion eine  $\pm 0\%$ -Abweichung von den Nominalwerten der einzelnen Kennwerte verwendet wird.

Charakteristisch für nicht erkennbare Fehler war die Lage der Unterbrechungen in einem Zweig, der parallel zu einem anderen Widerstandszweig liegt. Abbildung VII-27 zeigt ein Beispiel für einen nicht erkennbaren Unterbrechungsfehler bei parallel geschalteten Widerständen.

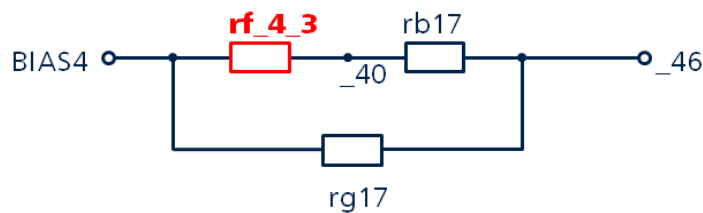


Abbildung VII-27: Beispiel für einen nicht detektierbaren resistiven Unterbrechungsfehler

### VII.3.3. Layout-Untersuchungen

Zur Vervollständigung der Beschreibung der Schaltungsstruktur sind einige Informationen aus dem Layout erforderlich. Wenn lange Parallelitäten zwischen zwei Signalleitungen vorliegen, besteht die Gefahr der gegenseitigen Beeinflussung, auch wenn kein Fehler vorhanden ist. Zur Ergänzung der Informationen zur ATPG für digitale Schaltungen ist die Information interessant, welche Signalleitungen, die keinen logischen Einfluss auf die jeweilige Standardzelle haben, im Layout über die Standardzelle geführt werden, wodurch ein Kurzschluss mit einer zellinternen Leitung nicht ausgeschlossen ist.

#### VII.3.3.1. Layout-Analyse bezüglich langer Leitungsparallelitäten

Zur Identifikation langer Leitungsparallelitäten wurden spezielle Features des analogen Fehlersimulators *aFSIM* 0 [2], [3] zur Extraktion der relevanten Koppelkapazitäten genutzt, die bis zur Visualisierung der interessanten Layoutbereiche im Layout reichen.

Durch entsprechende Verarbeitung von Informationen der Lage der aus einem Layout extrahierten Koppelkapazitäten und ihren Kapazitätswerten können Schwachstellen bezüglich kritischer Leitungsparallelitäten bestimmt werden.

Im Rahmen des entwickelten Softwarepaketes zur analogen Fehlersimulation *aFSIM* wird dazu eine geordnete Liste von Koppelkapazitäten erstellt. Auf deren Basis kann die entsprechende Visualisierung im Cadence Layout-Editor erfolgen. Zur Erzeugung einer geordneten Liste von Koppelkapazitäten unter Verwendung von Features des analogen Fehlersimulators ist eine Befehlsabfolge dokumentiert worden. Tabelle VII-3 zeigt einen Ausschnitt aus einer Liste geordneter Koppelkapazitäten.

Tabelle VII-3: Liste geordneter Koppelkapazitäten

OpAmp, coupling capacities					
ident	value	from node	to node	x-Koordinate	y-Koordinate
c507	3.515E-14	net0566	net0437	104.4125	57.211
c428	2.248E-14	net0530	net0558	131.182	87.5295



c409	1.829E-14	net0530	net0557	167.227	113.588
...	...	...	...	...	...
c233	1.068E-17	net0335	net0520	106.574	117.788

Abbildung VII-28 zeigt ein Beispiel für die Visualisierung einer langen Leitungsparallelität. Die Visualisierung erfolgt im Cadence Layout-Editor nach Auswahl einer Koppelkapazität in einem Pop-Up-Fenster.

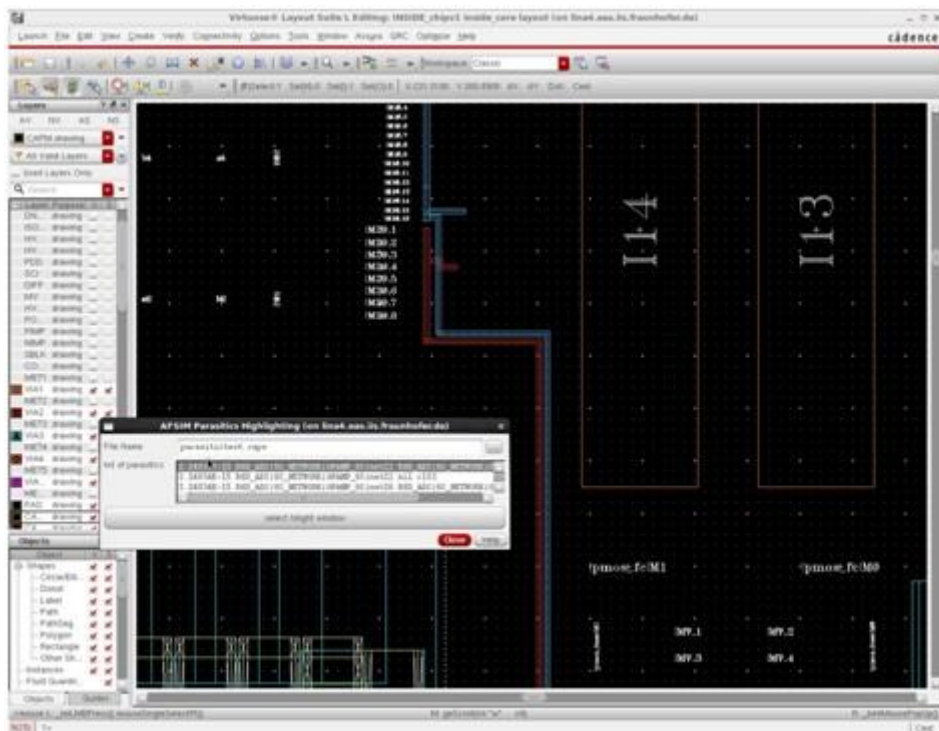


Abbildung VII-28: Visualisierung einer langen Leitungsparallelität (rote und blaue Leitung)

Aus der Ermittlung großer Koppelkapazitäten, die als Anhaltspunkt für lange parallele Leitungen dienen, lassen sich Hinweise zur Entwurfsverbesserung hinsichtlich der Robustheit von Schaltungen ableiten.

### ***VII.3.3.2. Identifikation potentieller Kurzschlüsse zwischen Leitungen, die in unterschiedlichen Beschreibungsebenen sind***

Durch die Ermittlung parasitärer Kapazitäten in extrahierten Netzlisten werden Referenzen zwischen Signalleitungen in digitalen Netzlisten mit Standardzellen und Leitungen in extrahierten digitalen Netzlisten der Standardzellen hergestellt.

Damit können für die Test-Pattern-Generierung mit kommerziellen ATPG-Tools neben den Standard-Bridging-Faults zusätzliche relevante Kurzschlussfehler berücksichtigt werden.

Basis dafür ist die Ermittlung aufeinanderfolgender geordneter Listen von Koppelkapazitäten gemäß unterschiedlicher Auswahlkriterien.

Ausgangspunkt ist die extrahierte Netzliste der zu analysierenden Schaltung. Durch Anwendung verschiedener Auswahlkriterien werden mögliche Kurzschlüsse in verschiedenen Kategorien ermittelt.

Für ein Beispiel mit einer Anzahl von 402 Knoten, bei dem sich die Anzahl theoretisch möglicher Kurzschlüsse zu 80601 ergibt, betrug die Gesamtzahl der parasitären Kapazitäten 2131. Von diesen parasitären Kapazitäten waren 204 zwischen Logic-Level-Leitungen und Leitungen innerhalb von Standardzellen.

Tabelle VII-4 zeigt das schrittweise Vorgehen bei der Ermittlung der verschiedenen Kategorien von Kurzschlussfehlern. Ein Befehlsablauf zur Durchführung dieser Prozedur ist dokumentiert worden.

Tabelle VII-4 Kategorien parasitärer Kapazitäten

Kategorie	Anzahl der noch zu bearbeitenden Kurzschlussfehler
theoretisch mögliche Kurzschlussfehler	80601
alle parasitären Kapazitäten	2131
961 parasitäre Kapazitäten zwischen den Anschlüssen der Transistoren	1170
37 parasitäre Kapazitäten zu Versorgungsspannungen	1133
901 parasitäre Kapazitäten, die nicht zwischen Logik-Level-Leitungen und Leitungen innerhalb von Standardzellen liegen	232
28 parasitäre Kapazitäten zwischen Ports und internen Leitungen von Standardzellen	204

Durch Ausgabe der Kurzschlussliste erfolgt im Format des ATPG-Tools, so dass eine Einbindung in einen erweiterten Flow für ATPG möglich ist.

Der Ausschluss von Koppelkapazitäten zwischen I/O-Leitungen von Standardzellen und Leitungen innerhalb von Standardzellen, ist noch nicht durch Software unterstützt.

#### **VII.3.4. Beschleunigung von Analysearbeiten durch Integration des Fehlersimulators aFSIM in die Cadence ADE**

Die Integration des Fehlersimulators *aFSIM* in das Cadence ADE dient durch Nutzung der Graphischen Oberfläche von Cadence der Erhöhung der Nutzerfreundlichkeit bei der Arbeit mit dem Fehlersimulator. Durch die Kopplung des analogen Fehlersimulators *aFSIM* mit Cadence-Features konnten Analysearbeiten

beschleunigt werden, insbesondere durch Visualisierung der nach einer Fehlersimulation als detektiert oder nicht detektiert ausgewiesenen Fehler in der Schaltung.

Der Aufruf des Fehlersimulators erfolgt aus dem Cadence Virtuoso ADE durch Auswahl im Menü „Tools“. Die Integration ist unter anderem durch folgende Merkmale gekennzeichnet:

- Graphische Oberfläche zur Steuerung der Signalauswertung
- Graphische Oberfläche für Optionen des Fehlersimulators
- Graphische Oberfläche für Statistiken zur Erkennbarkeit von Fehlern
- Graphische Oberfläche zur Steuerung von Fehlerinjektionen
- Möglichkeiten zur Visualisierung des Auftrittsortes von detektierten oder nicht-detektierten Fehlern im Schematic
- Möglichkeiten zur Visualisierung der Lage von potentiellen Schwachstellen im Layout, die sich durch hohe Werte von parasitären Koppelkapazitäten auszeichnen und auf lange parallele Leitungen hinweisen

In das Cadence Virtuoso ADE sind sowohl die Version des Fehlersimulators *afsim-digital*, zur Durchführung der analogen Fehlersimulation von auf elektrischer Netzwerkebene beschriebenen digitalen Schaltungen, als auch die Auslegung *afsim*, zur Durchführung der analogen Fehlersimulation von analogen Schaltungen, integriert worden.

Abbildung VII-29 zeigt exemplarisch anhand einer Fehlervisualisierung im Schematic

- die innerhalb der Cadence ADE eingebetteten graphischen Oberflächen des analogen Fehlersimulators *aFSIM*
  - zum Aufruf des Fehlersimulators durch Auswahl im ADE-Menü „Tools“,
  - für Fehlersimulatoroptionen zur Auswahl des zu untersuchenden Designs (DUT) und der zu verwendenden Widerstandswerte für resistive Unterbrechungen und Kurzschlüsse sowie
  - für die Auswahl der zu visualisierenden Fehler und
- den aufgeblendeten Schematic Editor mit den visualisierten Fehlerorten.

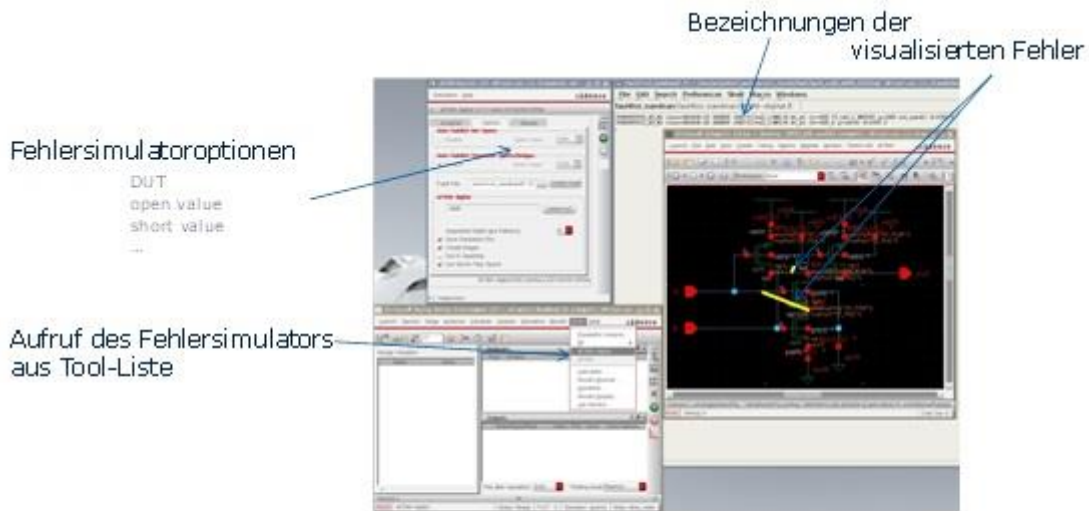


Abbildung VII-29: Illustration einer Visualisierung eines Fehlers im Schematic im Cadence ADE

### VII.3.5. Literatur

#### Zitierte Literatur

- [1] Fehlerbildbeschreibung eines Fehlerbildes an einem Tri-Port-RAM: Description of the fault at Tri-Port-RAM. Nichtöffentliche Information des ZMDI Dresden, 21.04.2011.
- [2] B. Straube, W. Vermeiren, B. Müller, C. Hoffmann, S. Sattler, "Analoge Fehlersimulation mit *aFSM*", Analog'99, 5. ITG/GMM-Diskussionssitzung 'Entwicklung von Analogschaltungen mit CAE-Methoden', München, 18. - 19. Februar 1999.
- [3] Hopsch, F.; Lindig, M.; Straube, B.; Vermeiren, W.: "Characterization of digital cells for statistical test," Proc. IEEE 14th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS), April 2011, pp. 255 -260.

#### Nicht zitierte Literatur

- [4] Goor, A.J. van de: Testing Semiconductor Memories: Theory and Practice. (reprinted with corrections April 1996). Chichester : Wiley, 1991. - ISBN 0-471-92586-1.
- [5] Bosio, A.; Dilillo, L.; Girard, P.; Pravossoudovitch, S.; Virazel, A.: Advanced Test Methods for SRAMs: Effective Solutions for Dynamic Fault Detection in Na-noscaled

Technologies. Boston, MA: Springer-Verlag US, 2010. - ISBN 978-1-4419-0937-4. - ISBN 978-1-4419-0938-1.

[6] Borri, S.; Hage-Hassan, M.; Dilillo, L.; Girard, P.; Pravossoudovitch, S.; Virazel, A.: Dynamic fault models for embedded-SRAMs. Analysis and Test. J Electron Test: Theory Appl. (JETTA), Springer, vol 21, 2005, pp. 169–178.

[7] Huang, R.-F.; Chou, Y.-F.; Wu, C.-W.: Defect Oriented Fault Analysis for SRAM. 12th Asian Test Symposium, Xian, China, 16-19 Nov. 2003, pp. 256-261.

[8] N. Engin, "Linking Mixed-Signal Design and Test: Generation and Evaluation of Specification-Based Tests", doctoral dissertation, Faculty of Electrical Eng., Univ. of Twente, Netherlands, 2000.

[9] Sunter, S.; Nagi, N., "Test metrics for analog parametric faults," 17th IEEE VLSI Test Symposium, 1999, pp.226-234.

[10] Kabisatpathy, P. ; Barua, A. ; Sinha, S., „Fault diagnosis of analog integrated circuits“, Dordrecht ; Springer ; 2005

[11] TITAN User’s Manual, Infineon Technologies AG, Sept. 2009

[12] Bushnell, M., Agrawal, V., “Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits”, Springer, ISBN : 978-0-7923-7991-1

[13] B. Vinnakota (ed.), “Analog and Mixed Signal Testing”, Upper Saddle River, New Jersey, Prentice Hall, 1998.

[14] Abderrahman, A.; Sawan, M.; Savaria, Y.; Khouas, A., "New Analog Test Metrics Based on Probabilistic and Deterministic Combination Approaches," 14th IEEE International Conference on Electronics, Circuits and Systems, 2007. ICECS 2007., pp.82-85, Dec. 2007

[15] International Test Conference 2010, October 31 – November 5, 2010, Austin, Texas, USA, Panel 2: Is A Scientific Method Necessary To Solve Analog, Mixed-Signal and RF Test Problems  
Moderator: F. Muradali, Organizer: G. Roberts  
Panelists: A. Chatterjee, R. Geiger, S. Ozev, G. Roberts, M. Slamani

[16] S. Sindia, V. D. Agrawal, V. Singh, “Parametric Fault Testing of Non-Linear Analog Circuits Based on Polynomial and V-Transform Coefficients”, J. Electronic Testing 28(5): 757-771 (2012)

#### VII.4. Ergebnisse zur Aufgabe 1.4, „Verbesserung der Testgüte für Digitalschaltungen auf Basis erweiterter Fehlermodelle“

Beim Test digitaler Schaltungen werden im Allgemeinen Testvektoren verwendet, die mit Werkzeugen zur automatischen Testpatterngenerierung (ATPG) erzeugt wurden. Dabei werden Testvektoren zur Erkennung von Stuck-at-0/1-Fehlern und Transition-Delay-Fehlern an den Ein- und Ausgängen der Standardzellen generiert. Durch den Einsatz dieser Testsätze ist jedoch, auch bei vollständiger Testüberdeckung, die Erkennung aller praktisch auftretenden erkennbaren Fehler nicht garantiert. Diese Tatsache führt zu der Aussage, dass die Modellierung von Defekten in der Logik-Bit-Ebene unzureichend ist. Um dem zu begegnen werden in der Literatur verschiedene Methoden vorgeschlagen.

Eine Methode ist der Multiple Detect Test von Stuck-at-0/1-Fehlern [4], wobei ein Stuck-at-0/1-Fehler durch mehrere verschiedene Testvektoren (bis hin zur vollständigen Funktionstabelle der Standardzelle) detektiert wird. Die Testvektoren entstehen dabei nicht zielgerichtet auf bestimmte Fehler hin. Wenn die Wertetabelle nicht vollständig abgearbeitet wird, ist die Erkennung aller erkennbaren Fehler nicht garantiert. Bei Einsatz der vollständigen Wertetabelle einzelner Standardzellen findet ein Funktionstest eines in eine Struktur eingebetteten Elementes statt. Funktionstest ist, auch wenn er nur auf eingebettete Elemente angewendet wird, aufwendiger als der Strukturtest.

Eine andere Vorgehensweise ist die defektorientierte Testgenerierung. Dabei werden technologiespezifische Defekte berücksichtigt und Layoutinformationen verwertet. Im Beitrag [5] wird die Testpatterngenerierung nach Betrachtung technologiespezifischer Defekte behandelt. In [6], [7] und [8] wird analoge Fehlersimulation von aus dem Layout extrahierten Netzlisten zur Ermittlung der erforderlichen Testvektoren eingesetzt. Diese Vorgehensweise ist an die genaue Struktur der Standardzellen gebunden und führt, bezogen auf die konkreten zellinternen Fehler und im Vergleich zu den Ergebnissen von Automatischer Testpatterngenerierung (ATPG) bei Verwendung einer Standardzellbibliothek, zu einer sehr guten deterministischen Verbesserung der Fehlererkennung. Die Ergebnisse gelten für das spezielle Element der entsprechenden Standardzellbibliothek und können nicht unbedingt auf Zellen anderer Bibliotheken angewandt werden. Analoge Fehlersimulation ist sehr aufwendig und die Ergebnisse werden sehr stark von den eingestellten Simulationsparametern beeinflusst. Die Beeinflussung der Fehlererkennung innerhalb der Standardzelle durch die Beschaltung anderer Standardzellen der Netzliste wird nicht berücksichtigt.

Testpatterngenerierung auf Transistorlevel, wie sie in [1] diskutiert wird, hat sich praktisch nicht durchgesetzt, macht aber den Wert der Analyse des Transistornetzes deutlich.

Im Projekt Diana wurde die Frage nach der Ursache für die unvollständige Fehlererkennung auch bei ausgewiesener vollständiger Testüberdeckung diskutiert. Ziel der Arbeit war es, über die Erkennung der Fehler an den Ein- und Ausgängen der Standardzellen hinaus, die Behandlung aller zellinternen Fehler zu sichern. Darüber hinaus wurden Konstruktionen in einer digitalen Netzliste untersucht, die Einfluss auf die Fehlererkennung haben.

### VII.4.1. Fehlermodellierung

Fehlermodellierung dient der Abbildung von Defekten, die auf dem Wafer vorhanden sind, als Fehler in einer Netzliste. Defekte sind eine unerwünschte Materialbeschaffenheit wie zum Beispiel Schichtabscheidungen, Schichtreste oder Öffnungen in Schichten.

Etablierte Fehlermodelle sind das Stuck-at-0/1-Fehlermodell, das Transition-Delay-Fehlermodell und das IDDq-Fehlermodell. Bei diesen Fehlermodellen wird angenommen, dass mit der Definition von Fehlern an den Ein- und Ausgängen von Standardzellen in einer Logiknetzliste die Defekte ausreichend abgebildet sind. Diese Fehlermodelle kommen bei ATPG-Tools und digitalen Fehlersimulatoren zum Einsatz.

In einer Transistornetzliste werden die Defekte als Kurzschluss zwischen elektrischen Verbindungen oder als Unterbrechung von elektrischen Verbindungen abgebildet. Diese Fehler werden mit analogen Fehlersimulatoren bearbeitet.

Die Fehler in einer Logiknetzliste sind an die verwendeten Standardzellen gebunden und zählbar. Damit kann eine eindeutige Fehlererkennungsrate ermittelt werden. Die Anzahl der theoretisch möglichen Kurzschlussfehler in einer Transistornetzliste ist sehr groß und führt zu erheblichem Aufwand bei der analogen Fehlersimulation. Um diesen Aufwand zu verringern werden die Kurzschlussfehler ausgewählt, die praktisch auch auftreten können. Dabei werden Kurzschlüsse zwischen Leitungen ausgewählt, die im Layout benachbart sind. Eine Fehlererkennungsrate in der Elektrikebene kann dadurch nicht für eine Transistornetzliste, sondern immer für nur ein konkretes Layout bestimmt werden.

Zur Bewertung der Qualität der Fertigung wird unter anderem die Defektdichte ermittelt. Die Anzahl der Defekte auf einem Chip wird nicht gezählt, eine Defekterkennungsrate kann nicht ermittelt werden.

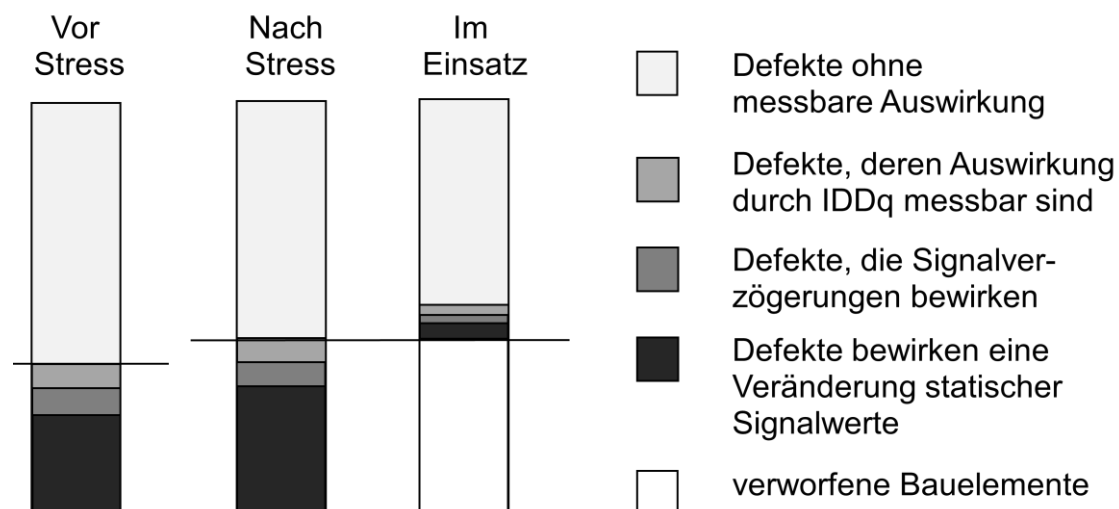


Abbildung VII-30: Defekte auf dem Wafer

Abbildung VII-30 zeigt qualitativ das Verhältnis von Defekten auf dem Wafer. Da die Defekte nicht gezählt werden, kann diese Darstellung nicht quantitativ genau sein. Nicht alle Defekte führen zu einer messbaren Veränderung der Schaltung. Ob Defekte zu einem Fehler führen, hängt von der Größe und der Position der Defekte ab. Grenzfälle werden durch Beanspruchung der Schaltung aktiviert. Um Ausfälle im Einsatz zu minimieren werden die Schaltkreise beim Hersteller einem Stress unterzogen. Werden beim Einsatz des Schaltkreises trotzdem Defekte so beeinflusst, dass sie einen Fehler bewirken, kommen die Diagnosemethoden zum Einsatz. Fehler, die im Fertigungstest erkannt wurden und Fehler, die im Einsatz des Schaltkreises aktiviert wurden sind gleichartig, es können also prinzipiell die gleichen Methoden für den Test verwendet werden. Die technischen Bedingungen beim Test und bei der Diagnose sind jedoch sehr unterschiedlich, so dass der Einsatz von Diagnosewerkzeugen einer speziellen Entwicklung bedarf.

#### **VII.4.2. Strukturtest**

Bei der Analyse der Schaltungsstruktur ist das Kriterium für die Auswahl der Testvektoren der Nachweis, dass die Schaltungsstruktur die Funktion fehlerfrei ausführt. Dabei werden alle Signalpfade über vorhandene Elementargatter und alle Pfade zwischen Versorgungsspannung und Masse aktiviert. Die Testvektoren werden so ausgewählt, dass bei einer ungewollten Veränderung der Schaltungsstruktur die Erzeugung des Funktionswertes maximal gestört und minimal unterstützt wird. Eine Veränderung der Struktur kann ein beliebiger durch einen Defekt verursachter Fehler, jedoch auch eine Veränderung der Eigenschaften der Elemente sein. Die so erzeugten Testvektoren sind geeignet zur Erkennung aller statisch wirksamen Fehler. Durch Anwendung der erforderlichen Testsequenzen werden auch die dynamischen Fehler erkannt.

#### **VII.4.3. Durchgeführte Untersuchungen**

Es wurden alle Elemente der Standardzellbibliothek einer ausgewählten Technologie und repräsentative Elemente aus Standardzellbibliotheken verschiedener Technologien untersucht. Dabei wurde die Testvektorerzeugung nach drei verschiedenen Methoden durchgeführt.

In der analytischen Methode wurden die Testvektoren für einen Strukturtest ermittelt. Dabei wurden die Testvektoren so bestimmt, dass alle Elemente und deren Verbindungen so aktiviert wurden, dass deren Beitrag zur Bildung des Funktionswertes nachgewiesen wird.

Ein zweiter Testsatz wurde mit einem ATPG-Tool berechnet, wobei unter Verwendung der Fehlermodelle für digitale Schaltungen durch das Tool ein Algorithmus abgearbeitet wurde.

In einer experimentellen Methode wurden die extrahierten Transistornetzlisten der Standardzellen mit einem analogen Fehlersimulator simuliert. Es wurden alle Signalkombinationen und alle Signalübergänge an den Eingängen der Standardzellen simuliert. Als Fehler wurden alle möglichen Unterbrechungen und Kurzschlussfehler in die extrahierte Transistornetzliste injiziert. Anschließend wurde



der Testsatz so ausgewählt, dass mit der minimalen Anzahl von Testvektoren die maximale Anzahl von Fehlern erkannt wird. Zur Erfüllung dieser Aufgabe erfolgte eine Weiterentwicklung des analogen Fehlersimulators in der Variante „aFSIM digital“.

Durch Vergleich der Ergebnisse und des nötigen Aufwandes kann die Leistungsfähigkeit der drei Verfahren bewertet werden.

#### VII.4.4. Klassifizierung von Standardzellen

Die Gültigkeit der Fehlermodelle wird durch zwei Komponenten bestimmt. Neben der Art der Behandlung der Fehler durch den ATPG-Algorithmus hat die Beschreibung der Standardzellen eine sehr große Bedeutung. Zur Beschreibung der Standardzellen gehören unter anderem das Layout, die Transistornetzliste und eine Funktionsbeschreibung. Das Layout und die Transistornetzliste haben einen direkten Bezug zueinander. Aus der Funktionsbeschreibung kann jedoch nicht sicher darauf geschlossen werden, welche konkrete Transistorschaltung vorliegt.

Die Standardzellen können in vier Kategorien eingeteilt werden.

1. Elemente mit den einfachen Grundfunktionen NAND, NOR, AND, OR und NOT,
2. CMOS-Komplexgatter, welche eine Funktion mit mehreren AND-, OR- und NOT Komponenten realisieren,
3. Standardzellen mit internen Verzweigungen und Maschen über logische Elemente ,
4. Standardzellen mit Tristate-Treibern oder Signalschaltern.

##### VII.4.4.1. Elemente mit einfachen Grundfunktionen

Als Repräsentant dieser Elemente wird ein NAND mit 3 Eingängen NAND3 diskutiert. Tabelle VII-5 zeigt den vollständigen Testsatz für das NAND3.

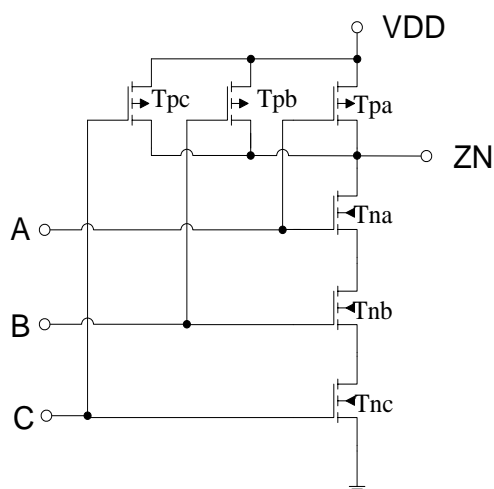


Tabelle VII-5: Testsatz für NAND3

ABC	ZN
1 1 1	0
0 1 1	1
1 0 1	1
1 1 0	1

Abbildung VII-31: Transistorschaltung des NAND3

In der Transistorschaltung des NAND3 (Abbildung VII-31) wird sichtbar, dass zur Erzeugung des Signalwertes 0 am Ausgang ZN ein Signalpfad über alle n-Kanal-Transistoren aktiviert werden muss. Alle p-Kanal-Transistoren müssen dabei gesperrt sein. Es gibt nur einen Vektor, mit dem der Test der Stuck-at-0-Fehler an den Eingänge A, B, C und des Stuck-at-1-Fehlers am Ausgang ZN durchgeführt werden kann. In diesem Vektor haben alle Eingänge den Wert 1.

Für den Test der Stuck-at-1-Fehler an den Eingängen des NAND3 ist jeweils ein einziger p-Kanal-Transistor leitend. Dabei sperrt jeweils nur ein n-Kanal-Transistor. Damit liegt die härteste einstellbare Testbedingung vor.

Der Stuck-at-0-Fehler an ZN wird durch drei Testvektoren getestet. Jede andere Signalkombination bei der nur ein Eingang den Wert 1 hat ist eine Entspannung der Testbedingungen, so dass eine Verbesserung der Testqualität dadurch nicht erreicht wird.

#### **VII.4.4.2. CMOS – Komplexgatter**

Als Repräsentant für diese Elemente wird eine NOR Verknüpfung zweier AND mit zwei Eingängen gewählt. Abbildung VII-32 zeigt das Symbol für diese Verknüpfung, Abbildung VII-33 zeigt die Transistorschaltung. Als Beispiel soll hier der Test der Stuck-at-0/1-Fehler am Eingang A1 diskutiert werden.

Um den Stuck-at-0-Fehler an A1 zu testen, müssen die Eingänge A1 und A2 mit dem Signalwert 1 stimuliert werden. Das Gatter AND2 muss den Ausgangswert 0 erzeugen. Dazu sind an den Eingängen B1 und B2 die Signalwerte 01, 10 oder 00 möglich. In der Transistorschaltung ist sichtbar, dass die einzelnen Operatoren nicht durch physisch getrennte Transistorgruppen realisiert sind. Die Signalpfade der beiden AND-Operatoren sind praktisch parallel geschaltet. Bei den Signalkombinationen 01 und 10 sperrt nur einer der beiden n-Kanal-Transistoren im AND2 und nur einer der beiden p-Kanal-Transistoren ist leitend. Als leistungsfähigster Testvektor wird eine Signalkombination gesucht, bei der die Erzeugung des fehlerfreien Ausgangssignals im Fehlerfall am wenigsten unterstützt und am meisten gestört wird. Diese Anforderung wird durch die Signalkombination 00 an den Eingängen B1 und B2 erfüllt.

Um den Stuck-at-1-Fehler an A1 zu testen, ist die Signalkombination 01 an den Eingängen A1 und A2 erforderlich. Das Gatter AND2 muss ebenfalls den Ausgangswert 0 erzeugen. Durch die Zusammenschaltung der PMOS-Transistoren entstehen verschiedene Signalpfade, die zu erhöhten Testanforderungen führen. Von der Versorgungsspannung VDD zum Ausgang ZN gibt es die Signalpfade VDD-Tpa1-Tpb2-ZN, VDD-Tpa1-Tpb1-ZN, VDD-Tpa2-Tpb2-ZN und VDD-Tpa2-Tpb1-ZN. Für einen vollständigen Test müssen also alle diese Signalpfade getestet werden. Die dafür zu verwendenden Signalkombinationen an B1 und B2 sind 10 und 01. In der Funktionsbeschreibung der Elemente ist die Struktur im Element nicht sichtbar, so dass das ATPG-Tool die erhöhten Testanforderungen nicht erfüllen kann. Tabelle VII-6 zeigt die Testvektoren für das Element, welche bei Berücksichtigung der inneren Struktur entstehen und Testpattern die von einem ATPG-Tool unter Verwendung der Funktionsbeschreibung generiert werden. Als Ergebnis der ATPG wird nur ein Beispiel von mehreren Varianten gezeigt.

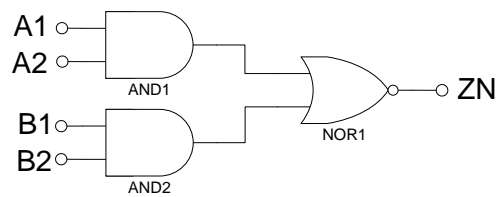


Abbildung VII-32: Symbol des AND-NOR

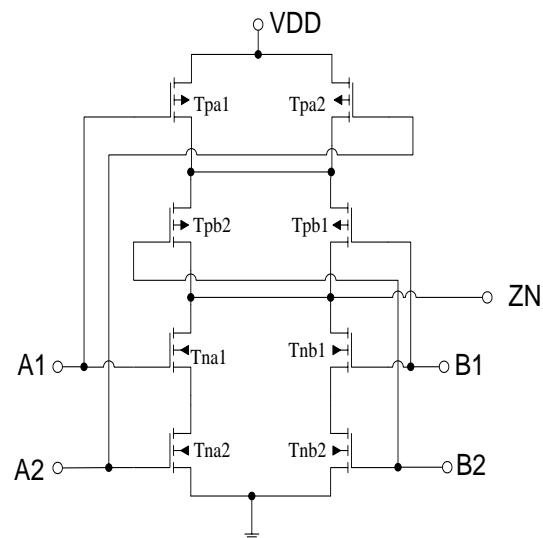


Abbildung VII-33: Transistorschaltung des AND-NOR

Tabelle VII-6: Pattern für AND-NOR

Testvektoren nach Analyse der Struktur					Testpattern als Ergebnis einer ATPG				
A1	A2	B1	B2	ZN	A1	A2	B1	B2	ZN
1	1	0	0	0	1	1	0	0	0
0	0	1	1	0	0	1	1	1	0
0	1	0	1	1	0	1	1	0	1
0	1	1	0	1	1	0	0	1	1
1	0	0	1	1					
1	0	1	0	1					

Die Testvektoren für Stuck-at-0/1-Fehler stimulieren auch Kurzschlüsse zwischen den Transistoranschlüssen. Ein Kurzschluss wird seine größte Wirkung zeigen, wenn

der kleinste Widerstand zur Spannungsversorgung eingestellt ist. Zum Beispiel für den Test des Kurzschlusses zwischen Gate und Source des Transistors Tpb1 ist der leistungsfähigste Testvektor A1,A2,B1,B2 = 0001. Dieser Testvektor, der im Testsatz für den Test der Signalpfade nicht enthalten ist, bewirkt durch Aktivierung zweier parallel geschalteter Transistoren den kleinsten Widerstand zwischen der Spannungsversorgung und dem Fehlerort. Bei der Anwendung dieses Testvektors kann der Kurzschluss einen größeren Widerstand haben als bei allen anderen Testvektoren, um den Signalwert des Komplexgatters zu verändern. Simulationen haben gezeigt, dass Kurzschlüsse mit einem Widerstand im Bereich von einigen k $\Omega$  auch bei Anwendung von Testpattern für den Strukturtest den Signalwert der Schaltung verändern. Durch Anwendung des Testvektors 0001 wird der Widerstandswert, der den Signalwert verändert, um etwa 1 k $\Omega$  erhöht. Die Art der Fehler wird durch den Testsatz zum Test der Schaltungsstruktur vollständig berücksichtigt. Die Erkennbarkeit einiger Fehler jedoch kann durch weitere Testvektoren verbessert werden.

Wird zur Erkennung von Kurzschlussfehlern auch die IDDq-Messung eingesetzt, ist eine Erweiterung des Testsatzes für den Strukturtest nicht notwendig.

#### **VII.4.4.3. Elemente mit internen Verzweigungen über logische Elemente**

Für diese Elemente ist das exklusive Oder ein sehr aussagekräftiger Repräsentant. Tabelle VII-7 zeigt die Funktionsbeschreibung und die vier Testsätze die daraus abgeleitet werden können. Bei dieser Beschreibung wird durch ein ATPG-Tool eine der vier möglichen Varianten eines Testsatzes entstehen. Das gleiche Ergebnis entsteht, wenn die Funktionsbeschreibung unter Verwendung von boolschen Funktionsprimitiven wie xor, and, und nor erfolgt.

Tabelle VII-7: Funktionstabelle des XOR und daraus abgeleitete Testsätze

Funktion	Test1	Test2	Test3	Test4
AB Z	AB Z	AB Z	AB Z	AB Z
0 0 1		0 0 1	0 0 1	0 0 1
0 1 1	0 1 1		0 1 1	0 1 1
1 0 1	1 0 1	1 0 1		1 0 1
1 1 0	1 1 0	1 1 0	1 1 0	

Die Struktur der inneren Schaltung der Standardzelle liegt als Transistorschaltung vor. Dabei sind verschiedene Schaltungen möglich. Abbildung VII-34 zeigt eine Variante. An dieser Schaltung ist sichtbar, dass für einen vollständigen Test vier Signalpfade in der Transistorschaltung aktiviert werden müssen. Es sind also alle vier Vektoren der Funktionsbeschreibung für den Test erforderlich.

In anderen Schaltungsausführungen des XOR kann die Transistorschaltung direkt auf logische Elemente abgebildet werden. Abbildung VII-35 zeigt eine solche Gatterschaltung. In dieser Schaltung verzweigen die Eingangssignale auf verschiedene logische Gatter. Die Verzweigungen werden vor dem Ausgang als Masche zusammengeführt. Um die Testvektoren für die Stuck-at-0/1-Fehler am

Eingang A zu generieren, wird zunächst ein Signalpfad von Eingang A über die Gatter NOR1 und NOR2 nach Z aktiviert. Dazu muss Eingang B mit dem Signalwert 0 stimuliert werden. Um beide Stuck-at-0/1-Fehler am Eingang von NOR1 zu testen wird der Eingang A mit den Werten 1 und 0 beschaltet, es entstehen zwei Testvektoren. Um alle Signalpfade in der Gatterschaltung zu testen muss nun ein Signalpfad vom Eingang A über die Gatter AND1 und NOR2 aktiviert werden. Dazu wird der Eingang B mit dem Signalwert 1 beschaltet. Um die Stuck-at-0/1-Fehler am Eingang von AND1 zu testen wird Eingang A mit den Signalwerten 1 und 0 beschaltet, es entstehen weitere zwei Testvektoren. Die Prozedur kann auch für Eingang B durchgeführt werden, wobei ebenfalls vier Testvektoren entstehen. Tabelle VII-8 zeigt die Signalpfade und die entstehenden Testvektoren von den Eingängen für die Gatterschaltung des XOR. Beim Test muss der vollständige Testsatz mit vier Testvektoren nur einmal ausgeführt werden, da mehrere Fehler gleichzeitig getestet werden.

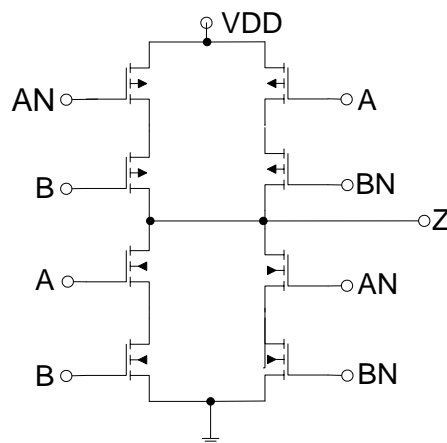


Abbildung VII-34: Transistorschaltung eines XOR

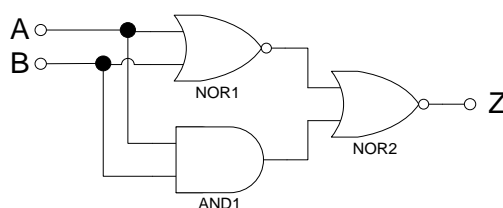


Abbildung VII-35: Gatterschaltung eines XOR

Tabelle VII-8: Signalpfade und Testvektoren für XOR

Test für Eingang A		Test für Eingang B	
	A B Z		A B Z
über NOR1	1 0 1	über NOR1	0 1 1
über NOR1	0 0 0	über NOR1	0 0 0
über AND1	1 1 0	über AND1	1 1 0
über AND1	0 1 1	über AND1	1 0 1

Es ist notwendig, die Informationen über die innere Struktur der Elemente an ein ATPG-Tool zu übergeben, so dass die Generierung der vollständigen Testsätze nach einem Algorithmus automatisch erfolgen kann.

#### VII.4.4.4. Elemente mit Tristate-Treibern

Tristate-Treiber spielen eine besondere Rolle, da sie nicht der booleschen Algebra folgen. Abbildung VII-36 zeigt einen mit Tristate-Treibern aufgebauten Multiplexer.

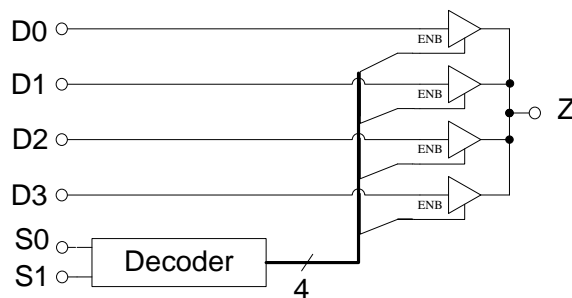


Abbildung VII-36: Schaltung eines Multiplexer

In den Standardzellbibliotheken sind verschiedene Beschreibungen der Multiplexer möglich. Die Beschreibung kann als Wertetabelle, als Schaltung mit Tristate-Treibern oder als Verknüpfung der Funktionsprimitive and, or und not erfolgen. Abhängig von der Art der Beschreibung entstehen bei der ATPG verschiedene Testvektoren. Bei der Generierung von Testvektoren nach dem Stuck-at-0/1-Fehlermodell für eine so aufgebaute Standardzelle werden etwaige Beeinflussungen zwischen den Eingängen D0 bis D3 nicht berücksichtigt, so dass zum Beispiel für den Test von Stuck-at-0/1-Fehlern am Eingang D0 die Signalwerte an den Eingängen D1, D2 und D3 keinen Einfluss haben [6].

Um das beste Testergebnis zu erhalten müssen die Testvektoren nach einer Regel generiert werden, die nicht in den Standardzellkatalogen beschrieben ist. Die Testvektoren sind so zu gestalten, dass jeweils die Eingänge, deren Signal nicht zum Ausgang geschaltet wird, mit dem invertierten Signalwert des Eingangs stimuliert wird, dessen Signal zum Ausgang geschaltet wird. Wenn die Erzeugung dieser Testvektoren nicht möglich ist, müssen zur Erreichung desselben Ergebnisses

mehrere alternative Testvektoren verwendet werden, bei denen das invertierte Signal des zum Ausgang geschalteten Dateneinganges an jeden anderen Dateneingang geschaltet wurde. Auf diese Weise entstehen Testvektoren, bei deren Anwendung alle erkennbaren Fehler innerhalb des Multiplexer erkannt werden. In Tabelle VII-9 sind die benötigten Testvektoren für einen Multiplexer mit 4 Dateneingängen gelistet. Für zwei Vektoren sind alternative Testsätze gelistet, die dann zum Einsatz kommen wenn der benötigte Testvektor nicht generierbar ist.

Tabelle VII-9: Testvektoren für einen Multiplexer

Benötigte Testvektoren	Alternative Testsätze
D0 D1 D2 D3 S0 S1 Z	D0 D1 D2 D3 S0 S1 Z
0 1 1 1 0 0 0	0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
1 0 0 0 0 0 1	
1 0 1 1 1 0 0	0 0 1 1 1 0 0 1 0 0 0 1 0 0
0 1 0 0 1 0 1	
1 1 0 1 0 1 0 0 0 1 0 0 1 1	
1 1 1 0 1 1 0 0 0 0 1 1 1 1	

#### VII.4.5. Zellübergreifende Aspekte in einer Netzliste

Die bisherigen Betrachtungen beschränken sich auf die innere Struktur von Standardzellen. Um eine vollständige Bewertung der Qualität von Testpattern zu erhalten ist die Bewertung der Beziehung zwischen verschiedenen Standardzellen und derer Verbindungen in einer Netzliste erforderlich.

##### VII.4.5.1. Zusammenschaltung von Standardzellen

Die Erkennung von Kurzschlüssen innerhalb von Standardzellen wird von der Signalerzeugung durch die Standardzellen an den Eingängen der zu testenden Standardzelle beeinflusst.

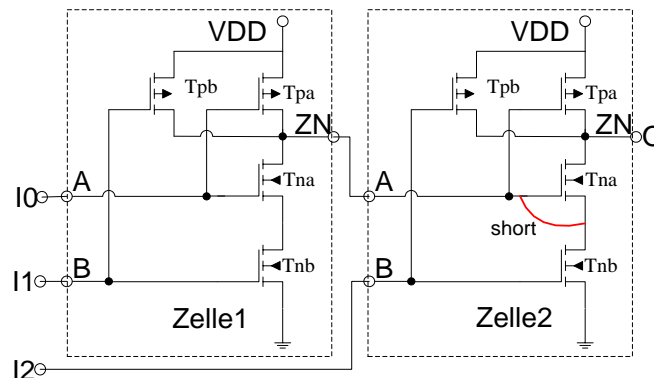


Abbildung VII-37: Verknüpfung zweier NAND2

Abbildung VII-37 zeigt eine Zusammenschaltung zweier NAND2. Es wird angenommen, dass ein Fehler als Kurzschluss zwischen Gate und Source des Transistors Tna in Zelle2 vorhanden ist. Für diese Schaltung ist zu diskutieren, mit welchen Testvektoren der Kurzschluss am besten erkannt wird [3].

Bei Anwendung des Stuck-at-0/1-Fehlermodells besteht der kürzeste Testsatz zum Test der Fehler an Zelle2 aus den Vektoren  $I_0, I_1, I_2 = (111), \{(011) \text{ oder } (101)\}$  und  $\{(010) \text{ oder } (100)\}$ .

Wird der Kurzschluss als Stuck-at-0-Fehler am Ausgang O sichtbar, ist für die Eingrenzung des Fehlerortes der Vektor 001 geeignet, da er Fehler an den Eingängen von Zelle1 ausschließt. Dieser Vektor bewirkt den kleinsten Widerstand zwischen der Spannungsversorgung und dem Kurzschluss, da er die beiden parallel geschalteten p-Kanal Transistoren in Zelle1 aktiviert. Dadurch kann sich der Signalwert 1 gegenüber dem über den Kurzschluss geleiteten 0-Pegel besser durchsetzen. Ein fehlerhafter Wert am Ausgang O entsteht also abhängig vom Testvektor und von der Größe des Kurzschlusswiderstandes. Es ist also möglich, dass der Kurzschluss bei den drei Testvektoren 011, 101 und 001 eine Veränderung des Signalwertes an O bewirkt, es ist aber auch möglich, dass nur bei den Vektoren 011 und 101 ein fehlerhafter Signalwert entsteht. Beim Versuch der Eingrenzung des Fehlerortes würde das Fehlerbild nicht mehr sichtbar sein.

#### VII.4.5.2. Schaltungen mit Tristate-Treibern

Sofern Tristate-Treiber nicht innerhalb eines Elementes angeordnet sind, sondern innerhalb der Netzliste zusammenschaltet sind, werden nicht die Testvektoren generiert, mit denen das beste Testergebnis erreicht wird. Wo immer es möglich ist auf Tristate-Treiber zu verzichten, ist eine Lösung ohne diese Elemente zu bevorzugen. Sind Tristate-Treiber unverzichtbar, müssen sie in einer Teilschaltung angeordnet werden, die als Element zu beschreiben ist. Dieses Element kann dann so behandelt werden, dass die erforderlichen Testvektoren entstehen.



### VII.4.5.3. Berücksichtigung des Layout des Schaltkreises

Kurzschlüsse zwischen verschiedenen Signalleitungen einer Netzliste können mit dem Bridging-Fehlermodell bearbeitet werden. Bei der Layoutgenerierung für digitale Schaltungen werden Metallebenen über die Standardzellen gelegt, so dass Leiterbahnen über Standardzellen geführt werden, die nicht mit der Standardzelle verbunden werden. Dadurch kommt es zu Nachbarschaften von Signalleitungen in der Netzliste mit Leitungen, die sich innerhalb von Standardzellen befinden und in der Netzliste nicht sichtbar sind. Zur Ermittlung solcher Leitungspaare ist eine Analyse des Layouts der gesamten Schaltung erforderlich. Zur Generierung von Testvektoren zum Ausschluss von Kurzschlüssen zwischen solchen Leitungspaaren muss der Signalwert der Signalleitung, die über die Standardzelle geführt wird, bei der Berechnung der Testsignale für die Standardzelle berücksichtigt werden.

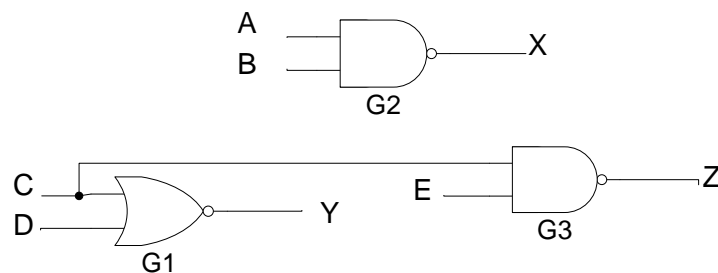


Abbildung VII-38: Elemente einer Schaltung ohne funktionellem Zusammenhang

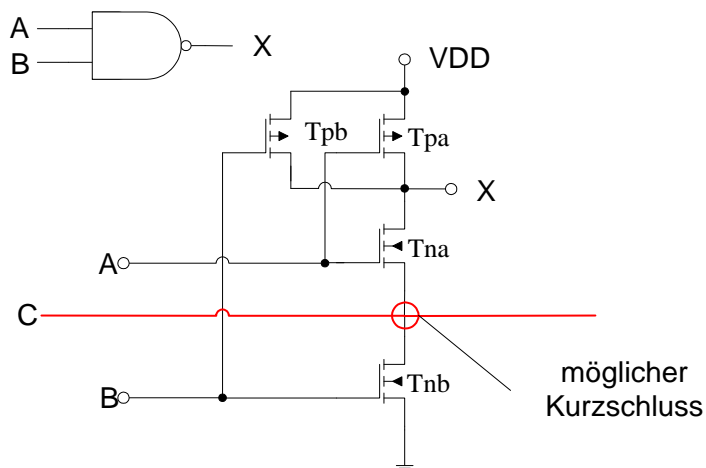


Abbildung VII-39: Kurzschluss mit zellinterner Leitung

Tabelle VII-10: erweiterte Testvektoren

C	A B	X
1	1 1	0
-	1 0	1
-	0 1	1

Wenn beispielsweise die in Abbildung VII-38 dargestellten Elemente im Layout so angeordnet sind, dass die Signalleitung C über die Zelle G2 geführt werden muss, um die Eingänge der Zellen G1 und G3 zu verbinden, entsteht ein potentieller Kurzschluss zwischen internen Leitungen von G2 und der Signalleitung C (Abbildung VII-39). Interne Leitungen von NAND-Gattern befinden sich zwischen den n-Kanal-Transistoren. Um einen Kurzschluss zwischen einer internen Leitung des NAND und der Signalleitung C erkennbar zu machen, muss die Signalleitung C den Wert 1 haben wenn der Ausgangswert 0 des NAND G2 erzeugt wird (Tabelle VII-10). Diese hier an einem sehr einfachen Beispiel beschriebene Konstellation ist besonders bei großen Standardzellen mit einer großen Anzahl interner Leitungen interessant.

#### **VII.4.6. Dynamische und potentielle Fehler**

Die bisherigen Ausführungen haben sich auf das Stuck-at-0/1-Fehlermodell bezogen. Bezüglich der erforderlichen Informationen über die Schaltung und die erreichbare Qualität der Testvektoren gelten die Aussagen vollständig auch für das Transition-Delay-Fehlermodell und für das IDDq-Fehlermodell. Besonderheiten werden nachfolgend erläutert.

##### **VII.4.6.1. Transition-Delay-Fehler**

Zum robusten Test von Transition-Delay-Fehlern ist vorzugsweise ein Signalpfad zu aktivieren, wobei zwischen launch und capture an nur einem Eingang ein Signalwechsel stattfindet. Finden mehrere Signalwechsel im Eingangsvektor statt, kann aufgrund verschiedener Signallaufzeiten nicht garantiert werden, dass die erforderliche Sequenz an allen Elementen angewandt wird. Wird nur ein einzelnes Element bewertet, ist der gleichzeitige Signalwechsel erlaubt, wenn der Vektor eingestellt wird, der aus den dominierenden Signalwerten besteht. Zum Beispiel ist für ein NAND3 der Wechsel der drei Eingangssignale von (000) zu (111) zulässig, da der Funktionswert erst mit dem Wechsel des letzten Eingangssignals verändert wird, und damit die längste Verzögerungszeit wirksam wird. Der Wechsel von (111) zu (000) ist jedoch nicht zulässig, da der Funktionswert sich bereits bei Veränderung des ersten Eingangssignals ändert, wodurch die kürzeste Verzögerungszeit wirksam wird und die Laufzeit der nachfolgend veränderten Eingangssignale nicht wirksam wird.

Bei Pattern nach dem Path-Delay-Fehlermodell verläuft ein Signalwechsel entlang eines Pfades, wobei keine Verfälschungen durch Impulse innerhalb der Schaltung stattfinden.

##### **VII.4.6.2. Vektoren zur IDDq-Messung**

Da die Signalgenerierung durch Parallelschaltung mehrerer Transistoren in verschiedenen Schaltungen die Sichtbarkeit eines Kurzschlusses als statische Abweichung eines Signals von seinem Sollwert verbessern oder verhindern kann, ist die Auswahl der geeigneten Testvektoren kompliziert. Abhilfe kann durch den Einsatz der IDDq-Messung geschaffen werden.

Für die IDDq-Messung müssen die Fehler stimuliert werden, brauchen aber nicht zu einem beobachtbaren Schaltungsausgang propagiert werden. Dadurch kann mit einer geringen Anzahl von Testvektoren eine sehr hohe Testüberdeckung erreicht werden. Die Anzahl der Testvektoren für IDDq-Fehler für eine Standardzelle ist größer als die Anzahl der Testvektoren für das Stuck-at-0/1-Fehlermodell.

Für ein NAND2 entstehen für das Stuck-at-0/1-Fehlermodell die Eingangsvektoren 11, 10 und 01. Für die IDDq-Messung entsteht zusätzlich der Eingangsvektor 00. Die Stimulation der möglichen Kurzschlüsse innerhalb der Standardzelle wird durch die Eingangsvektoren 00 und 11 gesichert. Das Ziel bei der Generierung von Pattern für die Short Overvoltage Evaluation (SHOVE) oder Spannungsstress besteht darin, mit wenigen Vektoren eine sehr große Anzahl von Transistoren zu belasten. Um dieses Ziel zu erreichen sind die Eingangsvektoren 00 (alle Eingangssignale sind 0) und 11 (alle Eingangssignale sind 1) für ein NAND2 ausreichend [2]. Vorteilhaft ist also ein ATPG-Programm, welches bezüglich dieser verschiedenen Anforderungen konfiguriert werden kann. Tabelle VII-11 zeigt die verschiedenen Testvektoren für die verschiedenen Anforderungen.

Tabelle VII-11: Pattern für NAND2

Testpattern für Stuck-at-0/1- Fehler	Testpattern für SHOVE	Testpattern für IDDq
A B C ZN	A B C ZN	A B C ZN
0 1 1 1		0 1 1 1
1 0 1 1		1 0 1 1
1 1 0 1		1 1 0 1
1 1 1 0	1 1 1 0	1 1 1 0
	0 0 0 1	0 0 0 1

#### VII.4.7. Zusammenfassung

Es wurden für Elemente von Standardzellbibliotheken verschiedener Technologien nach drei verschiedenen Vorgehensweisen nach dem Bewertungskriterium der jeweiligen Methode die vollständigen Testsätze generiert. Die Bewertung der Ergebnisse erfolgt nach dem Kriterium, wie vollständig die internen Fehler der Standardzelle durch die Testsätze erkannt werden, wie zellübergreifende Aspekte berücksichtigt werden und wie aufwendig das Generierungsverfahren ist. Gegenübergestellt wurde eine analytische Methode (Analyse der Schaltung und Test der Strukturelemente der Schaltung), die algorithmische Methode (ATPG bei Anwendung der digitalen Fehlermodelle) und die experimentelle Methode (analoge Fehlersimulation aller Übergänge aller Eingangssignalkombinationen der Standardzellen. Tabelle VII-12 zeigt die Bewertung einiger Aspekte.

Tabelle VII-12: Bewertung der Testgenerierungsmethoden

	Analytische Methode (Strukturtest)	Algorithmische Methode (ATPG)	Experimentelle Methode (analoge Fehlersimulation)
Erkennung der zellinternen Fehler	vollständig	Unvollständig	vollständig
Berücksichtigung zellübergreifender Besonderheiten	Ist gegeben	nicht vorgesehen	nicht möglich
Aufwand	Einmalige Analyse und Beschreibung der Schaltung	Sehr gering	Charakterisierung jeder Standardzellbibliothek; sehr hoher Rechenaufwand

Insgesamt ist die analytische Methode den anderen Methoden überlegen. Bezüglich der erreichten Testüberdeckung werden auf algorithmischem Wege die schlechtesten Ergebnisse erreicht.

#### **VII.4.8. Ausblick**

Eine analytische Methode ist immer leistungsfähiger als eine experimentelle Methode. Aus diesem Grunde kann die analoge Fehlersimulation digitaler Standardzellen nicht das Mittel der Wahl sein.

Anstrebenswert ist eine algorithmische Methode. Eine analytische Methode ist die Vorstufe für eine algorithmische Lösung. Auf analytischem Wege wird festgelegt, was ein Algorithmus leisten soll. Im Ergebnis von Diana ist es nun möglich die Testgenerierungsalgorithmen entsprechend weiterzuentwickeln.

Mit den verfügbaren ATPG Tools und den Ergebnissen von Diana kann jedoch die Leistungsfähigkeit der Tools beeinflusst werden, ohne deren Weiterentwicklung abzuwarten. Dazu müssen für die Tools geeignete Informationen in geeigneter Weise bereitgestellt werden.

Die Erkennung aller internen Fehler von Standardzellen wird durch eine geeignete Beschreibung der Standardzellen gesichert. Dabei ist die Kompatibilität zu den etablierten Fehlermodellen und zu allen anderen Methoden des Design for Test vorhanden. Neue Fehlermodelle und die Testpatterngenerierung in der Elektrikebene sind nicht erforderlich.

Auch die Berücksichtigung zellübergreifender Aspekte kann durch Abarbeitung zusätzlicher Scripts und durch Einstellung bestimmter Steuerparameter der ATPG erreicht werden.

Nicht zuletzt wird das Testergebnis durch Vermeidung von Schaltungsstrukturen beeinflusst, die durch ATPG-Tools nicht zufriedenstellend bearbeitet werden können.

#### VII.4.9. Literatur

- [1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of U. Hübner, U. Gläser and H. T. Vierhaus, „Mixed Level Hierarchical Test Generation for Transition Faults and Overcurrent Related Defects“, in Proc. IEEE Int. Test Conf. 1992.
- [2] J. T.-Y. Chang, E. J. McCluskey, "SHOrt Voltage Elevation (SHOVE) test for weak CMOS ICs." In 15th IEEE VLSI Test Symposium (VTS'97), April 27-May 1, 1997, Monterey, California, USA;
- [3] E.J. McCluskey, C.-W. Tseng, „Stuck-fault test vs. actual defects“, in Proc.ITC, 2000.
- [4] B. Benware, C. Schuermyer, S. Ranganathan, R. Magde, NTamarapalli, K.-H. Tsai, J. Rajski, „Impact of Multiple-Detect Test Patterns on Product Quality“, in Proc.ITC, 2003.
- [5] S. Spinner, I. Polian, P. Engelke, B. Becker, M. Keim, W.-T. Cheng, „Automatic test pattern generation for interconnect open defects“, in VLSI Test Symp., 2008.
- [6] F. Hapke, R. Krenz-Baath, A. Glowatz, J. Schloeffel, H. Hashempour, S. Eichenberger, C.Hora, D.Adolfsson, „Defect-oriented Cell-Aware ATPG and Fault Simulation for Industrial Cell Libraries and Designs“, in International Test Conference, 2009.
- [7] F. Hapke, W. Redemund, J. Schloeffel, R. Krenz-Baath, A. Glowatz, M.Wittke, H. Hashempour, S. Eichenberger, „Defect-oriented cell-internal testing“, in International Test Conference, 2010.
- [8] F. Hopsch, M. Lindig, B. Straube, W. Vermeiren, „Characterization of Digital Cells for Statistical Test“, in IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, 2011.
- [9] L. Grobelny, „Digitale Fehlermodelle und ihre Leistungsfähigkeit in der Praxis“, in DASS, 2012.
- [10] Lothar Grobelny, „Nutzung der vollständigen Beschreibung der Schaltungsstruktur für die automatische Testpatterngenerierung“, in „Testmethoden und Zuverlässigkeit von Schaltungen und Systemen“ 25. GI/GMM/ITG-Workshop Dresden, 24. bis 26. Februar 2013

## VIII. Ergebnisse in Arbeitspaket 2: Bauelemente Selbsttests und Selbstreparatur als Basis für die Diagnose des Gesamtsystems

Das Arbeitspaket 2 befasst sich mit den Maßnahmen auf Bauelemente-Ebene die getroffen werden müssen um eine Diagnose auf Systemebene zu ermöglichen. Das sind im Wesentlichen Maßnahmen zum Selbsttest und in einem weiteren Schritt dann Maßnahmen zur Selbstreparatur.

Es hat sich gezeigt, dass viele Maßnahmen auf Chipebene zum Test der Bausteine so erweitert werden können, dass sie auf Systemebene zur Verfügung gestellt werden können. Einige bereits heute im Produktionstest angewendete Verfahren lassen sich so erweitern, dass sie auch in der Systemdiagnose angewendet werden können. An anderer Stelle müssen neue Verfahren eingeführt werden. Diese können dann in der Regel sowohl für den Produktions-Test und Diagnose als auch für die Systemdiagnose verwendet werden.

Es gibt keine universelle Maßnahme, die sich für alle Arten von Schaltungen eignet. Die gewählten Methoden müssen an die jeweilige Kategorie der Schaltung angepasst werden. Um eine optimale diagnostische Auflösung zu erzielen, kann auch eine Kombination von verschiedenen Methoden nötig sein.

Generell kann man folgende drei Klassen von Schaltungen unterscheiden:

- Speicher
- Analoge Schaltungen
- Digitale Schaltungen

Bei Speichern wird der Selbsttest schon verbreitet eingesetzt. Man kann ihn auch relativ einfach auf Systemebene zugänglich machen. Hier ist auch die Selbstreparatur am weitesten entwickelt. Von den betrachteten Methoden hat sich das Verfahren zur automatischen Fehlerkorrektur (ECC = Error Code Correction) als für die Systemanwendung am effektivsten erwiesen.

Bei Digitalschaltungen zeigt sich für die Fehlerdiagnose eine Kombination aus Software basierten Selbsttest und aus zufallsgenerierten Testmustern am effektivsten. Die Forschungsarbeiten im Projekt Diana haben gezeigt, wie die Selbstreparatur grundsätzlich möglich ist. Im industriellen Bereich wird die Selbstreparatur nach sorgfältiger Kosten/Nutzenanalyse nicht flächendeckend sondern selektiv für kritische Schaltungsbereiche angewandt.

Im analogen Bereich wird der Produktionstest bis jetzt stark durch den Einsatz von Halbleitertestautomaten dominiert, die für die Systemdiagnose nicht zur Verfügung stehen. Im Projekt Diana wurden für verschiedene Arten von Analogschaltungen Selbsttestmethoden untersucht und entwickelt, die langfristig die Testautomaten basierten Tests sowohl im Produktionstest als auch für die Systemdiagnose ablösen werden.

### **VIII.1. Ergebnisse zur Aufgabe 2.1: Algorithmen zur analogen Messdatenerfassung und –auswertung mit eingeschränkten Hardware Ressourcen**

In Aufgabe 2.1 werden Algorithmen sowie Schaltungen bereitgestellt, die es erlauben Halbleiterbauelemente mit einer dem Produktionstest vergleichbaren Qualität und unabhängig von der jeweiligen Systemumgebung zu testen

Ziel dieser Algorithmen und Schaltungen ist es dem Automobilhersteller zu ermöglichen sporadische, nichtsystematische Fehler und Systemausfälle zu erkennen und zu lokalisieren.

Da Steuermodule oftmals aus weiteren Submodulen aufgebaut sind, in welchen wiederum mehrere Halbleiterbausteine verbaut sind, ist es für den Automobilhersteller äußerst schwierig, die korrekte Funktion einzelner Komponenten zu überprüfen, was dazu führt, daß oft, nur aufgrund eines nicht konkret belegbaren Verdachts, komplette Module ausgetauscht werden, was nicht immer zu einer Beseitigung des Fehlers führt.

Aufgrund der Tatsache, daß diese Funktionalität auf möglichst vielen Chips vorhanden sein soll, ist es von Vorteil wenn die dafür notwendigen Verfahren bzw. Schaltungen dafür ausgelegt sind auf Systemen mit eingeschränkten Hardwareressourcen lauffähig zu sein. Damit wird erreicht, daß weniger komplexe Bausteine auch über Selbstdiagnosefähigkeiten verfügen. Eine lückenlose Überwachung aller Systemkomponenten wird dadurch möglich.

Bei den heute üblichen komplexen elektronischen Steuereinheiten ist eine lückenlose Überwachung nötig, da bei der Komplexität Fehler oft nicht auf Anhieb gefunden werden können.

Die Messungen, die durchzuführen sind, umfassen analoge sowie Mixed-Signal Größen. Würden diese Größen ohne weiteres dem Automobilhersteller zur Verfügung gestellt, so würde dies bedingen, daß die Module aufwendige Referenzschaltungen enthalten müssen. Weiter wären Entkoppelverstärker nötig, um die analogen Meßsignale nicht zu belasten, was die Meßwerte verfälschen würde. Dies alles würde die Kosten der Module und ihre Komplexität steigen lassen.

Um dies zu vermeiden, müssen die entwickelten Konzepte und Schaltungen möglichst ohne Referenzen funktionsfähig sein. Die Meßdaten sollten zudem in digitaler Form zur Verfügung gestellt werden um schon vorhandene Datenbusse zu nutzen. Damit können, im laufenden System, die Messungen ohne weiteres angestoßen werden. Die digitalen Meßdaten können dann von den übergeordneten Systemkomponenten ausgewertet werden.

Über die analogen Parameterwerte des Mixed-Signal Tests werden Statistiken geführt, die genauen Aufschluß über die fertigungsbedingten Schwankungen geben und so für jeden Parameter ein zulässiges Zielfenster definieren. Während des Selbsttests werden einige dieser Parameter mit auf dem Baustein vorhandenen

Funktionsblöcken gemessen, sollten unzulässige Abweichungen feststellbar sein, dann kann ein Alarm vom System oder vom Baustein ausgelöst werden

Zusammenfassend kann also gesagt werden:

1. Diagnosemöglichkeiten werden im System gebraucht
2. Analoge Größen müssen bei der Diagnose einbezogen werden
3. Schaltungen, die dies ermöglichen müssen relativ "einfach" sein
  - a. aus Kostengründen
  - b. um deren Einsatzhäufigkeit zu erhöhen
4. Messwerte die dem System bereitgestellt werden müssen in digitaler Form vorliegen
5. Unterstützende "Hardware" um die Messungen vornehmen zu können soll minimal sein

#### **VIII.1.1. Konzepte für interne Messungen analoger Größen**

Zur Messung interner Größen wurden verschiedene Konzepte erstellt. Diese Konzepte haben alle die Eigenschaft die Messdaten in digitaler Form zur Verfügung zu stellen.

##### **VIII.1.1.1. Oszillatorkalibrierung und Onlinemessung**

Durch Ausnutzen der Hochfrequenzeigenschaften heutiger Technologien, können Oszillatoren so gebaut werden, dass deren Kalibrierung wesentlich vereinfacht wird.

Oszillatoren, die im VHF-Bereich schwingen, werden nicht mehr anhand von Stellspannungen kalibriert sondern schwingen frei. Die benötigten Niederfrequenzschwingungen erhält man durch einen digitalen Frequenzteiler (Zähler).

Dies erlaubt es, die Benötigte Frequenz im Einsatz zu beobachten und gegebenenfalls durch Änderung des Teilverhältnisses zu korrigieren. Dies geschieht, indem zu gegebenen, vom System abhängigen, Zeitpunkten eine Referenzperiode des erwarteten Taktes an den Baustein angelegt wird.



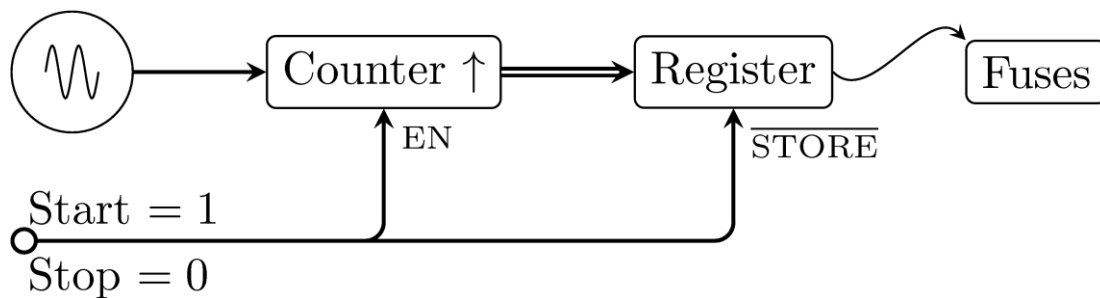


Abbildung VIII-1: Oszillatorkalibrierung

Abbildung VIII-1 zeigt einen Hochfrequenzoszillator in der "Kalibrierkonfiguration". Während dieser Phase wird vom System das Startsignal auf "1" gesetzt. Der Zähler (Counter) wird bei anlegen der "1" aktiviert und zählt die Perioden des Hochfrequenzoszillators solange die "1" anliegt. Wählt man die Zeit während der die "1" anliegt so, dass sie einer halben Periode der gewünschten Periodendauer entspricht, so enthält der Zähler die Anzahl der Takte des Hochfrequenzoszillators während dieser Zeit. Beim Übergang des Startsignals von "1" nach "0" wird der Wert des Zählers in "Register" gespeichert.

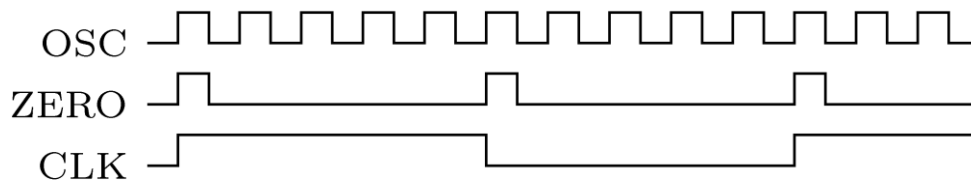


Abbildung VIII-2: Signalübersicht bei der Kalibrierung

Abbildung VIII-2 zeigt eine Übersicht der Signale, die während der Kalibrierungsphase anliegen. "OSC" ist das Hochfrequenzsignal des Oszillators, "CLK" ist der gewünschte Takt, "Zero" wird zur Initialisierung des Zählers benutzt. Sobald "CLK" den Wert "0" hat, wird der Wert vom Zähler in das Register gespeichert, bei Bedarf kann der Wert in Fuses gespeichert werden als Initialkalibrierungswert.

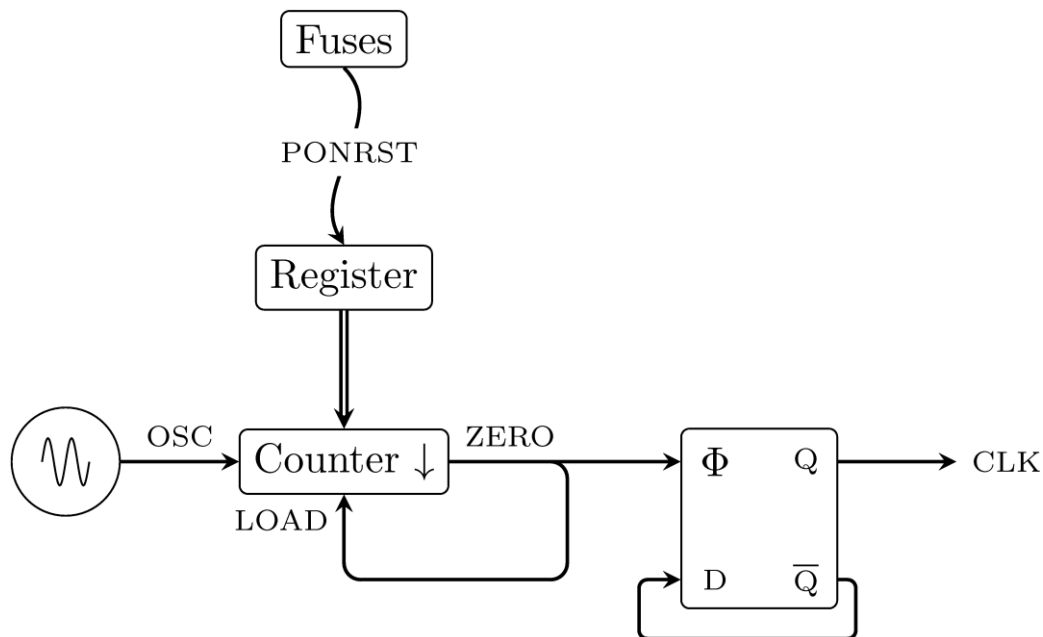


Abbildung VIII-3: Clackerzeugung im Betrieb

Abbildung VIII-3 zeigt die Konfiguration der Clackerzeugung im Betriebsfall. Der Wert aus dem Register wird in den Zähler geladen, dieser zählt nun nach unten und erzeugt einen Puls beim Erreichen der Null, die so erzeugten Pulse werden vom Flip-Flop dann in einen Taktsignal mit der gewünschten Periode gewandelt.

Der Vorteil dieses Clackerzeugungskonzeptes liegt darin dass man:

6. Keine Spannungseinstellung des Oszillators braucht
7. Im Betrieb die Frequenz ändern kann
8. Alterungseffekten entgegenwirken kann, da man die erzeugte Taktfrequenz im Betrieb "einlernen" kann
9. Nach einer Kalibrierung den Wert des Registers auslesen kann und gegebenenfalls ein Alarm auslösen kann.

#### **VIII.1.1.2. Interne Strommessung**

Interne Referenzströme oder Ströme zur Einstellung der Arbeitspunkte analoger Schaltungen müssen oftmals gemessen werden. Vor allem Arbeitspunktströme können einen Aufschluss geben über die Funktionsfähigkeit der analogen Schaltungen. Ihre Messung kann daher dazu dienen einen Alarm auszulösen wenn Abweichungen von den spezifizierten Werten festgestellt werden. Intern generierte Referenzströme werden gemessen um sicherzustellen, dass sie einen spezifizierten Wert aufweisen.

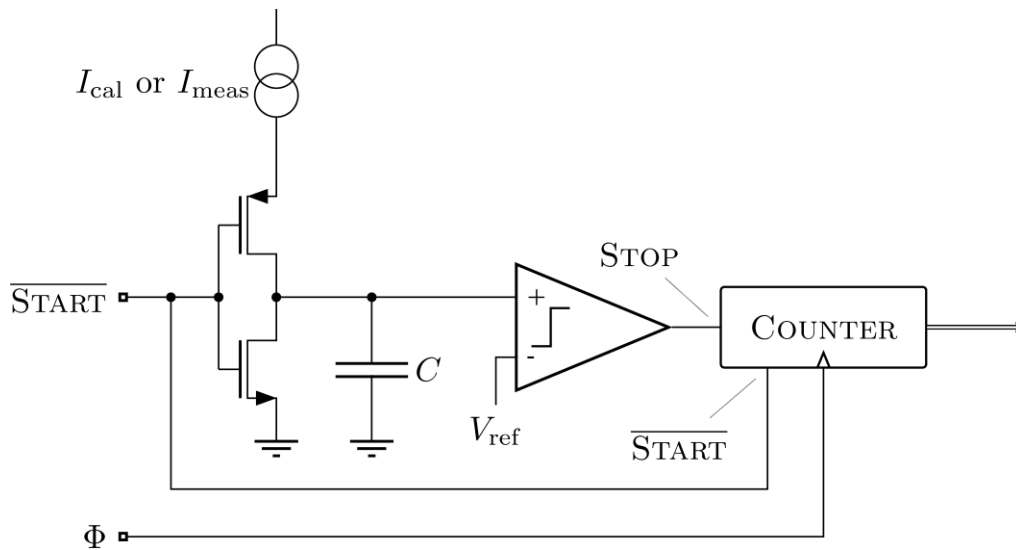


Abbildung VIII-4: Schaltung zur internen Strommessung

Abbildung VIII-4 zeigt eine Schaltung, die es erlaubt mit geringem Schaltungsaufwand einen Strom intern im Baustein zu messen.

Diese Schaltung ist eine Variante des Zweirampenverfahrens, das bei Analogdigitalwandlern zur Anwendung kommt.

Während einer Kalibrierphase, die entweder beim Produktionstest oder im laufenden Betrieb stattfinden kann, wird ein bekannter Strom mit der Schaltung gemessen um eine Referenzmessung zu erhalten.

Die Messung läuft wie folgt ab:

1. Der zu messende Strom  $I_{cal}$  oder  $I_{meas}$  wird angelegt
2. Das Signal "Start" wird auf "1" gesetzt, die Kapazität  $C$  wird entladen
3. Das Signal "Start" wird auf "0" gesetzt, dies hat zwei Effekte
  - a. Der Zähler (Counter) wird gestartet, und zählt nun die Takte des Referenztaktes
  - b. Der Strom lädt die Kapazität  $C$  auf
4. Die Spannung an der Kapazität steigt bis sie die Komparatorschwelle  $V_{ref}$  erreicht hat
5. Der Komparator kippt bei Erreichen von  $V_{ref}$  und stoppt den Zähler

Der Zähler enthält nun einen Wert, der proportional zum angelegten Strom ist:

$$V_{\text{ref}} + V_{\text{off}} = \frac{n T I}{C}$$

Gleichung 1

Wobei  $V_{\text{ref}}$  die Komparatorschwellschwellspannung,  $V_{\text{off}}$  der Offset des Komparators,  $n$  der Zählerwert,  $T$  die Periodendauer des angelegten Takts,  $I$  der angelegte Strom und  $C$  der Kapazitätswert sind.

Gleichung 1 gilt sowohl für die Referenzmessung mit dem Bekannten Strom als auch für eine Messung mit unbekanntem Strom. Wenn  $I_{\text{cal}}$  der Referenzstrom ist und  $I_{\text{meas}}$  der zu messende Strom ist so gilt:

$$V_{\text{ref}} + V_{\text{off}} = \frac{n_{\text{cal}} T I_{\text{cal}}}{C}$$

Und:

$$V_{\text{ref}} + V_{\text{off}} = \frac{n_{\text{meas}} T I_{\text{meas}}}{C}$$

Durch Gleichsetzen beider Ausdrücke erhält man den Wert des gemessenen Stromes:

$$I_{\text{meas}} = \frac{n_{\text{cal}}}{n_{\text{meas}}} I_{\text{cal}}$$

Gleichung 2

Gleichung 2 zeigt wie man den Stromwert berechnet, im System würde das Auslesen des Zählerwertes ausreichen um zu erfahren ob der gemessene Strom im erlaubten Bereich. Bei Abweichungen kann hier Alarm ausgelöst werden. Die Sollwerte des Zählers können entweder Werksseitig oder bei der Initialisierung des Systems in den Baustein eingespeichert werden.

### **VIII.1.1.3. Interne Spannungsmessung**

Wie bei den Strömen, besteht Bedarf an Spannungsmessungen. Diese Messungen sollten ebenso Werte in Digitalform ergeben. Ebenso sollten die hierfür nötigen Schaltungen einen geringen Flächenbedarf aufweisen.

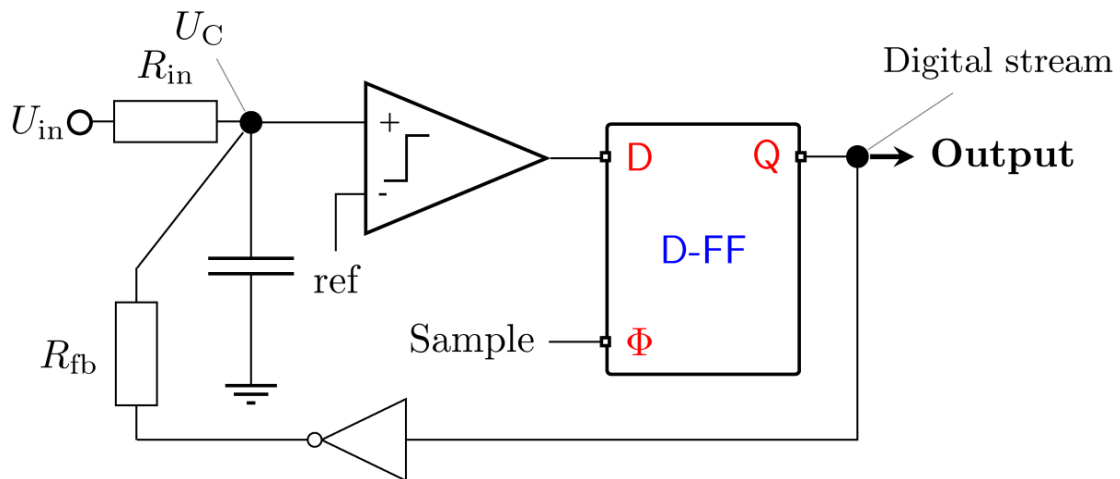
Abbildung VIII-5:  $\Sigma\Delta$  AD-Umsetzer

Abbildung VIII-5 zeigt einen  $\Sigma\Delta$  AD-Umsetzer [1]. Die Struktur eines solchen Umsetzers ist besonders einfach und braucht somit eine relativ geringe Fläche für die Implementierung. Begünstigt wird der Einsatz eines solchen Umsetzers dadurch, dass die heutigen Technologien es erlauben höherfrequente Schaltungen zu implementieren.  $\Sigma\Delta$  AD-Umsetzer erlauben es Genauigkeit durch Geschwindigkeit zu erreichen, und sind somit die Lösung der Wahl bei den heutigen Technologien.

Um einen Einblick in die Funktionsweise des ADC zu gewinnen, müssen einige Annahmen gemacht werden:

1.  $U_{ref} = V_{dd}/2$  ( $U_{ref}$ : Komparatorschwellspannung,  $V_{dd}$ : Versorgungsspannung)..
2. Taktfrequenz hoch:  $RC \gg T$
3. Beide Widerstände sind gleich

Mit diesen Annahmen kann man analysieren wie sich die Spannung der Kapazität zeitlich verhält.

Während eine Clockperiode gilt allgemein:

$$\Delta V = I \frac{T}{C} = (U - U_C) \frac{T}{RC}$$

Gleichung 3

Der Rückkopplungsinverter kann nur den Wert "1" oder "0" an seinem Ausgang aufweisen, die ergibt für jeden Fall:

- "1" am Invertiereingang:
 
$$\Delta V_1 = (U_1 - U_C) \frac{T}{RC}$$
- "0" am Invertiereingang:
 
$$\Delta V_0 = (U_0 - U_C) \frac{T}{RC}$$

Die Eingangsspannung wirkt auf die Kapazität auch während der Clockperiode, und dies unabhängig vom Ausgangswert des Inverters, dies ergibt:

$$\Delta V_{in} = (U_{in} - U_C) \frac{T}{RC}$$

Nach Einsatz der tatsächlichen Spannungen, und unter Berücksichtigung der Tatsache, dass die Rückkopplung im System die Spannung der Kapazität auf einen Mittelwert von  $U_{ref} = V_{dd}/2 = U/2$  hält:

- "1" am Invertereingang:  

$$\Delta V_1 = (0 - U_C/2) \frac{T}{RC}$$
- "0" am Invertereingang:  

$$\Delta V_0 = (U - U/2) \frac{T}{RC}$$
- Effekt der Eingangsspannung:  

$$\Delta V_{in} = (U_{in} - U/2) \frac{T}{RC}$$

Unter der Annahme, dass die Schaltung längere Zeit läuft und den eingeschwungenen Zustand erreicht hat, kann man Aussagen darüber treffen was "im Mittel" geschieht:

$$-\frac{1}{2}n_1U\frac{T}{RC} + \frac{1}{2}n_0U\frac{T}{RC} + n_sU_{in}\frac{T}{RC} - \frac{1}{2}n_sU\frac{T}{RC} = 0$$

Wobei  $n_1$  und  $n_0$  die respektive Anzahl der "1" bzw. der "0" am Eingang des Inverters sind.  $n_s$  stellt die Summe der Takte während des Beobachtungszeitraums dar. Damit gilt dass  $n_1 + n_0 = n_s$

Anders ausgedrückt: Die Auswirkungen der Rückkopplung und die der Eingangsspannung heben sich auf, da im Mittel die Kapazitätsspannung den konstanten Wert von  $U/2$  aufweist.

Umgeformt, ergibt dies:

$$n_sU_{in} = \frac{U}{2} (n_1 - n_0 + n_s) = n_1U \Rightarrow U_{in} = \frac{n_1}{n_s}U$$

Gleichung 4

Gleichung 4 besagt, dass die Anzahl der Outputbits mit dem Wert "1" im beobachteten Zeitraum proportional der Eingangsspannung ist.

Ein Zähler am Ausgang des ADC, der die "1" Bits zählt, würde also am Ende einer Wandlung eine Zahl enthalten, die proportional dem Eingangsspannungswert ist. Dieser Wert ist der gewandelte Wert der Eingangsspannung.

### VIII.1.2. Messungen an komplexen Schaltungen

Messungen einzelner Werte innerhalb eines Bausteins sind ein wichtiger Schritt in Richtung Selbstdiagnose. Heutige Bausteine wie z.B. Microcontroller enthalten aber komplexere Schaltungen oder Systeme, z.B. Analogdigitalumsetzer, Digitalanalogumsetzer oder Phasenregelschleifen.

Würden Messungen an diesen Systemen mithilfe der vorigen Schaltungen durchgeführt, so würde deren Test bzw. Diagnose zu lange dauern um im System (im laufenden Betrieb, im Fahrzeug) effektiv eingesetzt zu werden.

Bei komplexeren Baugruppen innerhalb eines Bausteins müssen also dedizierte Lösungen zur Selbstdiagnose entwickelt werden.

Anhand des Beispiels eines Analogdigitalkonverters wurden hier Diagnosekonzepte entwickelt. Analogdigitalkonverter wurden in diesem Falle ausgewählt weil sie in fast jedem Microcontroller zu finden sind. Zudem gestaltet sich ihre Messung, aufgrund der hiermit verbundenen Genauigkeitsanforderungen, als zeitaufwendig.

Da diese Schaltungen zur Erfassung von u.U. wichtigen Signalen benutzt werden, ist deren Funktionalität ein wichtiger Aspekt für die Sicherheit im Fahrzeug.

#### VIII.1.2.1. Test von Mehrfach-ADC

Befinden sich wie in Abbildung VIII-6 mehrere ADC in einem Baustein, so müsste man, um eine Diagnose durchzuführen, alle auf ihre Funktion hin überprüfen.

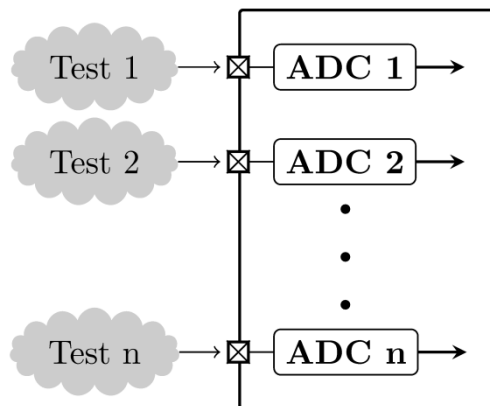


Abbildung VIII-6: Test mehrerer ADC

Dies kann entweder sequentiell geschehen indem man Stimuli an den ADC-Eingängen anlegt und ihre Kennlinie misst. Alternativ können mehrere Stimuligeneratoren an den ADC-Eingängen angelegt werden um die Messungen parallel durchzuführen.

Die erste Möglichkeit scheitert daran, dass ein Stimuligenerator im System vorhanden sein muss. Da der Generator eine hochpräzise Schaltung sein muss, stellt dies ein nicht unerheblicher Kostenfaktor für die Systemhersteller dar. Dazu kommt, dass die Messungen sequentiell stattfinden was den Zeitaufwand für eine Baustein Diagnose verlängert. Ein nicht zu vernachlässigendes Problem stellt auch die Signalintegrität der Stimuli dar, diese müssen ungestört vom Generator bis zum

Baustein geführt werden, was zur Komplexität der Platine führt und somit wiederum zu einer Erhöhung der Kosten führt.

Die zweite Möglichkeit fällt noch negativer in ihren Auswirkungen aus, denn um die Messzeit zu reduzieren, werden nun *mehrere* Generatoren notwendig. Für die gelten wiederum alle zuvor genannten Nachteile, allerdings nun mehrfach.

Es ist also naheliegend die gesamte Messapparatur im Baustein unterzubringen damit der Selbsttest möglichst ohne Störungen, ohne Zusatzbeschaltung und in einer annehmbaren Zeit stattfinden kann.

Eine Möglichkeit den Test ohne Zusatzhardware durchzuführen bestünde darin *einen* der im Baustein befindlichen ADC zu testen, diesen als DAC zu betreiben und mit diesem DAC die notwendigen Spannungen zum Test der restliche ADC heranzuziehen. Dies wird schematisch in Abbildung VIII-7 dargestellt.

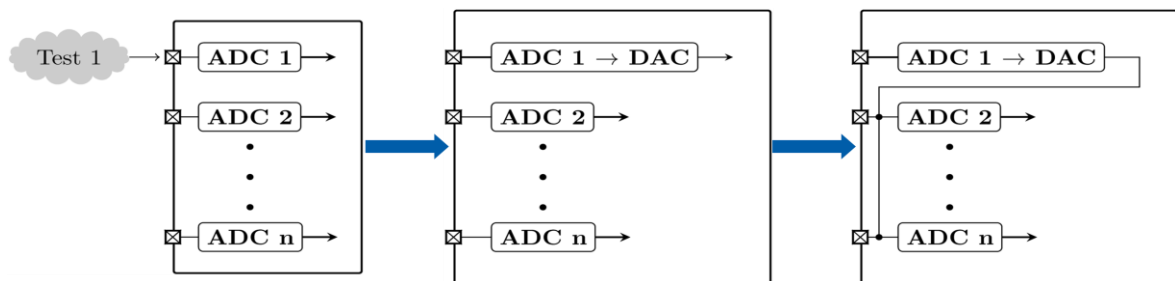


Abbildung VIII-7: Test mit minimaler Hardwareunterstützung

Nun ist das Problem der Messung darauf reduziert worden, dass man einen Stimuligenerator braucht. Für diesen gelten aber die zuvor genannten Nachteile. Wenn aber ADC1 ein 1-Bit  $\Sigma\Delta$  AD-Umsetzer ist, dann reduziert sich die Aufgabe des Stimuligenerators auf die Bereitstellung von zwei verschiedenen Gleichspannungen. Denn diese ADC sind inhärent linear, d.h. um ihre Funktionalität zu prüfen werden nur zwei Werte gebraucht: Offset und Gain, diese können aus lediglich zwei Messungen mit unterschiedlichen Eingangsspannungen bestimmt werden.

Als ADC kann man entweder einen im Baustein vorhandenen  $\Sigma\Delta$  ADC heranziehen oder den zuvor in VIII.1.1.3 beschriebenen ADC für diese Messaufgabe im Baustein implementieren.



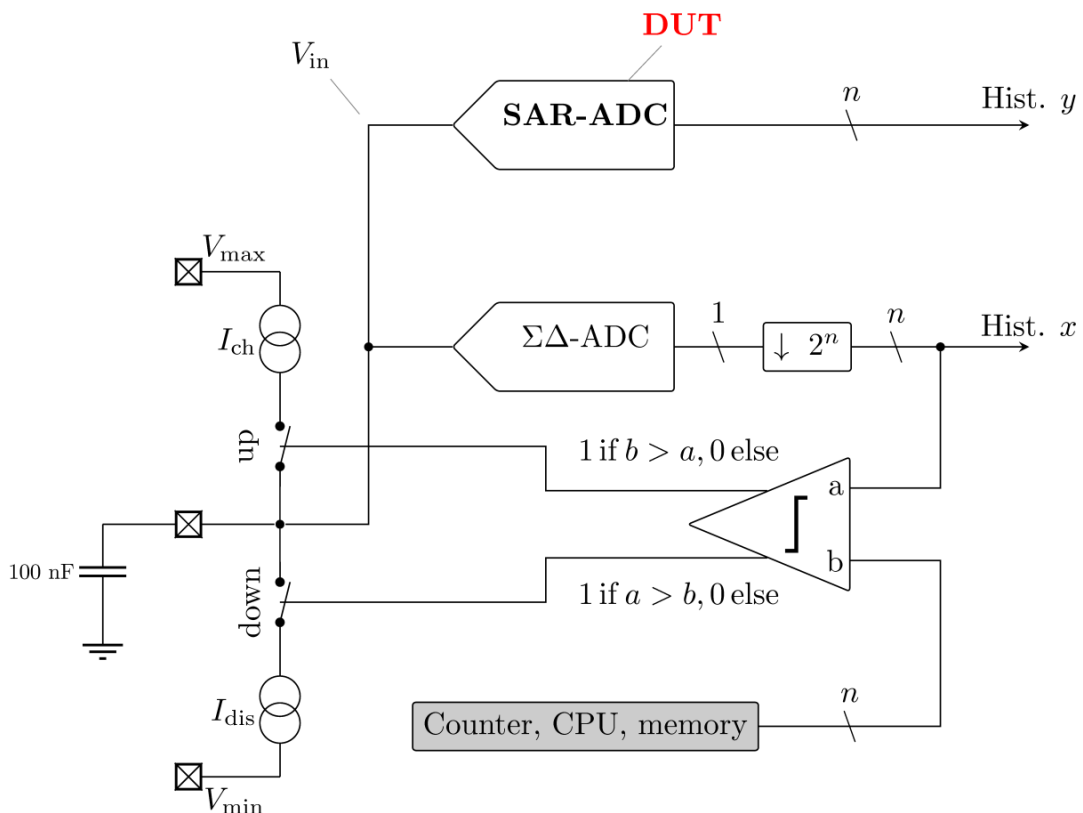


Abbildung VIII-8: Implementierung des internen ADC Tests, schematische Darstellung

Abbildung VIII-8 zeigt das Systemschaltbild der vorgestellten Lösung. Der  $\Sigma\Delta$  AD-Umsetzer wird, nach Prüfung seiner Funktion und der Bestimmung seines Offsets und Gains, zu einem DAC umgewandelt. Dies geschieht indem man den Ausgang des ADC zu einem digitalen Komparator führt (Eingang  $a$ ), dieser Komparator wird solange Strom aus den beiden Stromquellen in die Kapazität fließen lassen, bis der Wert des Wortes am Ausgang des ADC dem Wert gleicht, der am Eingang  $b$  angelegt wurde.

Die Spannung  $V_{in}$ , die sich am Eingang des rückgekoppelten ADC einstellt, wird abgegriffen und dem Eingang des zu testenden ADC (Als DUT bezeichnet) zugeführt. Ein Vorteil dieser Methode ist, dass nunmehr, abgesehen von den beiden Spannungswerten, mit denen Offset und Gain bestimmt werden, keine analogen Spannungen vom System bereitgestellt werden müssen. Alle Eingangsgrößen sowie alle Ausgangsgrößen liegen in digitaler Form vor und können somit vom System abgerufen und ausgewertet werden. Alle Messungen der einzelnen Parameter eines ADC, siehe [2], können nur vom Baustein selbst durchgeführt werden.

### VIII.1.2.2. Exponentialrampen ADC-Test

Bei einzelnen ADC oder bei Bausteinen, die keinen  $\Sigma\Delta$  ADC enthalten, muss auch eine Lösung angeboten werden.

Nicht wenige Microcontroller enthalten nur einen ADC. Hinzu kommt, dass die Implementierung eines  $\Sigma\Delta$  ADC in machen Technologien nicht möglich ist, sei es

aufgrund der maximal möglichen Taktfrequenz oder weil der Flächenbedarf eines solchen ADC zu hoch ist.

Eine Möglichkeit trotzdem einen Selbsttest durchzuführen besteht darin den ADC mit der Lade- bzw. Entladekurve eines RC-Gliedes zu stimulieren.

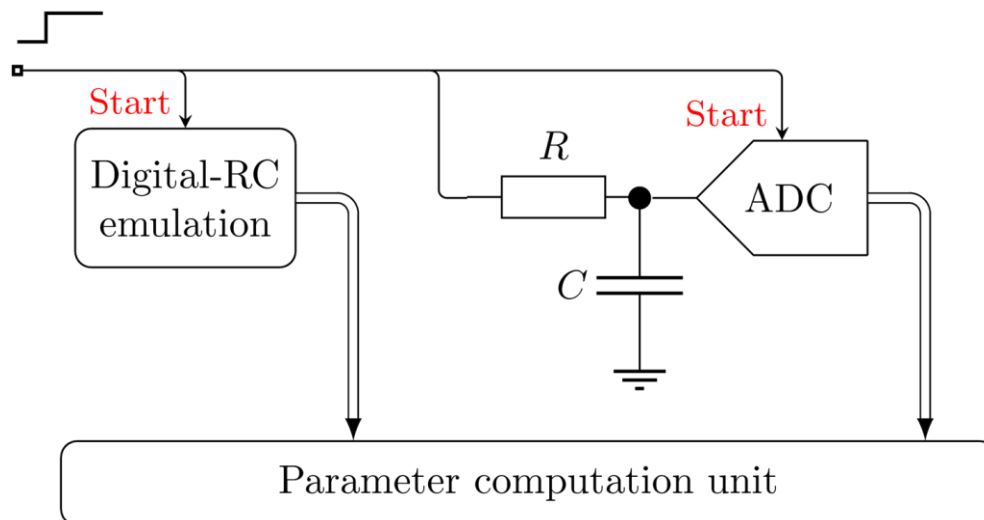


Abbildung VIII-9: Exponentialrampen-ADC-Test, schematische Darstellung

Abbildung VIII-9 zeigt ein Blockschaltbild des Verfahrens. Ein RC-Glied (Testsignalgenerator) wird geladen bzw. entladen.

Während einer Initialisierungsphase wird die Zeitkonstante des RC-Glieds bestimmt. Mit Kenntnis der Zeitkonstante kann ein digitaler Tiefpass das RC-Glied emulieren und erzeugt dann digitale Werte, die den Ausgangswerten eines idealen ADC entsprechen würden. Dadurch, dass jetzt die idealen Werte und die Werte, die der zu testende ADC liefert gleichzeitig bekannt sind, kann man aus dem Vergleich dieser Werte Parameter des zu testenden ADC bestimmen.

Das Konzept wurde in Hardware implementiert um realitätsnahe Ergebnisse zu erhalten. Zu den Einzelheiten des Verfahrens und den Herleitungen für einzelne Größen siehe [3], im Meilensteinbericht M2.1.2 werden die hier benutzten Zusammenhänge hergeleitet und theoretisch untermauert.

Die "Parameter computation unit", die digitale RC-Emulation sowie die Ablaufsteuerung wurden auf einem Evaluationboard des TC1797 (Infineon Technologies Microcontroller) implementiert. Der Microcontroller enthält auch den zu testenden ADC (FADC, Fast ADC).



Abbildung VIII-10: Microcontroller Evaluationboard (TC1797)

Abbildung VIII-10 zeigt den für den Prototypenaufbau benutzten Microcontroller Evaluationboard.

Benutzt wurden beim Prototypenaufbau: der FADC als DUT, die DMA-Fähigkeiten des auf dem Board befindlichen Microcontrollers (AUDO) um die ADC Daten in das 128 kB große RAM zu transferieren, die serielle Kommunikationsschnittstelle um die Daten aus dem Evaluationboard auszulesen und den Ablauf zu steuern sowie die GPTA (General Purpose IO) um eine Platine mit den RC-Stimuligenerator anzusteuern: TSG (Test Signal Generator).

Der Testsignalgenerator wurde diskret aufgebaut. Abbildung VIII-11 zeigt den Aufbau des Testsignalgenerators, in Abbildung VIII-9 ist nur ein RC-Glied dargestellt. Die zusätzlichen Multiplexer und der Verstärker dienen der Verringerung der Messzeit, sowie der Entkopplung der Referenzspannung.

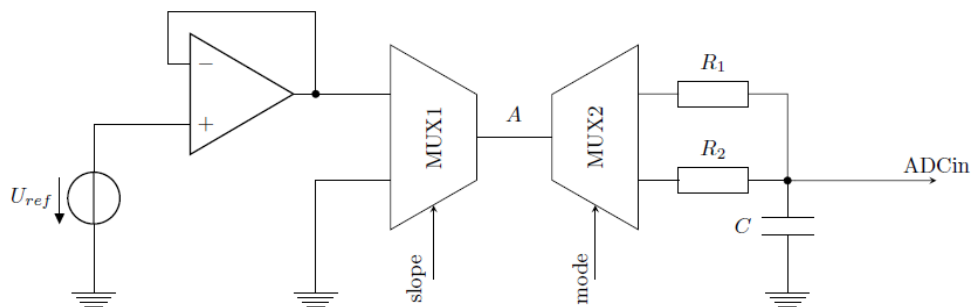


Abbildung VIII-11: Aufbau des Testsignalgenerators

Beim ADC-BIST wird erst eine Ladekurve erzeugt, die bis zur Hälfte der Referenzspannung reicht, was  $0:7 T$  in Anspruch nimmt. Um die obere Region des Eingangsspannungsbereiches des ADC messen zu können, wird eine Entladekurve herangezogen. Diese fängt bei der Referenzspannung an und hört bei Erreichen der halben Referenzspannung auf. Da dies wiederum auch nur  $0:7 T$  dauert, reduziert sich die Messzeit auf  $1:4 T$  was wesentlich kürzer ist als die  $7 T$  die zu warten wären bis die Spannungen die Genauigkeiten erreichen, die es es erlauben würden einen 10 Bit ADC zu messen. Angesteuert wurden die Schalter über die zur Verfügung stehenden General Purpose IO-Pins des Triboards.

Abbildung VIII-12 zeigt schematisch den TSG zusammen mit dem Evaluationboard

sowie die Hauptsignalfade zwischen diesen beiden Schaltungen.

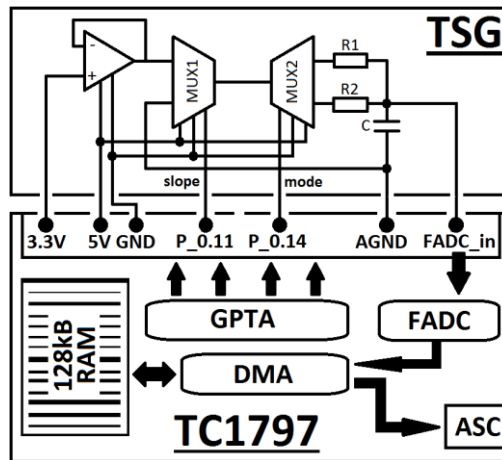


Abbildung VIII-12 Evaluationboard und Testsignalgenerator, Gesamtschaltung

Mit diesem Messaufbau wurde der sich im Microcontroller befindliche ADC getestet. Man sieht in Abbildung VIII-13, dass die Schaltung funktionsfähig ist. Beide Kurven, die fallende und die steigende Flanke treffen sich am Zeitpunkt  $\ln(2) \tau$ .

Damit sind die Eigenschaften des RC-Gliedes bekannt. Der digitale Tiefpaß zur RC-Emulation wird entsprechend mit diesem Wert programmiert.

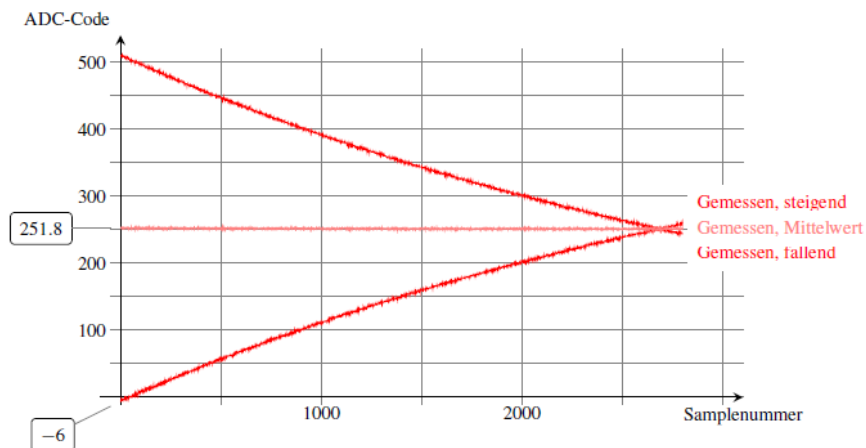


Abbildung VIII-13: Gemessene Daten

Abbildung VIII-14 zeigt eine Vergrößerung des Bereiches in dem sich beide Kurven kreuzen.

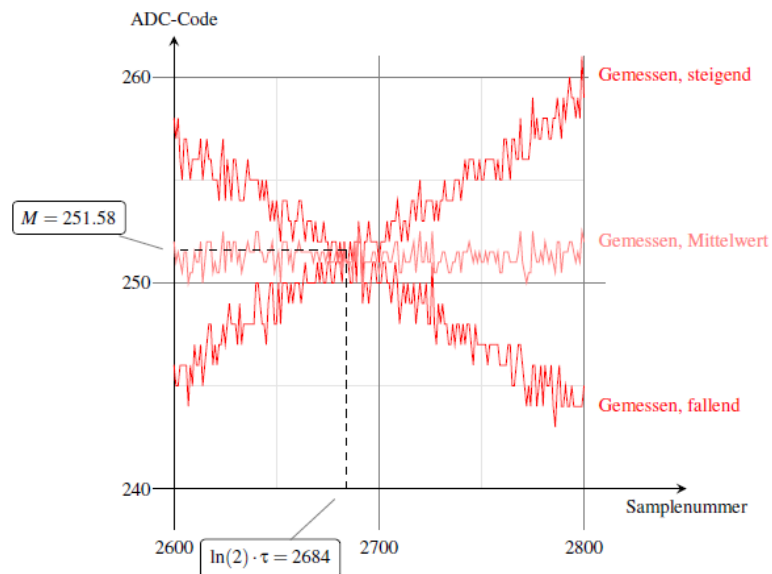


Abbildung VIII-14 Vergrößerung des Kreuzungsbereichs

Mit den Messwerten des ADC und denen des Referenzsignalgenerators können nun Offset und Gain des ADC bestimmt werden.

Die berechneten Offset- und Gainwerte werden in der folgenden Tabelle aufgelistet:

Offset	-5:3695
Gain	1.0057
$\epsilon_{\text{err}}$	5.7e-3

Mit diesen Werten können die ADC Codes korrigiert werden, in Abbildung VIII-15 werden die ADC-Daten mit denen des Referenzsignalgenerators (RSG) verglichen. Der Effekt der Korrektur ist deutlich, links sieht man die aufgenommenen ADC Codes ohne Korrektur der Gain- und Offset Fehler. Rechts sieht man, dass sich die Kurven, RSG und korrigierte ADC-Daten, überdecken

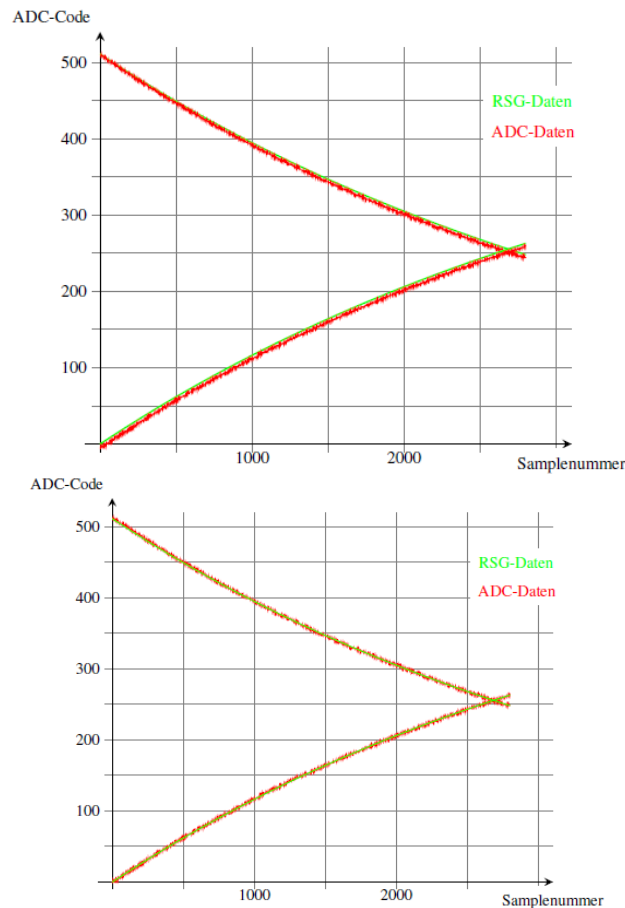


Abbildung VIII-15: ADC Codes vor und nach Korrektur der Offset- und Gainfehler  
Abbildung VIII-16 zeigt eine Vergrößerung des Bereiches um den Kreuzungspunkt der Kurven, man sieht deutlich wie sich die Kurven überdecken.

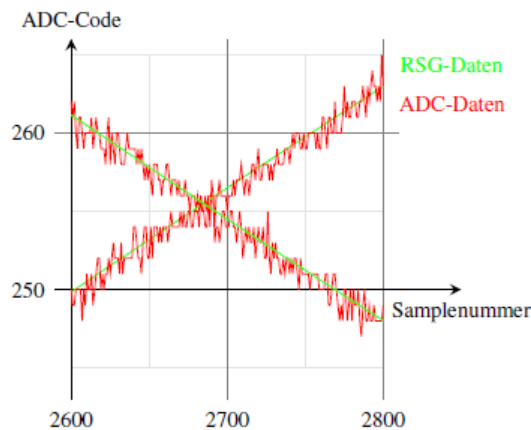


Abbildung VIII-16: Kreuzungspunkt nach Korrektur von Gain- und Offsetfehler

### **VIII.1.3. Zusammenfassung für die Mixed-Signal Selbstdiagnose**

In Aufgabe 2.1 wurden Konzepte und Schaltungen entworfen, die es erlauben Mixed-Signal sowie analoge Schaltungen mit Selbstdiagnosefähigkeiten auszustatten.

Um die Komplexität der Systeme nicht zu erhöhen, wurde besonders Wert darauf gelegt, dass die gelieferten Diagnosewerte digital vorliegen, dies erlaubt die Nutzung vorhandener Busse zum Auslesen der Diagnosewerte.

Ebenso wurde darauf geachtet, dass die Diagnoseschaltungen möglichst ohne Unterstützungshardware funktionsfähig sind. Sollten Referenzwerte benötigt werden, so wurde darauf geachtet, dass nur Gleichstrom oder -spannungswerte benötigt werden.

Schaltungen zum Messen einzelner Spannungen, Ströme oder Frequenzen wurden entworfen. Es wurde darauf geachtet, dass diese Messschaltungen möglichst wenig Flächenbedarf im Baustein beanspruchen. Um die Breite des Einsatzes zu gewährleisten wurde darauf geachtet, dass die Messsysteme möglichst autark funktionsfähig sind.

Da die Selbstdiagnose komplexerer Systeme mit Einzelmessungen unzulänglich ist, wurden, am Beispiel von ADC-Strukturen, Konzepte zur Selbstdiagnose dieser Systeme entworfen.

Mit diesen Messstrukturen wird es möglich Bausteine, die analoge oder Mixed-Signal Schaltungen enthalten, im System auf ihre Funktionsfähigkeit hin zu untersuchen. Bei Abweichungen von den erwarteten Werten können Alarme ausgelöst werden und die den Alarm auslösenden Bausteine bzw. Strukturen im Baustein lokalisiert werden.

### **VIII.1.4. Literatur**

- [1] Steven R. Norsworthy, Richard Schreier, Gabor C. Temes, "Delta-Sigma Data Converters", IEEE Press ISBN 0-7803-1045-4
- [2] IEEE Std 1658™-2011, "IEEE Standard for Terminology and Test Methods of Digital-to-Analog Converter Devices", ISBN 978-0-7381-7147-0
- [3] Diana-Meilensteinbericht, Arbeitspaket 2.1, M2.1.2

## **VIII.2. Ergebnisse zur Aufgabe 2.2(Teil1): Online Speicher-Selbsttest und Selbstreparatur in der Applikation für RAM-Speicher**

### **VIII.2.1. Ziele**

Memory-Selbsttest ist heute ein allgemein anerkanntes Verfahren, um die Fehlerfreiheit von integrierten Speichern in SoCs zu überprüfen. Allerdings werden diese Tests in der Regel nur vor der Auslieferung der Bausteine an die Kunden durchgeführt, so dass altersbedingt auftretende Fehler nicht erkannt werden. Die Logik, die zum Test von Speichern auf dem Baustein implementiert ist, sollte also so modifiziert werden, dass diese Selbsttests auch später im Feld verwendet werden können, um die fehlerfreie Funktionalität der Bausteine zu gewährleisten.

Das gilt ähnlich auch für die Selbstreparatur von integrierten Speichern. Dieses Verfahren zur Erhöhung der Ausbeute ist während des Produktionstests ebenfalls weit verbreitet und wird auch in zahlreichen Konferenzbeiträgen (siehe z.B. [KIAM] und [OEHL]) behandelt. Auch im Infineon eigenen und in diversen Bausteinen bereits verwendeten „MBISTPLUS“ IP könnte eine solche Selbstreparaturlogik leicht mit eingebaut werden. Allerdings wird auch diese Methode zur Qualitätserhöhung von Halbleiterbausteinen während der Betriebsphase noch nicht effizient verwendet.

Zunächst sollte eine erweiterte Recherche durchgeführt werden, um den aktuellen Stand der Technik und die zurzeit auf dem Markt vorhandenen Methoden auf ihre Eignung für die vorgesehene Verwendung bewerten zu können. Die Ergebnisse dieser Recherchen werden in einem 2. Schritt mit den eigenen Ideen verglichen und die bestmögliche Variante wird dann so angepasst, dass sie problemlos in das Infineon eigene IP zum Memory-Selbsttest eingebaut werden kann. Randbedingungen sind der benötigte Flächenbedarf und die weitgehende Unabhängigkeit des Selbsttestansatzes von der zu testenden Schaltung sowie die Sicherstellung der höchstmöglichen Qualität der später auszuliefernden Bausteine. Auf Basis der erstellten Spezifikation wird eine logische Schaltung entwickelt, die später problemlos in neue Produkte integriert werden kann. Die logische Funktionalität der entwickelten Schaltung wird mittels geeigneter Tools verifiziert. Die einfache Integration wird mittels eines Pilotprojektes überprüft, mit dessen Musterbausteinen die Funktionalität dann auch im Labor verifiziert wird.

Hauptanwendungsgebiet ist zunächst die Automobilindustrie mit ihren hohen Anforderungen an Qualität und Zuverlässigkeit von elektronischen Schaltungen. Der erweiterte Selbsttest wird zunächst beim Systemstart durchgeführt. Das bedeutet beim Automobil in der Regel beim Starten des Motors. Je nach Relevanz des aufgetreten Fehlers kann dann nur ein Warnung ausgegeben, ein Notlaufprogramm gestartet oder dem Fahrer detaillierte Hinweise zur Relevanz des Fehlers und konkrete Verhaltensmaßnahmen gegeben werden. Diese Arbeiten bilden die Grundlage für eine Fortführung zu einem permanenten Selbsttest im Betrieb mit weiteren Systemimplikationen. Ein erstes Beispiel dazu ist der Speichertest im Betrieb.

Durch den Einbau der erweiterten Selbsttestfunktionalität können mehrere der angegebenen Ziele erreicht werden:

- Es wird möglich sein, fehlerhafte Speicherzellen auf dem Schaltkreis schneller zu identifizieren. Dadurch können Technologieschwachstellen besser identifiziert und schneller behoben werden.
- Die Verwendung der zu entwickelnden Schaltkreise ermöglicht den Einsatz von genaueren Analyse- und Diagnoseverfahren auch bei Modullieferanten, OEMs und beim Endanwender, da die Diagnosedaten während des Betriebs gesammelt werden und zur Auswertung lediglich ausgelesen werden müssen.
- Die gesammelten Daten und die dadurch mögliche exakte Lokalisierung der Fehlerquelle(n) werden sich ebenfalls positiv auf die Diagnosefähigkeit der Werkstätten auswirken, da mit ihrer Hilfe eine zielgerichtete Reparatur durchgeführt werden kann.



## VIII.2.2. Ergebnisse

### VIII.2.2.1. Einleitung

Zuverlässigkeitsprobleme mit Durchkontaktierungen und Kontakten werden mit kleiner werdenden Strukturen mehr und mehr zu einem Problem. Das Hauptproblem besteht darin, dass die Überwachung der Qualität dieser Durchkontaktierungen in der Produktion recht aufwändig ist. Zurzeit wird bei Infineon ein verbesserter „Via Testchip“, der 10E9 Durchkontaktierungen pro Wafer aufweist, verwendet. Es ist ziemlich unwahrscheinlich, dass die Anzahl dieser Strukturen weiter erhöht werden kann, weil alle diese Durchkontaktierungen mit Hilfe von Messungen auch kontrolliert werden müssen. Daher beträgt die Entdeckungswahrscheinlichkeit für dieses Problem zurzeit 10E-9. Während dieses Resultat für die eigentlichen Durchkontaktierungsprobleme (Stress- und Elektronenmigration), die gut kontrollierbar sind und für die es Beschleunigungsmodelle gibt, auszureichen scheint, sieht die Situation bei extrinsischen Problemen nicht so gut aus. Extrinsische Probleme sind Zuverlässigkeitsprobleme, die durch sich mit der Zeit auflösende Durchkontaktierungen, die von Polymer Partikeln, die bei der Herstellung der Wafer auf diesem verbleiben, verursacht werden. Diese Polymer Partikel verteilen sich gleichmäßig über den Chip und sind die Hauptursache für sich auflösende Durchkontaktierungen. Zurzeit existiert kein gültiges Beschleunigungsmodell, um diesen Effekt künstlich zu beschleunigen. Und selbst wenn man 100 Wafer mit einer vorhandenen Methode behandeln würde, so würde dies die Entdeckungswahrscheinlichkeit für dieses Problem lediglich auf 10E-11 erhöhen. Darüber hinaus müssten diese 100 Wafer permanent überwacht werden, um die Zeitabhängigkeit des Prozesses beobachten zu können. Das muss nicht bedeuten, dass der Prozess so schlecht ist, er kann sogar deutlich besser sein. Aber so lange keine Methode existiert, mit der dieser Prozess überwacht werden kann, muss von der schlimmsten Wahrscheinlichkeit ausgegangen werden, die mit angemessenem Aufwand überprüft werden kann. Diese beträgt zurzeit 10E-11.

Bedenkt man, dass ein aktuelles Design für einen Microcontroller Chip in der Automobilindustrie mehr als 10 Millionen einzelne Durchkontaktierungen besitzt, so führt dies zu einer theoretischen und berechneten dppm Rate über die Lebenszeit von 100 und größer. Viele dieser Durchkontaktierungen befinden sich in SRAMs und dort selbst in der eigentlichen Zellmatrix, in der man die Anzahl der Durchkontaktierungen nicht einfach verdoppeln kann weil sich die Fläche des Speichers dadurch ebenfalls ungefähr verdoppeln würde. Genauso ergeben sich negative Auswirkungen für den Energieverbrauch und die Zugriffszeiten.

Für Kontakte, deren Anzahl im Speicher sich auf ungefähr 17 Millionen beläuft, gelten die gleichen Überlegungen (auch hier liegt die Mehrzahl der Kontakte innerhalb der eigentlichen Zellmatrix).

Da insbesondere die Automobilindustrie nach immer besseren dppm Raten („Automotive Excellence Program“) verlangt, sind die genannten Ausfallraten allein für eines der genannten Probleme inakzeptabel. Die momentanen Anforderungen zielen auf Werte kleiner 1 dppm für die gesamte Lebenszeit des Chips.

Es ist ganz offensichtlich, dass verschiedene Lösungsansätze verfolgt werden müssen, um die genannten Probleme zu überwinden. Als erstes muss die Technologie verbessert werden, um zu verhindern, dass diese Zuverlässigkeitsprobleme überhaupt erst auftreten. Zweitens muss das Testen bzw. Überwachen der Bausteine verbessert werden, um die Entdeckungswahrscheinlichkeit für diese Schwächen weiter zu erhöhen. Und drittens müssen die Schaltungen so entworfen werden, dass sie unempfindlicher auf die Effekte der genannten Probleme reagieren.

Im Rahmen des DIANA Projektes sollte zunächst eine dritte Möglichkeit, nämlich eine Designmethodik, genannt „SMART Repair“, daraufhin untersucht werden, ob mit ihr die genannten Auswirkungen des Zuverlässigkeitsproblems entschärft werden können. Dieser Ansatz wurde notwendig, da die konventionelle „Brute-Force“ Methode mittels ECC Überwachung des Speicherinhalts mit ihren Hauptnachteilen wie zusätzlicher Flächen- und Energieverbrauch sowie schlechteres Zugriffszeitverhalten nicht akzeptabel erschien. In den vorgesehenen Anwendungen beträgt die Breite des Datenbusses bei Schreibzugriffen für die meisten verwendeten Speicher lediglich 16 Bit so dass für jeweils 16 Datenbits 6 zusätzliche Kontrollbits vorgesehen werden müssen. Diese schmalen Schreibzugriffe können aus Gründen des Datendurchsatzes nicht auf breitere „read modify write“ Zugriffe abgebildet werden. Da sich die Anzahl der Kontrollbits wie folgt berechnet

Anzahl der Kontrollbits =  $\text{Id}(\text{Datenbreite}) + 2$  Bits für SECDED ECC Kodierung

bewirken schmale Schreibzugriffe auf den Speicher mit ECC relativ höhere Flächen als dies bei Speichern mit breiten Datenbussen der Fall ist.

### **VIII.2.2.2. SMART Repair**

#### *Voraussetzung*

Die Voraussetzung zur Anwendung der SMART Repair Funktion ist das Vorhandensein des programmierbaren MBISTPLUS IPs. MBISTPLUS (siehe auch Abbildung VIII-17) ist eine programmierbare BIST Lösung mit folgenden Eigenschaften:

- Einstellung des Adressbereiches des zu testenden Speichers
- Einstellung von entweder anwenderspezifischen oder vorgegebenen Testalgorithmen mit denen der Speicher getestet werden soll
- Verwendung verschiedener Testmuster
- Redundanzregister, um defekte Speicherworte durch Flipflops zu ersetzen

Die zu reparierenden Bereiche werden vom MBISTPLUS IP während des Tests der Speicher automatisch ermittelt. Die Information, welche Speicherzellen defekt sind, wird nach dem Test in einem nichtflüchtigen Speicher (Flash), abgelegt. Beim Hochfahren des Bausteins wird diese Information aus dem Flash Speicher ausgelesen und in den Redundanzblock zurückgeschrieben.

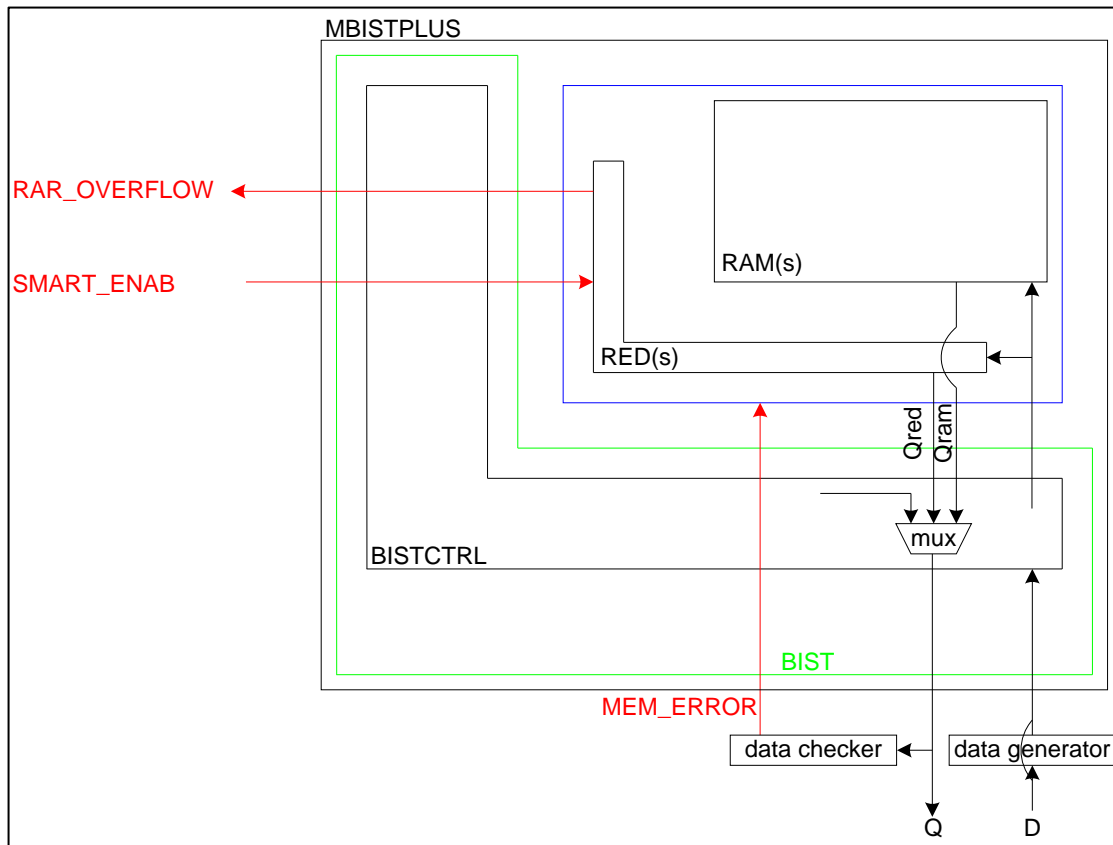


Abbildung VIII-17: Blockschaltbild MBISTPLUS IP

Es gilt zu beachten, dass alle Speicher mit einem Parity Schutz zur Erkennung von latenten Einbit Fehlern, ausgestattet sind. Um zusätzlich zu permanenten Fehlern den gleichen Schutz gegen latente Fehler zu erhalten, benötigt man eine SECDED ECC Kodierung, um zwei Fehler, korrigieren zu können (Doppelbit Fehler Erkennung in einer Zeile mit einem permanenten Fehler ist äquivalent zu einem Parity Fehler in einer fehlerfreien Speicherzeile). Deshalb werden sämtliche Vergleiche gegen eine SECDED Lösung und nicht gegen eine simple SEC Lösung (SEC benötigt ein Kontrollbit weniger als SECDED) durchgeführt

#### *Leitgedanke von SMART Repair*

Die Hauptidee der SMART Repair Lösung besteht darin, dass ein Zuverlässigkeitsproblem mit einem latenten Fehler, der während der Betriebszeit auftritt, oder mit einem Herstellungsfehler, der während des Tests entdeckt und während der Hochlaufphase des Bausteines repariert werden kann, gleichgesetzt wird. Wenn man diese Fehler gleich klassifiziert, hat das den Vorteil, dass man fast den gleichen Hardware und Software Overhead, den man zur Entdeckung der anderen Fehlermechanismen verwendet hat, wiederverwenden kann. Andererseits muss man die Anzahl der Redundanzzeilen erhöhen, um Zuverlässigkeitsfehler, die während der Lebenszeit des Produktes auftreten, zusätzlich zu den Herstellungsfehlern, reparieren zu können.

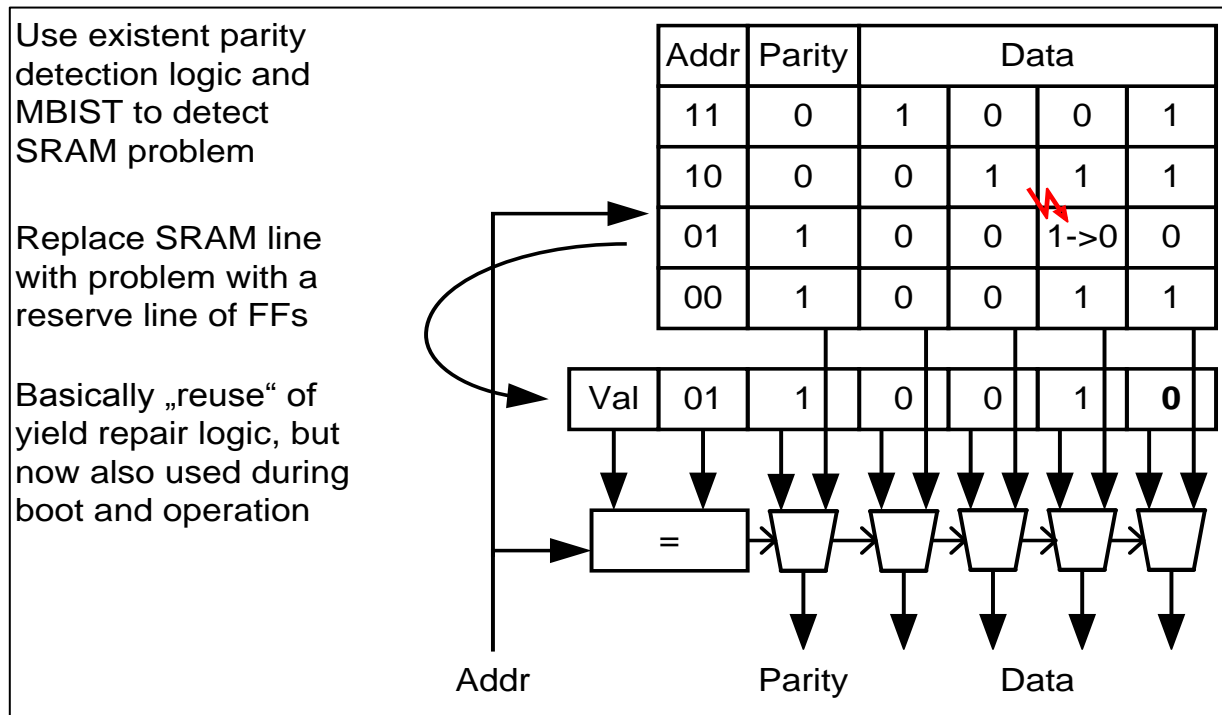


Abbildung VIII-18: Übersicht SMART Repair Funktionalität

### Ausbaustufe 1

Beim Hochfahren des Bausteins nach Anlegen der Versorgungsspannung können Zuverlässigkeitsprobleme, die über die Lebenszeit entstehen, durch einen reduzierten MBISTPLUS Algorithmus, der so viele Speicher wie möglich parallel testet (Energieverbrauch), entdeckt werden. Dabei sind als Randbedingungen die Hochlaufzeit auf der einen und der Energieverbrauch auf der anderen Seite zu beachten, um zu bestimmen, wie viele Speicher parallel getestet werden können. Glücklicherweise können fast alle Fehlermechanismen, die mit dieser Methode repariert werden können, mit einem reduzierten Testalgorithmus entdeckt werden. Normalerweise werden während des Produktionstestes ein Checkerboard und ein March 14N Test (siehe [GOOR]) durchgeführt. Diese beiden Tests dauern insgesamt „(4 + 14) \* Speichertiefe“ Taktzyklen. Für die SMART Repair Lösung kann der Checkerboard Algorithmus (4 Taktzyklen pro Speicheradresse) verwendet werden, weil kompliziertere Lebenszeit Fehler (wie z.B. ein Wortleitungsdekodierfehler) mit der hier beschriebenen Methode der Redundanzaktivierung nicht repariert werden können. Diese Einschränkung ist kein Nachteil gegenüber der ECC Methode, weil diese ebenfalls nicht in der Lage ist, solche Fehlermechanismen auszugleichen. Andererseits können damit die Einschränkungen bezüglich der Testzeit während des Hochlaufens des Bausteins eingehalten werden. Natürlich wird auch der Energieverbrauch durch die Frequenz beeinflusst, so dass ein Kompromiss zwischen Geschwindigkeit, mit der auch zeitabhängige Probleme detektiert werden können, und Energieverbrauch gefunden werden muss. Glücklicherweise ist fast die komplette funktionale digitale Logik des Bausteins während des Hochlaufens deaktiviert, so dass selbst bei maximaler Aktivität der Speicherblöcke (paralleles Testen aller Speicher) der Energieverbrauch innerhalb der spezifizierten Werte

liegen sollte. Zu beachten ist jedoch, dass der schnelle Anstieg der benötigten Energie ein Problem für den externen Spannungsregler darstellen kann.

Sollten während des Hochlaufens defekte Speicherzeilen gefunden werden, so werden die entsprechenden Redundanzregister automatisch programmiert. Normalerweise müsste diese Testroutine während des Hochlaufens gestartet werden, nachdem die Boot Software die Redundanzmodule mit der Reparaturinformation, die während des Produktionstestes ermittelt wurde, vorgeladen hat.

Der Test des Speichers läuft zeitgleich zur Initialisierung des Flash Speichers ab. Nachdem beide Vorgänge abgeschlossen sind, werden die während des Hochlaufens gefundenen Redundanzinformationen mit den Werten im Flash Speicher verglichen. Diese Daten werden dann aneinander angepasst und ggf. erneut in den Redundanzmodulen abgespeichert. Im Produktionstest werden die Speicher mit deutlich komplexeren Algorithmen und unter den unterschiedlichsten Randbedingungen (Temperatur, Spannung) getestet, während der Test während des Hochlaufens nicht zwingend zum gleichen Resultat führen muss.

### *Ausbaustufe 2*

Die zweite Ausbaustufe der SMART Repair Funktion berücksichtigt die Tatsache, dass ein Problem nur unter bestimmten Betriebsbedingungen auftritt. Darüber hinaus können auch zusätzliche Fehler nach dem Hochlaufen oder sporadische Fehler auftreten. Auch um diese Probleme zu lösen wird der (fast) gleiche Lösungsansatz verwendet, der auch bei der Behandlung von latenten Fehlern benutzt wird. Zur Erkennung von latenten Fehlern wird die Parity Erkennungslogik benutzt, die einen NMI an die Systemsoftware sendet. Die Software muss dafür Sorge tragen, dass die Speicher neu initialisiert werden und das System neu gestartet wird. Dieser Lösungsansatz wird auch bei auftretenden Zuverlässigkeitsproblemen vom SMART Repair benutzt. Um solch einen dauerhaften Fehler (im Gegensatz zu einem latenten Fehler, der nach einem Software Reset nicht mehr auftritt) korrekt zu behandeln, muss die Parity Erkennungslogik die automatische Belegung eines zusätzlichen Redundanzregisters im Redundanzblock anstoßen. Die Adresse, bei der der Fehler aufgetreten ist, muss genauso wie die fehlerhaften Daten in das Redundanzregister übertragen werden. Die anschließende Vorgehensweise entspricht der oben beschriebenen beim Auftreten eines latenten Fehlers: ein NMI muss generiert werden, der die Speicher neu initialisiert und das System neu startet. Im Gegensatz zum Hochfahren nach Einschaltung der Versorgungsspannung werden die Inhalte der Redundanzregister in den Redundanzblöcken genauso wie die Inhalte der Speicher nicht neu initialisiert. Damit ist gewährleistet, dass die defekten Speicherzellen weiterhin maskiert bleiben. Weil die Parity Erkennungslogik nicht zwischen einem permanenten und einem latenten Fehler unterscheiden kann, ist auch keine Unterscheidung des Verhaltens möglich. Ein latenter Fehler wird mit sehr hoher Wahrscheinlichkeit beim nächsten Einschaltvorgang nicht wieder auftreten, während dauerhafte Fehler auch zukünftig direkt wieder entdeckt werden. Um eine mögliche Überbelegung der Redundanzregister zu erkennen, ist in jedem MBISTPLUS IP ein Bit implementiert, das dafür sorgt, dass beim Auftreten eines Parity Fehlers in einem Modul, dessen Redundanzregister bereits voll belegt sind, deren Inhalt nicht überschrieben wird. Stattdessen wird ein Überlauf Signal gesetzt,

das von der Software gelesen werden kann. Da der Parity Fehler einen Interrupt erzeugt, ist es Aufgabe der Software herauszufinden in welchem Speicher der Überlauf aufgetreten ist und entsprechend zu reagieren. Es sollte auf jeden Fall ein kompletter Neustart des Systems durchgeführt werden, um mit Hilfe des dann laufenden Speichertests zu überprüfen, ob ein latenter Fehler den Überlauf erzeugt hat. Tritt der Überlauf nach Auswertung dieses Tests und unter Berücksichtigung der Redundanzinformation aus dem Produktionstest dagegen erneut auf, so deutet dies auf einen permanenten Fehler hin, und der Baustein ist als defekt zu betrachten

### Ausbaustufe 3

Es besteht ein gewisses Risiko, dass ein dauerhafter Fehler, der von der Ausbaustufe 2 der SMART Repair Funktion während der Laufzeit gefunden wurde während der Hochlaufphase von Ausbaustufe 1 der SMART Repair Funktion niemals gefunden wird, weil der Fehler nur unter bestimmten Betriebsbedingungen auftritt. Mit einiger Wahrscheinlichkeit wird der Fehler (wenn er erneut auftreten sollte) von Ausbaustufe 2 wieder erkannt und repariert. Allerdings ist solch ein Verhalten äußerst unerwünscht (der Warmstart des Steuerungsbausteins stellt kein kritisches Problem für den Motor selbst dar, könnte aber vom Endkunden bemerkt werden). Theoretisch kann dieses Problem nach einer gewissen Laufzeit immer wieder auftreten, während es beim Hochfahren des Systems niemals entdeckt wird. Um das zu verhindern, wird eine 3. Softwarestufe implementiert. Ausbaustufe 3 verhält sich identisch zu dem Teil der Software, die die Hardware von Ausbaustufe 1 und 2 ansteuert und verhindert, dass solche Fehler immer wieder auftreten. Ausbaustufe 3 besteht aus einer Software Bibliothek, die an den Kunden ausgeliefert wird und die bei jedem Laufzeitfehler, der von Ausbaustufe 2 entdeckt wird, gestartet wird. Sie stellt sicher, dass die Fehlerhistorie im nichtflüchtigen Speicher des Bausteins abgelegt wird. Zusätzlich gibt es eine Erweiterung zur Software, die das Hochlaufen kontrolliert. Diese Erweiterung kann dazu benutzt werden, die von Ausbaustufe 2 gefundenen Fehler zu der Information aus dem Produktionstest hinzuzufügen. Dabei gibt es einen wichtigen Unterschied: Das erste Auftreten eines Fehlers während der Laufzeit darf nicht zusätzlich zu den Produktionsfehlern in die Redundanzregister eingetragen werden. Weil auch latente Fehler während der Laufzeit auftreten und nicht von dauerhaften Fehlern unterschieden werden können, macht es keinen Sinn diesen Fehler direkt in ein Redundanzregister einzutragen. Daher muss die Software das Auftreten von Fehlern und deren Häufigkeit protokollieren. Erst wenn ein Fehler auf einer Adresse zum zweiten Mal auftritt kann die gemeldete Adresse zusätzlich zu den Daten aus dem Produktionstest in den Flash Speicher eingetragen werden. Mit dieser Methode kann ein dauerhaftes Zuverlässigkeitsproblem lediglich zweimal auftreten (weil es dann entweder von Ausbaustufe 1, Ausbaustufe 2 oder einer Kombination aus beiden entdeckt wurde) bevor es dann permanent bei jedem Einschaltvorgang durch die Ausbaustufe 3 repariert wird.

Die Ausbaustufe 3 kann noch erweitert werden, indem man das MBISTPLUS IP dazu benutzt zunächst den fehlerhaften Inhalt der defekten Speicherzeile zu sichern und anschließend den Typ (dauerhaft oder latent) des Fehlers zu bestimmen. Außerdem kann MBISTPLUS dazu benutzt werden, die genaue Bit Position des Fehlers zu bestimmen, was mit einfacher Parity Erkennung nicht möglich ist. Indem man das defekte Bit invertiert ist es dann sogar möglich, das defekte Datum zu reparieren

ohne einen Warmstart durchführen zu müssen. Dies ist auch einer der Gründe, weshalb man das fehlerhafte Datenwort aus der defekten Speicherzeile in das neu allokierte Redundanzregister kopiert. Dadurch, dass Ausbaustufe 3 eine pure Softwarelösung und entsprechend flexibel ist, kann sie jederzeit an die aktuellen Systemanforderungen angepasst werden.

Man sollte sich jedoch stets bewusst machen, dass die Wahrscheinlichkeit für zwei solcher Zuverlässigkeitsprobleme auf einem Baustein, auch wenn sie einzeln zu dppm Raten im Bereich größer 100 führen, sehr gering ist. Sie ergibt sich aus der Multiplikation der Wahrscheinlichkeit für das Auftreten eines einzelnen Defektes, weil es sich um voneinander unabhängige Ereignisse handelt, mit der geometrischen Verteilung der Polymer Partikel auf dem Baustein. Darum ist es sehr unwahrscheinlich, dass ein solches Problem auf einem Chip zweimal auftritt

#### Optimierung und weitergehende Überlegungen

Ein wichtiger Aspekt bei der Verwendung der SMART Repair Lösung ist die Frage, wie viele zusätzliche Redundanzregister eingebaut werden müssen. Dazu muss man wissen, welches Problem ein fehlerhaftes Kontaktloch hervorrufen kann. Ein Kontaktloch (oder auch ein normaler Kontakt) in einer Speicherzelle wird nicht immer nur von einer sondern mehrheitlich von zwei bzw. sogar vier Zellen benutzt. Beachtet man nun, dass das Auftreten von mehreren solcher Zuverlässigkeitsproblemen auf einem Baustein oder sogar innerhalb eines Speichers sehr unwahrscheinlich ist, ist die Annahme, dass vier zusätzliche Redundanzregister pro Speicher zur Lösung des Problems ausreichend sind, verlässlich. Aber auch das Hinzufügen von lediglich zwei zusätzlichen Redundanzregistern kann 85% der Kontaktlochprobleme lösen, da 85% der Kontaktlöcher einer Zelle von zwei Zellen benutzt werden. Selbst ein zusätzliches Redundanzregister kann Probleme, die von Kontakten verursacht werden, deutlich reduzieren, da die Anzahl der Kontakte, die nicht von mehreren Speicherzellen gemeinsam verwendet werden, recht hoch ist.

Außerdem gilt es zu beachten, dass in der Realität lediglich für einen kleinen prozentualen Anteil der verwendeten Speicher die zum Beheben von Produktionsfehlern vorgesehenen Redundanzregister bereits voll belegt sind. In den momentan gefertigten Technologien sind sogar die überwiegende Zahl der Speicher fehlerfrei, so dass nach dem Produktionstest noch alle Redundanzregister verfügbar sind. Berücksichtigt man diese Randbedingungen, so kann man die Redundanzregister für die diskutierten Verwendungszwecke gemeinsam verwenden. Die erreichbaren dppm Werte werden dadurch etwas kleiner, weil Speicher, die z.B. Redundanzregister für den Produktionstest verbrauchen, entsprechend weniger Redundanzregister für die „Lebenszeitreparatur“ zur Verfügung haben. Aber wie früher bereits erwähnt, wenn von vier Redundanzregistern eines für die Reparatur eines Produktionsfehlers benötigt wird, so können mit den verbleibenden drei Redundanzregistern immer noch 85% der Durchkontaktierungsprobleme repariert werden (eigentlich werden sogar nur zwei zur Behebung von 85% der Ausfälle gebraucht). Und bei allen Bausteinen, bei denen die einzelnen Speicherblöcke fehlerfrei sind, können 100% der einfachen Durchkontaktierungsprobleme repariert werden. Die dppm Rate geht also nur für eine recht kleine Anzahl von fehlerhaften Speicherblöcken um 15% nach unten. Das vorgestellte Szenario stellt einen angemessenen Kompromiss dar, wenn man die eingesparte Fläche (gegenüber

einer Lösung mit zusätzlichen Redundanzen) und die Erhöhung der Ausbeute durch die zusätzlichen Redundanzregister betrachtet (bei vier Redundanzregistern können zwei zur Reparatur von Produktionsfehlern verwendet werden ohne dass man die Möglichkeit der späteren Reparatur von Zuverlässigkeitsproblemen allzu sehr beeinträchtigt).

Berücksichtigt man all diese Faktoren und berechnet daraus die sich ergebende Ausbeute und dppm Werte für jeden Speicherblock, so sind zwei Redundanzregister (gegenüber einem für die Reparatur der Produktionsfehler) für die meisten der kleineren Speicher und vier (gegenüber ein bis zwei für die Produktionsfehler) für mittlere Speichergrößen ausreichend. Lediglich große Speicher benötigen sechs Redundanzregister – zwei zur Erhöhung der Ausbeute und zusätzlich vier zur Reparatur von Lebenszeitfehlern. Momentan werden auf einem aktuell produzierten Automobilbaustein durchschnittlich drei Redundanzregister pro Speicher für SMART Repair vorgesehen, wohingegen zur Erhöhung der Ausbeute 1,8 Redundanzregister wirtschaftlich sinnvoll sind.

### *Optimierungspotential*

Um das Optimierungspotential bezüglich einer Verbesserung der dppm Werte bestimmen zu können, ist es erforderlich einen Blick auf die Anzahl der Kontaktlöcher und Kontakte in einer Speicherzelle zu werfen. SMART Repair kann, genauso wie eine ECC Lösung, lediglich Einzelfehler in einem Speicherwort reparieren. Es können, im Gegensatz zur ECC Lösung, keine Fehler im Datenpfad repariert werden. Und es können keine Fehler in den Wortleitungsdekodierern repariert werden, was jedoch auch mit ECC nicht möglich ist. Daraus ergeben sich mehrere wichtige Schlussfolgerungen und Maßnahmen:

Der Speicher muss so aufgebaut sein, dass ein Kontaktloch- oder Kontaktfehler niemals einen Zweibitfehler in einem adressierten Speicherwort hervorrufen darf. Das bedeutet, dass in unterschiedlichen Speicherworten immer nur die gleichen Bits ein gemeinsames Kontaktloch oder Kontakt haben dürfen. Diese Einschränkung gilt auch für ECC Lösungen, weil auch damit nur ein Fehler korrigiert werden kann. Glücklicherweise unterstützen die eingebauten Speicherzellen diesen Ansatz indem sie Bitleitungsmultiplexer verwenden.

Diese Einschränkung gilt auch für den kompletten Datenpfad wenn ECC verwendet wird. Mit SMART Repair können dagegen keine Datenpfadfehler korrigiert werden, weil die Anzahl von zusätzlichen Redundanzregistern limitiert ist.

Jeder Kontaktloch- oder Kontaktfehler in den Wortleitungsdekodierern kann weder auf eine einfache Art und Weise entdeckt noch repariert werden. Daraus ergibt sich zwingend, dass die Kontaktlöcher in den Wortleitungsdekodierern verdoppelt werden müssen um die verbleibende dppm Rate zu reduzieren.

Bei Verwendung von SMART Repair ist es daher notwendig, dass redundante Kontaktlöcher sowohl in der Kontrolllogik als auch im Datenpfad verwendet werden.

Folgt man diesen Empfehlungen so kann man mit Hilfe der SMART Repair Lösung für Kontaktlochprobleme bei geringerem Flächenzuwachs prinzipiell die gleiche dppm Verbesserung erzielen, die man auch mit einer ECC Lösung erreicht.



Berücksichtigt man zusätzlich Kontaktfehler so liefert eine ECC Lösung jedoch nach wie vor eine wesentlich bessere dppm Verbesserung als SMART Repair.

### *Kosten der SMART Repair Lösung*

Berechnet man die Kosten der SMART Repair Lösung so ergibt sich folgendes Bild: In einem Baustein, der aktuell für die Automobilindustrie gefertigt wird, sind 23 Speicherblöcke platziert. Die Durchschnittsgröße der Speicher beträgt 2407 Worte x 35,1 Bit. Unter der Voraussetzung, dass das MBISTPLUS IP aus Testgründen schon verwendet wird und dass bereits Redundanzregister und Parity Logik zum Reparieren von Produktionsfehlern bzw. latenten Fehlern vorhanden sind, werden durch die zusätzlichen Redundanzregister (durchschnittlich 1,2 pro Speicher), die für eine effektive Anwendung der SMART Repair Lösung erforderlich sind, ca. zusätzliche 1500 Flipflops benötigt.

Die Kosten für diese zusätzlichen Flipflops entstehen hauptsächlich durch die zusätzliche Fläche, die bei einem C11FL Baustein im Bereich von ca. 0,5 mm<sup>2</sup> liegt. Dabei sind die größeren Multiplexer innerhalb der Redundanzregister genauso berücksichtigt wie ein Aufschlag von 30%, der für die Verdrahtung benötigt wird. Das zeitliche Verhalten ändert sich so gut wie nicht, da bereits ein Multiplexer verwendet wird, um das vorhandene Redundanzregister in den Lesepfad des Speichers einzufügen. Die zusätzlich benötigte Multiplexerstruktur lässt sich relativ einfach in einem unkritischen Pfad „verstecken“. Der zusätzliche Energieverbrauch im aktiven Modus sowie der zusätzliche Leckstrom, der von den ca. 50000 zusätzlichen Transistoren benötigt wird, kann im Vergleich zu den ca. 6 Millionen Transistoren im Rest der digitalen Schaltung des Bausteins vernachlässigt werden.

Der größte Nachteil der SMART Repair Lösung besteht in der Kundenakzeptanz. Das ist aber hauptsächlich ein psychologisches Problem, da die explizit geforderte Reparaturfunktionalität zum Beheben von Produktionsfehlern auf genau dem gleichen Schaltungsprinzip basiert und eine ECC Lösung zur Reparatur von fehlerhaften Speichern nicht akzeptiert wird. Beide Lösungsmöglichkeiten gleich ist jedoch die Maskierung von fehlerhaften Speichern mit normaler digitaler Logik: die eine Lösung maskiert ein komplettes Speicherwort während die andere Lösung lediglich ein Bit maskiert.

### *Vergleich mit ECC*

Betrachtet man abermals einen durchschnittlich großen Speicher (2407 x 35,1 Bit) eines aktuellen Automobilbausteins und beachtet, dass die meisten der eingesetzten Speicher mit einer Busbreite von 16 Bit beschrieben werden (lediglich bei einer kleinen Anzahl der Speicher werden mehr als 16 Bit gleichzeitig geschrieben), so müsste die Breite der Speicher bei Verwendung einer ECC Lösung jeweils um 5 Bit für 16 Datenbits erhöht werden (6 Bit SECDED ECC verglichen mit 1 Bit bei Verwendung von einem Parity Bit pro 16 Datenbit). Dies ergibt eine zusätzliche Fläche von 30% (5 Zusatzbit für jeweils 16 Datenbit) im Speicherbereich. Wenn man bedenkt, dass die Gesamtgröße aller eingebauten Speicher ca. 7,3 mm<sup>2</sup> von ca. 90 mm<sup>2</sup> Chipfläche ausmacht, so summiert sich dies zu einer Zusatzfläche von ca. 2,2 mm<sup>2</sup>. Dabei ist der Flächenzuwachs noch nicht einmal der größte Nachteil, da der Energieverbrauch im gleichen Maß ansteigt. Der Energieverbrauch im Betrieb steigt proportional zum Flächenzuwachs ebenfalls um 30% an, was sich je nach Grad der

Aktivität auf 50 – 100 mW für einen typischen Baustein aufsummiert. Noch schlimmer stellt sich die Situation beim Leckstrom dar: Die Transistoren innerhalb der Speicher machen heute bereits ca.  $\frac{3}{4}$  aller digitalen Transistoren auf einem Baustein aus. Erhöht man ihre Anzahl um 30% so bedeutet das einen Zuwachs des Leckstroms um ca. 20% oder fast 66 mW. Liegt der Energieverbrauch bereits an der vom Kunden oder Gehäuse vorgegebenen Grenze, so benötigt man ein teureres Gehäuse oder zusätzliche Anschlüsse für eine ausreichende Energieversorgung. Ist der Baustein padlimitiert, so können die zusätzlichen Versorgungsanschlüsse dann zu einem neuerlichen Flächenzuwachs führen, der ein größeres Gehäuse erforderlich macht. Ein anderer Hauptnachteil der ECC Lösung liegt in der Änderung des zeitlichen Verhaltens der Schaltung. Für eine 16/6ECC Anwendung werden 5 Logikstufen benötigt. Erste Versuche haben ergeben, dass dies zu einer Zusatzverzögerung von 1 ns im Lesepfad führt. Außerdem wird der Speicher selbst auch mit ansteigender Breite langsamer. Dies bedeutet eine erhebliche Beeinträchtigung für eine Schaltung, die mit 180 MHz betrieben wird und die über eine nutzbare Taktperiode von ca. 5 ns verfügt. Und auch der Ausbeuteverlust, den 30% mehr Transistoren im Speicher erzeugen, ist nicht unerheblich. Außerdem wird auch die Zuverlässigkeit von 30% mehr Kontaktlöchern und Kontakten, die ausfallen können, negativ beeinflusst, wobei dies mit Hilfe der ECC Schaltung wieder korrigiert wird.

Der Hauptvorteil einer ECC Lösung ist ihre breite Akzeptanz beim Kunden sowie ihre Einfach- und Robustheit. Der Verifikationsaufwand ist minimal, da diese formal durchgeführt werden kann, und auch der Aufwand für die Pflege der Software, die z.B. für die Ausbaustufe 3 der SMART Repair Lösung oder neue bisher noch unbekannte Speicherschwächen benötigt wird, entfällt beim Einsatz von ECC.

### **VIII.2.2.3. ECC**

Die Arbeiten an der untersuchten Alternative zum ECC-Verfahren, dem „SMART Repair“ wurden nicht fortgesetzt, da die Nachteile wie fehlende Kundenakzeptanz und deutliche Erhöhung der Hochlaufzeit die Vorteile überwiegen. Stattdessen wurden detaillierte Untersuchungen und Verbesserungen von ECC durchgeführt.

#### *ECC Codes*

Die entwickelten ECC Algorithmen sind SECDED bzw. DED Codes. Sie verwenden speziell entwickelte Hsia Codes, die für eine Dreifach Fehlererkennung optimiert sind. Sie entdecken einfache Bitfehler und erlauben deren Korrektur, entdecken alle Zweibitfehler und entdecken eine relativ große Anzahl von Dreibitfehlern. Enthalten alle Speicherzellen das logische Signal Eins oder das Signal Null (beides sind unerlaubte Datenwörter), so wird ein Fehler angezeigt. Die Algorithmen sind darüber hinaus invertierbar, d. h. wenn man alle Bits einschließlich der Prüfbits invertiert, erhält man wiederum ein legales Datenwort, das keinen ECC Fehler erzeugt. Diese Eigenschaft erlaubt es einen einfachen linearen Test, der zumindest alle stuck-at Fehler finden kann, ohne dass der ursprüngliche Inhalt des Speichers zerstört wird, laufen zu lassen. Dieser Test greift viermal auf jedes Speicherwort zu:

Lese das komplette Datenwort inklusive der Prüfbits  
Invertiere alle Bits und schreibe sie zurück in den Speicher

Lese das komplette Datenwort inklusive der Prüfbits  
Invertiere alle Bits und schreibe sie zurück in den Speicher

Nach dieser Prozedur stehen die Benutzerdaten unverändert wieder im Speicher und jedes Bit hat einmal den Wert ,0‘ und ,1‘ angenommen. Ein einzelner stuck-at Fehler wird als korrigierbarer Fehler angezeigt.

Die Eigenschaften der entwickelten Codes, deren Einzelheiten im Anhang aufgeführt sind, lassen sich wie folgt zusammenfassen:

Die Erkennung von 3fach Fehlern gegenüber der Verwendung von „normalen“ Hsiao Codes konnte um bis zu 20% verbessert werden.

Durch Optimierung der Zeilen- und Spaltensumme konnte die benötigte Fläche (Anzahl der EXOR Gatter) minimiert werden.

Durch Zeilenbalanzierung (Austausch von Vektoren) konnte die Durchlaufzeit deutlich verringert werden.

Die „All 0/All 1“ Erkennung als Fehler konnte durch Invertierung von zwei Prüfbits erreicht werden.

### VIII.2.2.4. Schaltungsaufbau

#### Hierarchischer Aufbau des MBIST IPs

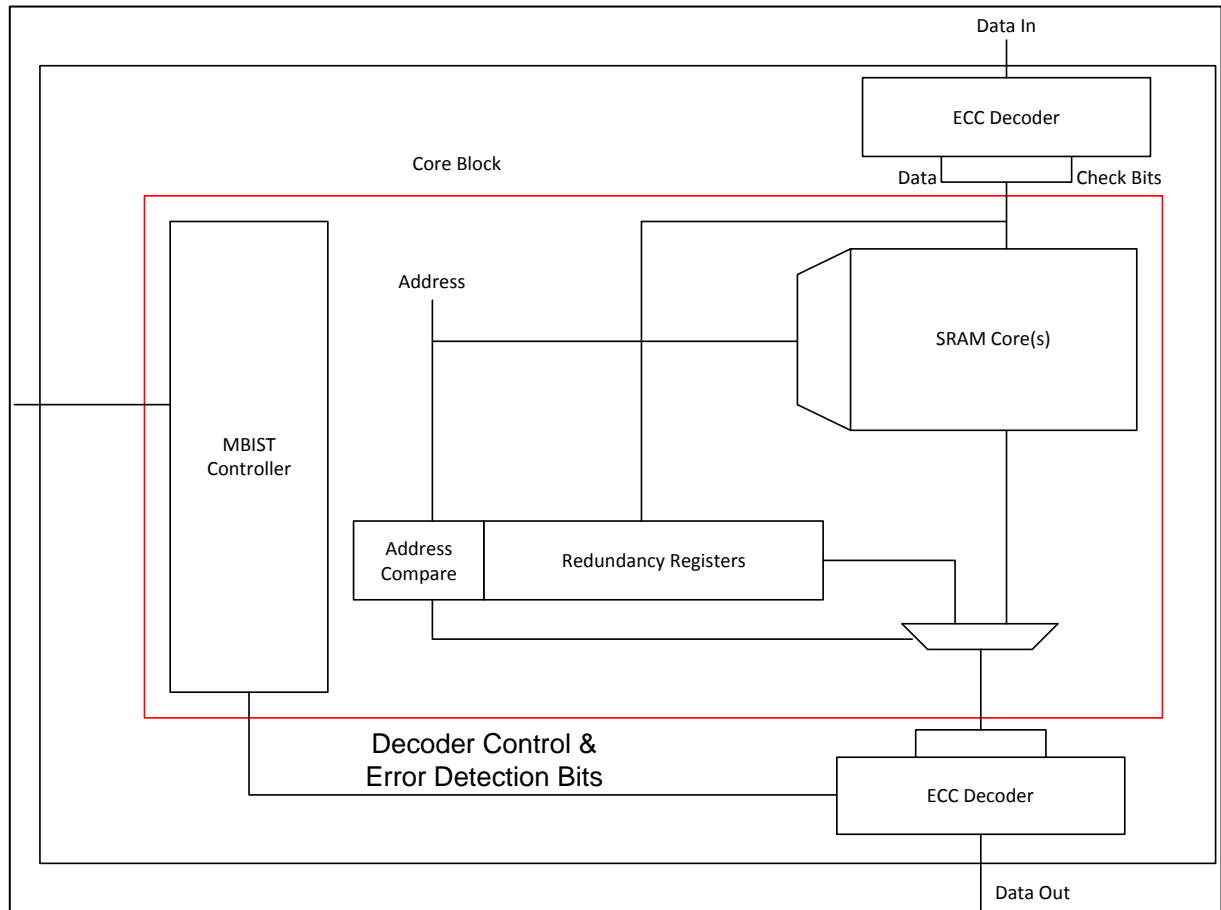


Abbildung VIII-19 Vereinfachte MBIST Hierarchie

Die in Abbildung VIII-19 dargestellte vereinfachte Struktur zeigt die wichtigsten Blöcke aus denen das MBIST IP aufgebaut ist. Es gibt einen internen Kern, in dem die Speicher, Redundanzregister und der eigentliche MBIST Controller, der aus Registern und FSMs besteht, enthalten sind. Eine externe Hülle beinhaltet dann den ECC Encoder und Decoder.

Durch den modularen Aufbau des IPs ist es relativ einfach möglich zusätzliche oder geänderte ECC Funktionalität zu integrieren.

### Strukturelle Erweiterung um ECC Funktionalität

Um die ECC Funktionalität zu integrieren wurde das MBIST IP entsprechend der Abbildung VIII-20 erweitert:

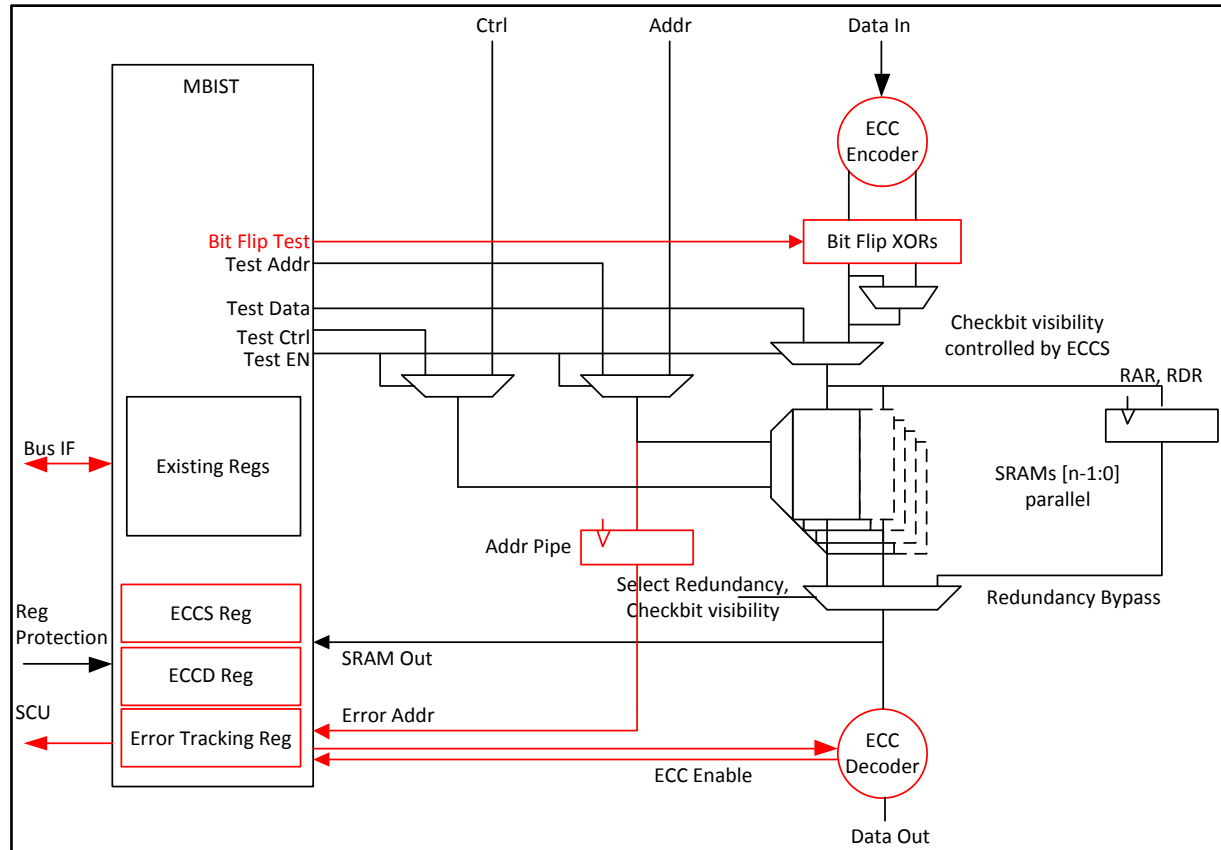


Abbildung VIII-20: Struktur des MBIST IP

Die einzelnen Blöcke haben die folgenden Funktionen:

**ECC Encoder:** Erzeugt die dem Algorithmus entsprechenden Prüfbits

**Bit Flip EXORs:** Die Struktur zwischen Encoder und Decoder ist redundant und deshalb nicht testbar. Mit Hilfe der Bit Flip EXOR Gatter kann man jedes Bit umschalten, so dass beliebige Fehler in den Pfad injiziert und damit der Pfad als ganzer getestet werden kann. Ebenfalls für Testzwecke können die Speicher mittels der Redundanzregister überbrückt werden.

**ECC Decoder:** Vergleicht mit Hilfe der Prüfbits die empfangenen Daten. Beim Auftreten einer Diskrepanz können bei Verwendung eines geeigneten Algorithmus korrigierbare Fehler korrigiert werden, falls die Korrektur eingeschaltet ist. Das Vorliegen eines nicht korrigierbaren Fehlers kann durch ein Signal angezeigt werden.

**ECC Register:** Es gibt drei ECC Register:

**ECCS (Error Code Correction Safety Register):** Kontrollregister für ECC

**ECCD (ECC Code Correction Detection Register):** In diesem Register werden die Fehlermeldungen abgespeichert

ETRR (Error Tracking Register): In diesem Register werden die logischen Adressen eines fehlerhaften Speicherwortes abgelegt. Beinhaltet dieses Register einen gültigen Wert und es tritt ein neuer Fehler auf, so wird ein Overflow Signal generiert.

Address Pipeline Register: Die verwendeten Speicher sind synchrone Speicher, d. h. Lesedaten sind einen Takt nach dem Lesen am Datenausgang verfügbar. Aus diesem Grund muss die Adresse um einen Takt verzögert werden, damit die Adresse in Phase mit den Fehlererkennungssignalen liegt und korrekt im ETRR Register abgespeichert wird.

#### **VIII.2.2.5. Implementierung und Verifikation der ECC Codes**

Zu den verwendeten ECC Codes läuft derzeit ein Patentantrag. Die Implementierung des ECC Encoders und Decoder kann erst nach erfolgter Patenterteilung veröffentlicht werden.

#### **VIII.2.2.6. Verifikation**

Die entwickelten Algorithmen wurden in RTL Code implementiert und formal verifiziert. Da es sich bei der entwickelten Logik um rein kombinatorische Schaltungen handelt, ist die Verwendung dieser Methode am besten geeignet, weil so eine vollständige Verifikation mit mathematischen Methoden erreicht wird. Die unterschiedlichen Konfigurationen (Anzahl der Daten- und Prüfbits, SECCDED oder DED Codes) können über sogenannte Generics eingestellt werden, so dass eine Testbench zur Verifikation sämtlicher Codes verwendet werden kann. Für die formale Verifikation wurde die Software „OneSpin 360™ MV Product Family“ der Firma OneSpin Solutions GmbH <http://www.onespin.de> verwendet. Das Tool ermöglicht die vollständige Verifikation mit Hilfe mathematischer Methoden und erspart das aufwändige Schreiben von Testpattern. Es können skriptgesteuert sämtliche Konfigurationen verifiziert werden, da die unterschiedlichen Werte für die Generics an den RTL Code übergeben werden. Für die Verifikation wurden die entwickelten ECC Encoder und Decoder in eine kleine VHDL Testbench, die in Abbildung VIII-21 dargestellt ist, integriert:

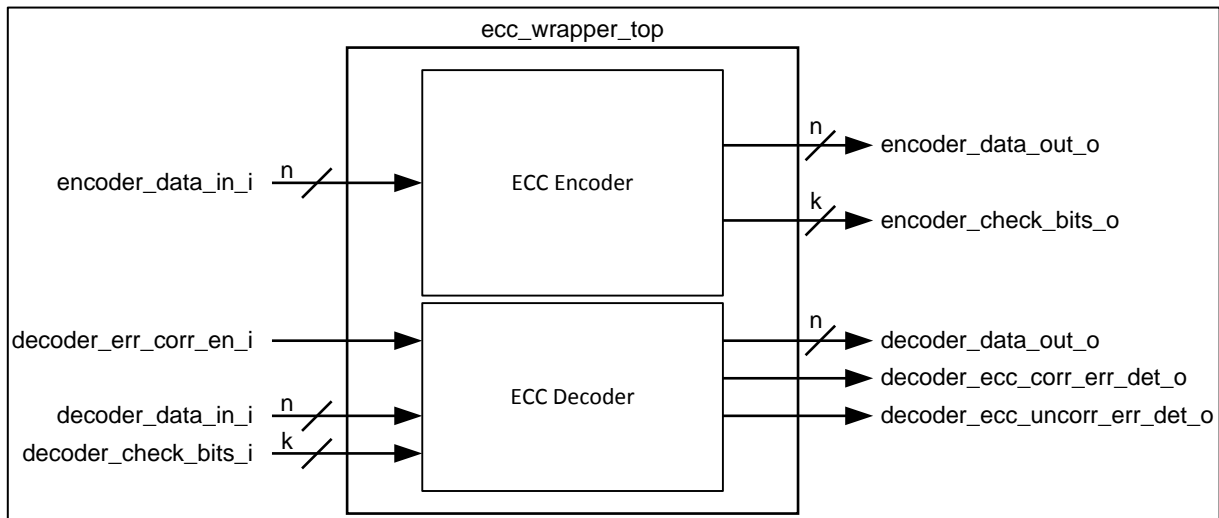


Abbildung VIII-21: Testbench zur formalen Verifikation von ECC Encoder und Decoder

Der Speicher selbst, ist in der Testbench nicht enthalten sondern wird mathematisch beschrieben. Zur Verifizierung der Fehlererkennung und –korrektur werden die Ausgänge des Encoders direkt mit den Eingängen des Decoders verbunden. Mit Hilfe von sogenannten Properties kann man der Software mitteilen welche Eigenschaften der Schaltung verifiziert werden sollen.

Nachdem die Theoreme (im Anhang ist ein Beispielcode für die Verifikation der Eigenschaft „erkenne und korrigiere einfache Bitfehler“ eingefügt), mit denen die Eigenschaften der Schaltung verifiziert werden sollen, geschrieben wurden, wird ein Regressionslauf gestartet. Die Ergebnisse sind dann im Ganzen verfügbar und können für Dokumentationszwecke archiviert werden.

Da die ECC Funktionalität (relativ) unabhängig von der restlichen Logik des MBIST IPs verifiziert werden kann, konnte die ursprüngliche MBIST Verifikationsumgebung zum Großteil wiederverwendet werden. Zusätzliche Stimuli wurden lediglich zur Überprüfung der korrekten Interaktion zwischen ECC und der restlichen Logik notwendig. Zur besseren Validierung der Schaltung im Labor wurden diverse Testmöglichkeiten implementiert, insbesondere ist es über einen einstellbaren Testmodus möglich, gezielt einzelne Bits (sowohl Daten- als auch Prüfbits) zu manipulieren, um dann beim Auslesen des Speichers das fehlerfreie Verhalten der ECC Logik zu überprüfen. Abhängig davon welche und wie viele Bits man manipuliert kann man die Fehlererkennung (korrigierbare und/oder nicht korrigierbare Fehler) und Fehlerkorrektur validieren.

### VIII.2.2.7. Validierung

Labormessungen an den gefertigten Musterbausteinen zeigen, dass die neu entwickelten ECC Funktionen ordnungsgemäß funktionieren. Dies konnte insbesondere mit Hilfe der eingebauten Testlogik, die es relativ einfach macht, einzelne Bits im Speicher zu manipulieren, nachgewiesen werden. Auch das korrekte Verhalten des zerstörungsfreien Invertierungstestes konnte bestätigt werden. Parallel dazu wurden die Bausteine in Evaluierungsboards eingebaut werden, um auch erste

(interne) Systemtests durchzuführen und die Firmware zu evaluieren und debuggen. Die SRAMs selbst wurden mit einer Vielzahl von bekannten Testalgorithmen (siehe z.B. [GOOR]) auf ihre Eignung bezüglich der höchstmöglichen Fehlerabdeckung hin untersucht. Ziel dieser Tests war es, durch eine optimale Wahl der verwendeten Algorithmen mit einer möglichst geringen Anzahl von Testschritten eine höchstmögliche Fehlerabdeckung zu erhalten

### VIII.2.2.8. Verwendete Speichertest Algorithmen

Fehlermodelle für Speicher sind in der einschlägigen Fachliteratur, z.B. [GOOR] oder [HAM] vielfach und ausführlich beschrieben. Ebenso werden in diesen Publikationen eine Vielzahl von unterschiedlichen Speichertest Algorithmen, mit deren Hilfe bestimmte Fehler entdeckt werden können, beschrieben. Für das im Rahmen des Förderprojektes weiterentwickelte IP war es aufgrund der Randbedingungen (Fläche!) notwendig, die maximale Länge einzelner „Marchelementen“ zu begrenzen.

Mit Marchelement bezeichnet man eine Sequenz von Lese- und Schreiboperationen, die sequentiell für eine Speicheradresse durchgeführt werden bevor zur nächsten Speicherzelle weitergegangen wird. Ein Marchelement wird auf alle Adressen des Speichers angewandt bevor. Ein Testalgorithmus wird durch geschweifte Klammern begrenzt {Testalgorithmus}, jedes Marchelement wird durch runde Klammern begrenzt (Marchelement). Marchelemente werden durch Semikolons getrennt und die einzelnen Lese- bzw. Schreiboperationen innerhalb eines Marchelementes sind durch Kommas getrennt. Ein  $\uparrow$ ,  $\downarrow$  oder  $\updownarrow$  bezeichnet die Adressierungsrichtung aufsteigend (von der niedrigsten zur höchsten Adresse), absteigend (von der höchsten zur niedrigsten Adresse) bzw. beliebige Adressierungsrichtung. Eine Operation, die auf eine Zelle angewandt wird, kann entweder ein ‚w0‘ (write ‚0‘), ‚r0‘ (read ‚0‘), ‚w1‘ oder ‚r1‘ sein. Das  $D$  in der Beschreibung des March G Algorithmus bezeichnet eine Verzögerungszeit zwischen zwei Marchelementen.

Mit der im MBIST IP implementierten Logik sind daher die folgenden Algorithmen programmier- und ausführbar:

#	Algorithmus	Sequenz	TL
1	SCAN	{ $\updownarrow(w0); \updownarrow(r0); \updownarrow(w1); \updownarrow(r1)$ }	4n
2	SCAN+	{ $\uparrow(w0); \uparrow(r0); \uparrow(w1); \uparrow(r1); \downarrow(w0); \downarrow(r0); \downarrow(w1); \downarrow(r1)$ }	8n
3	MATS	{ $\updownarrow(w0); \updownarrow(r0, w1); \updownarrow(r1)$ }	4n
4	MATS+	{ $\updownarrow(w0); \uparrow(r0, w1); \downarrow(r1, w0)$ }	5n
5	MATS++	{ $\updownarrow(w0); \uparrow(r0, w1); \downarrow(r1, w0, r0)$ }	6n
6	March C-	{ $\updownarrow(w0); \uparrow(r0, w1); \uparrow(r1, w0); \downarrow(r0, w1); \downarrow(r1, w0); \updownarrow(r0)$ }	10n
7	March A	{ $\updownarrow(w0); \uparrow(r0, w1, w0, w1); \uparrow(r1, w0, w1); \downarrow(r1, w0, w1, w0); \downarrow(r0, w1, w0)$ }	15n
8	March B	{ $\updownarrow(w0); \uparrow(r0, w1, r1, w0, r0, w1); \uparrow(r1, w0, w1); \downarrow(r1, w0, w1, w0); \downarrow(r0, w1, w0)$ }	17n
9	Algorithm B	{ $\updownarrow(w0); \uparrow(r0, w1, w0, w1); \uparrow(r1, w0, r0, w1); \downarrow(r1, w0, w1, w0); \downarrow(r0, w1, r1, w0)$ }	17n
10	March C+	{ $\uparrow(w0); \uparrow(r0, w1, r1); \uparrow(r1, w0, r0); \downarrow(r0, w1, r1); \downarrow(r1, w0, r0); \downarrow(r0)$ }	14n
11	PMOVI	{ $\downarrow(w0); \uparrow(r0, w1, r1); \uparrow(r1, w0, r0); \downarrow(r0, w1, r1); \downarrow(r1, w0, r0)$ }	13n
12	March 1/0	{ $\uparrow(w0); \uparrow(r0, w1, r1); \downarrow(r1, w0, r0); \uparrow(w1); \uparrow(r1, w0, r0); \downarrow(r0, w1, r1)$ }	14n
13	March TP	{ $\downarrow(w0); \uparrow(r0, w1); \uparrow(r1, w0); \downarrow(r0, w1, r1); \downarrow(r1, w0, r0)$ }	11n
14	March U	{ $\updownarrow(w0); \uparrow(r0, w1, r1, w0); \uparrow(r0, w1); \downarrow(r1, w0, r0, w1); \downarrow(r1, w0); \downarrow(r0)$ }	14n



15	March X	$\{\uparrow(w_0); \uparrow(r_0, w_1); \downarrow(r_1, w_0); \uparrow(r_0)\}$	6n
16	March Y		8n
17	March LR	$\{\uparrow(w_0); \downarrow(r_0, w_1); \uparrow(r_1, w_0, r_0, w_1); \uparrow(r_1, w_0); \uparrow(r_0, w_1, r_1, w_0); \uparrow(r_0)\}$	14n
18	March LA	$\{\uparrow(w_0); \uparrow(r_0, w_1, w_0, w_1, r_1); \uparrow(r_1, w_0, w_1, w_0, r_0); \downarrow(r_0, w_1, w_0, w_1, r_1); \downarrow(r_1, w_0, w_1, w_0, r_0); \downarrow(r_0)\}$	22n
19	March RAW	$\{\uparrow(w_0); \uparrow(r_0, w_0, r_0, r_0, w_1, r_1); \uparrow(r_1, w_1, r_1, r_1, w_0, r_0); \downarrow(r_0, w_0, r_0, w_1, r_1); \downarrow(r_1, w_1, r_1, r_1, w_0, r_0); \uparrow(r_0)\}$	26n
20	March RAW1	$\{\uparrow(w_0); \uparrow(w_0, r_0); \uparrow(r_0); \uparrow(w_1, r_1); \uparrow(r_1); \uparrow(w_1, r_1); \uparrow(r_1); \uparrow(w_0, r_0); \uparrow(r_0)\}$	13n
21	March AB	$\{\uparrow(w_0); \uparrow(r_0, w_1, r_1); \downarrow(r_1, w_0, r_0); \downarrow(r_0)\}$	22n
22	March AB1	$\{\uparrow(w_0); \uparrow(w_1, r_1, w_1, r_1, r_1); \uparrow(w_0, r_0, w_0, r_0)\}$	11n
23	March BDN	$\{\uparrow(w_0); \downarrow(r_0, w_1, r_1, w_1, r_1); \downarrow(r_1, w_0, r_0, w_0, r_0); \uparrow(r_0, w_1, r_1, w_1, r_1); \uparrow(r_1, w_0, r_0, w_0, r_0); \uparrow(r_0)\}$	22n
24	March SR	$\{\downarrow(w_0); \uparrow(r_0, w_1, r_1, w_0); \uparrow(r_0, r_0); \uparrow(w_1); \downarrow(r_1, w_0, r_0, w_1); \downarrow(r_1, r_1)\}$	14n
25	March SS	$\{\uparrow(w_0); \uparrow(r_0, r_0, w_0, r_0, w_1); \uparrow(r_1, r_1, w_1, r_1, w_0); \downarrow(r_0, r_0, w_0, r_0, w_1); \downarrow(r_1, r_1, w_1, r_1, w_0); \uparrow(r_0)\}$	22n
26	BLIF	$\{\uparrow(w_0); \downarrow(w_1, r_1, w_0); \uparrow(w_1); \downarrow(w_0, r_0, w_1)\}$	8n
27	Ham5R	$\{\uparrow(w_0); \uparrow(w_1, r_1^5); \uparrow(w_0, r_0^5); \downarrow(w_1, r_1^5); \downarrow(w_0, r_0^5)\}$	25n
28	Ham5W	$\{\uparrow(w_0); \uparrow(w_0^5, r_0); \uparrow(w_1^5, r_1); \downarrow(w_0^5, r_0); \downarrow(w_1^5, r_1)\}$	25n
29	March G	$\{\uparrow(w_0); \uparrow(r_0, w_1, r_1, w_0, r_0, w_1); \uparrow(r_1, w_0, w_1); \downarrow(r_1, w_0, w_1, w_0); \downarrow(r_0, w_1, w_0); \mathbf{D}; \uparrow(r_0, w_1, r_1); \mathbf{D}; \uparrow(r_1, w_0, r_0)\}$	23n
30	Ham_Walk	$\{\uparrow(w_1); \uparrow(w_0); \uparrow(r_0, w_1, r_1, w_0, r_0); \uparrow(r_0, w_1); \uparrow(r_1, w_0, r_0, w_1, r_1); \uparrow(r_1)\}$	15n

Tabelle VIII.1: Benutzbare Speichertest Algorithmen im MBIST IP

Durch umfangreiche Testreihen, die mit einer früheren Version des IP für ältere Technologien durchgeführt wurden [LIN] konnten die folgenden Algorithmen als diejenigen, die eine sehr hohe Fehlerabdeckung erzielen, ermittelt werden:

- Algorithm B
- March U
- March LR
- March SR
- Ham\_Walk

Als effektivste Datenpattern wurden dabei das „column stripe“ (Spalten werden abwechselnd mit 0 und 1 belegt) bzw. „Checkerboard“ (Schachbrett) Pattern ermittelt. Die Adressierungsart spielt für eine hohe Fehlerabdeckung keine wesentliche Rolle.

### VIII.2.2.9. Software Suite

Um die zusätzlichen Features des MBIST IPs anzu steuern zu können, wurde der sogenannte „MBIST Configurator“, eine Software mit graphischer Benutzeroberfläche, entsprechend angepasst und erweitert.

Die generelle Architektur des „MBIST Configurators“ sieht folgendermaßen aus:

- PERL/CGI Skript unter IIS Web Server (Microsoft)
  - Läuft entweder lokal auf dem Arbeitsplatzrechner oder zentral auf einem Remote Server
  - Sämtliche Einstellungen werden auf dem Server gespeichert
- HTML GUI
  - Firefox Browser wird empfohlen
- Hierarchische Test Struktur

- Tests sind produktunabhängig (können aber von der eingesetzten MBIST Version abhängen)
- Jeder Test kann durch einen anderen Test gestartet werden

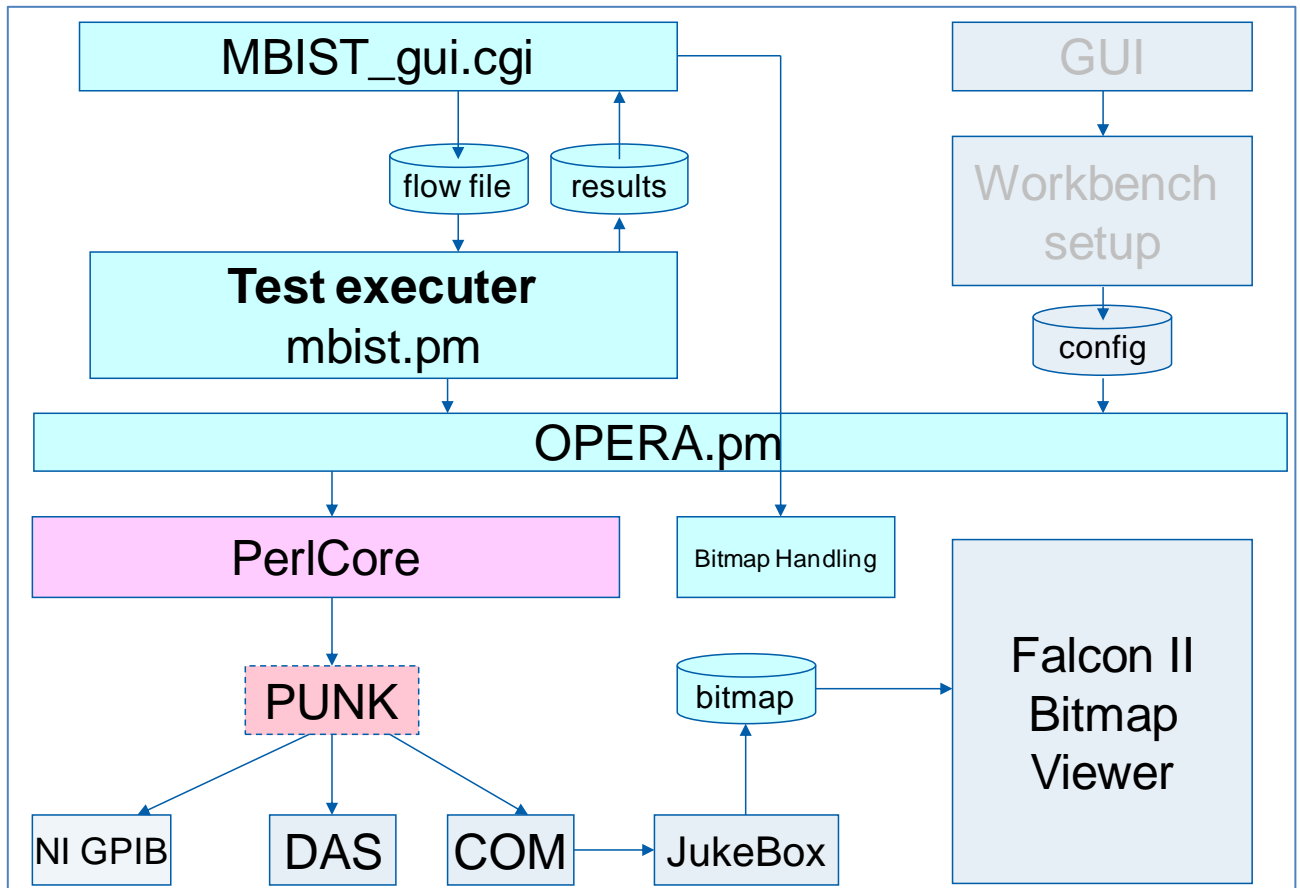


Abbildung VIII-22: Software Architektur

Mit Hilfe des MBIST Configurators kann man den jeweiligen Speicher auf dem Baustein, der getestet werden soll, auswählen. Für den zu testenden Speicher kann man anschließend den Testalgorithmus auswählen und den Test starten.

Bei defekten Speichern meldet die Software bei welchem Testalgorithmus der Fehler aufgetreten ist, und man kann sich den genauen Testverlauf hierarchisch anzeigen lassen. Man kann so einfach erkennen bei welchem Marchelement innerhalb des Algorithmus der Speicher ausgefallen ist, so dass man dieses Marchelement für eine genauere Fehleranalyse erneut laufen lassen kann.

#	edit	ON	Dir	M1	M2	M3	M4	M5	M6	A-Mode	AECC	Toggle	Fast	PST			BG	BL	RA	RA	1.80V 200MHz 35°	
														RD	WR	WW						pat
005	<input type="radio"/>	<input checked="" type="checkbox"/>	UP	Reset Setting Algorithms																	PASS	
010	<input type="radio"/>	<input checked="" type="checkbox"/>	-	CS	-	-	CK	MU		LIN	-	solid	X							PASS		
015	<input type="radio"/>	<input checked="" type="checkbox"/>	-	-	-	-	-	MC		LIN	-	solid	X							PASS		
016	<input type="radio"/>	<input checked="" type="checkbox"/>	-	-	-	-	-	MC		LIN	-	solid	Y							PASS		
020	<input type="radio"/>	<input checked="" type="checkbox"/>	-	Call	Testsuite/HamWalk										-	solid	Y				FAIL	
030	<input type="radio"/>	<input checked="" type="checkbox"/>	-	Call	Testsuite/MARCH_CM										-	solid	X					PASS
040	<input type="radio"/>	<input checked="" type="checkbox"/>	-	Call	Testsuite/MarchRAW										-	solid	Y					PASS
045	<input type="radio"/>	<input checked="" type="checkbox"/>	-	Call	Testsuite/HAM5R										-	solid	Y					PASS
050	<input type="radio"/>	<input type="checkbox"/>	-	Call	Testsuite/decoder										-	solid	-					

#	edit	ON	Dir	M1	M2	M3	M4	M5	M6	A-Mode	AECC	Toggle	Fast	PST			BG	BL	RA	RA	1.30V 200MHz 33°
														RD	WR	WW					
10	<input type="radio"/>	<input checked="" type="checkbox"/>	UP	W1	-	-	-	-	-	-	-	-	-							PASS	
20	<input type="radio"/>	<input checked="" type="checkbox"/>	UP	W0	-	-	-	-	-	-	-	-	-							PASS	
30	<input type="radio"/>	<input checked="" type="checkbox"/>	UP	R0	W1	R1	W0	R0	-	-	-	-	-							FAIL	
40	<input type="radio"/>	<input checked="" type="checkbox"/>	UP	R0	W1	-	-	-	-	-	-	-	-							PASS	
50	<input type="radio"/>	<input checked="" type="checkbox"/>	UP	R1	W0	R0	W1	R1	-	-	-	-	-							FAIL	
60	<input type="radio"/>	<input checked="" type="checkbox"/>	UP	R1	-	-	-	-	-	-	-	-	-							PASS	

Parameters get Inherited

Fails are reported up

Abbildung VIII-23: Auflösung der Testalgorithmen in einzelne Marchelemente

Da es mit Hilfe des MBIST Configurators auch möglich ist, gezielt Fehler in einzelne Speicher zu injizieren, wird das Tool auch dazu benutzt, die neuen Algorithmen für einen Online Test („non destructive inversion test“) zu untersuchen und zu bewerten.

### VIII.2.2.10. Erweiterte ECC Lösungen

Für die unsymmetrischen und, im Vergleich mit den bereits beschriebenen symmetrischen Codes, schmalen ECC Codes, die beim „non destructive inversion test“ verwendet werden, konnten zwei Lösungen erarbeitet werden: eine Lösung benötigt zwei verschachtelte Teilinvertierungen der Speicherdaten während die andere Lösung mit einer Komplettinvertierung, bei der die Checkbits im Invertierungsschritt beeinflusst werden, arbeitet. Nähere Informationen zu diesen beiden neuen Lösungsansätzen kann den beiden im Rahmen des DIANA Projekts eingereichten Patenten [PA1] und [PA2] entnommen werden.

Beide neuen Lösungen sind mittlerweile in VHDL implementiert und formal verifiziert. Voraussichtlich werden sie in einem der nächsten Produkte Verwendung finden.

### VIII.2.3. Zusammenfassung

Es wurde zunächst untersucht inwieweit es eine Alternative zum ECC (Error Code Correction)-Verfahren gibt, um Zuverlässigkeitsprobleme bei SRAMs, die durch immer kleiner werdende Transistorstrukturen bei gleichzeitig steigender Anzahl der Transistoren auf Grund steigender Speichergröße entstehen, zu beheben. ECC ist die klassische Methode, um solche Probleme zu handhaben. Ungeachtet der Einfachheit und Robustheit dieser Lösung stellt sie jedoch für einige Anwendungen

nicht die ideale Lösung dar, weil die klassische ECC einen erheblichen Flächen-, Energie- und Zugriffszeitnachteil hat und durch zusätzliche Zellen zunächst die Ausbeute, Zuverlässigkeit und Verfügbarkeit vermindert werden. Dies ist besonders für Systeme in denen partielle Schreibzugriffe, die aufgrund von Leistungsanforderungen nicht auf Read-Modify-Write Zugriffe abgebildet werden können, ein Problem. Da sich die Anzahl von Kontrollbits logarithmisch proportional zur Anzahl der Datenbits verhält, verursachen schmalere Zugriffsbreiten einen größeren Zusatzaufwand als breite Zugriffe. In dieser Aufgabe wurde daher zunächst eine alternative Methode, die sogenannte „SMART Repair“ Methode, untersucht, die bereits existierende Schaltungsteile wiederverwendet, um das Problem mit einem anderen Lösungsansatz, der weniger Fläche, Energieverbrauch und Zugriffszeitnachteile erzeugt als die ECC, zu beheben.

Die erarbeiteten Lösungen wurden näher untersucht, implementiert und mit existierenden ECC Lösungen verglichen. Darüber hinaus wurden noch zusätzliche Lösungsansätze mit optimierten ECC Codes untersucht, die dann letztendlich implementiert worden sind, da die Ergebnisse der „SMART Repair“ Lösung nicht die erwünschten Resultate lieferten, und es sich in Diskussionen mit potentiellen Kunden gezeigt hat, dass diese der „SMART Repair“ Funktionalität sehr ablehnend gegenüberstehen. Darüber hinaus hat sich gezeigt, dass die zusätzliche Verzögerung, nach der das System betriebsbereit ist und die durch Abarbeitung der für die „SMART Repair“ notwendigen Firmware entsteht, ebenfalls nicht akzeptabel erscheint. Deshalb wurde zunächst noch an verbesserten ECC Algorithmen, die dann letztendlich auch implementiert und verifiziert wurden, weitergearbeitet.

Da es sich bei den entwickelten ECC Codes um rein kombinatorische Schaltungen handelt, konnte ihre Funktionalität sehr gut durch formale Verifikation (vollständige Verifikation mittels mathematischer Modelle) vollständig überprüft werden. Die resultierende Schaltung wurde in ein Pilotprojekt integriert. Die gefertigten Bausteine wurden dann im Labor überprüft und validiert.

Darüber hinaus wurden als Erweiterung der ursprünglich geplanten Arbeiten zusätzlich zu den beschriebenen Hsiao Codes sogenannte CPC (Cross Parity Check) und unsymmetrische invertierende Hsiao Codes entwickelt. CPC Codes sind im Vergleich zu Hsiao Codes schneller, benötigen allerdings mehr Fläche. Diese Zusatzarbeiten wurden durchgeführt, da absehbar ist, dass schnellere ECC Schaltungen für zukünftige Systeme, benötigt werden. Unsymmetrisch invertierende Hsiao Codes ermöglichen im Vergleich zu symmetrischen invertierenden Hsiao Codes die Verwendung von schmalere Speicher (Flächenreduzierung).

#### **VIII.2.4. Literatur**

[KIAM] Kiamal Pekmestzi, Nicholas Axelos, Isidoros Sideris, Nicolaos Moshopoulos, “A BISR Architecture for Embedded Memories”, 14th IEEE International On-Line Testing Symposium 2008

[OEHL] Philipp Öhler, Alberto Bosio, Giorgio Di Natale, Sybille Hellebrand, “A Modular Memory BIST for Optimized Memory Repair”, 14th IEEE International On-Line Testing Symposium 2008

[HSIAO] C. L. Chen, M. Y. Hsiao, „Error-Correcting Codes for Semiconductor Memory Applications: a State-Of-The-Art Review", "IBM Journal of Research and Development", volume 28, number 2, pp. 124--134, March 1984

[GOOR] A. J. van de Goor, “Testing semiconductor memories: theory and practice”, ComTex Publishing, Gouda, The Netherlands, 1998

[HAM] S. Hamdioui, “Testing static random access memories: Defects, fault models and test patterns”, Kluwer Academic Publishers, Dordrecht, The Netherlands

[LIN] Michael Linder, “Test Set Optimization for Industrial SRAM Testing”, Dissertation an der Technischen Universität München, Lehrstuhl für Entwurfsautomatisierung, 2012

[PAT1] S. Hosp, Infineon, M. Gössel, “Patentvorschlag Prüfbit-Kompaktierung für Cross-Parity-Codes”, 2013

[PAT2] Infineon, K. Oberländer, S. Hosp, M. Gössel, „Patentvorschlag Invertieren von Bits“, 2012

### **VIII.3. Ergebnisse zur Aufgabe 2.2 (Teil2): Online Speicher-Selbsttest und Selbstreparatur in der Applikation für Flash-Speicher**

Die Anforderungen an Flash Speicher die in Automotive-Microcontrollern integriert sind steigen stetig an. Das betrifft wichtige Parameter wie Speichergrößen (bis zu 16MB) als auch die Anzahl der geforderten Schreib-/Lösch- Zyklen, die je nach Speicherart zwischen 1k und 500k Zyklen liegen können. Randbedingungen wie Temperatur (170°C Tjunction), erhöhte Zuverlässigkeit und neue Störmechanismen durch elektromagnetische Felder (insbesondere bei Elektro- und Hybridfahrzeugen) verschärfen die Situation zusätzlich. An höchster Stelle steht im und am Auto die Sicherheit der beteiligten Personen was dem Microcontroller und Flashspeicher maximal robuste und zuverlässige Operation abverlangt.

Bei Speichern spielt für die Erhöhung der Zuverlässigkeit Fehlererkennungs- und Korrekturschaltungen eine zentrale Rolle. Die Grundlage hierfür bildet die Codierungstheorie und die dazugehörigen Fehlerkorrektur Codes (Error Correction Codes). Die Anforderungen an Korrektur- und Erkennungsfähigkeit wachsen dabei insbesondere wenn neue Technologien mit kleineren Transistor - Geometrien eingeführt werden müssen, um dem steigenden Kostendruck Rechnung zu tragen. Kleinere Zellabmessungen bedingen aber in der Regel eine höhere Empfindlichkeit für Defekte und damit ein erhöhtes Ausfallrisiko.

Im Gegensatz zum Consumer – Bereich (z.B. Flash Karten) müssen im Automobil – Bereich aufgrund der notwendigen Echtzeitfähigkeit sehr schnelle kombinatorische Fehlerkorrekturschaltungen entwickelt werden, die so „auf dem Markt“ nicht verfügbar sind. Außerdem sind aufgrund der verschiedenen Safety Standards (z.B. ASIL-D) spezielle Eigenschaften der Korrekturschaltungen notwendig, wie z.B. eine eingebettete Adressfehlererkennung und eine hohe Fehlererkennungsrate bei Multi-Bit Fehlern. Durch verschiedene Speicherarten (Code-Flash und Data-Flash), sowie

verschiedene Bus-Breiten (64 .. 256 bit) sind unterschiedliche für den jeweiligen Einsatzzweck optimierte Codes notwendig.

Aufgrund der Relevanz der Fehlerkorrektur und -erkennung im Flashspeicher muss in zunehmenden Maß auch die zuverlässige Funktion der Fehlererkennung mitberücksichtigt werden.

Ziel dieser Aufgabe ist zum einen existierende Fehlerkorrektur Codes auf den Einsatz im Automobilbereich zu bewerten. Der andere Aspekt ist die Entwicklung einer entsprechenden Online-ECC Überprüfung.

### **VIII.3.1. Online-ECC Überprüfung**

Im Projektzeitraum wurde eine On-line-Fehlererkennungsschaltung zur Detektion von Fehlern in einer ECC für einen linearen fehlerkorrigierenden Code C entwickelt. Dabei besteht die ECC aus einem (ersten) Syndromgenerator SG1, einem nachgeschalteten Decoder DC1 und aus einer zusätzlichen Schaltung DFA1 (Daten-Fehler-Anzeige) zur Fehleranzeige der aufgetretenen Fehler in den zu korrigierenden Daten.

Der Syndromgenerator SG1 und der Decoder DC1 bilden die Korrekturschaltung KOR1. Diese Korrekturschaltung bestimmt die Korrekturbits. Die Korrekturbits werden mit den zu korrigierenden Daten bitweise stellenrichtig zu den korrigierten Daten XOR-verknüpft.

Die Fehler der ECC werden dadurch erkannt, dass die korrigierten Daten in einen (zweiten) Syndromgenerator SG2 eingegeben werden. Ein von diesem Syndromgenerator SG2 ausgegebenes Syndrom zeigt dann an, ob nach der Korrektur ein Codewort oder kein Codewort vorliegt. Am Ausgang des zweiten Syndromgenerators erfolgt keine Korrektur, sondern nur eine Fehlererkennung.

Liegt kein Codewort vor, dann kann das prinzipiell dadurch verursacht sein,

- dass ein Fehler in der ECC vorliegt,
- dass ein nicht korrigierbares Wort am Eingang der ECC anliegt,
- oder dass sowohl ein Fehler in der ECC vorliegt als auch ein nicht korrigierbares Wort am Eingang der ECC anliegt.

Die Wahrscheinlichkeit, dass ein korrektes Codewort, das am Eingang der ECC anliegt, im Falle eines Fehlers der ECC in ein anderes Codewort gestört wird, ist gering, sodass ein Fehler der ECC praktisch stets erkannt wird. Wird ein nicht korrigierbares Codewort von einer fehlerfreien ECC verarbeitet, dann ist das Resultat kein Codewort, und es wird auch von dem (zweiten) Syndromgenerator als Nicht-Codewort erkannt.

Ebenso ist die Wahrscheinlichkeit gering, dass ein nicht korrigierbares Codewort durch einen Fehler in der ECC in ein Codewort korrigiert wird, sodass auch in dem Fall, wenn ein nicht korrigierbares Codewort am Eingang der ECC anliegt ein zusätzlicher Fehler in der ECC praktisch nie zur Anzeige "fehlerfrei" fährt.

Der technische Aufwand für den zweiten Syndromgenerator wurde begrenzt. Prinzipiell könnte der zweite Syndromgenerator SG2 dem ersten Syndromgenerator SG1 gleich sein. Im verwendeten Entwurf realisiert der zweite Syndromgenerator

SG2 nur eine Teilmenge der Komponenten des Fehlersyndroms des ersten Syndromgenerators SG1. Dadurch ist eine Flächeneinsparung für diesen Syndromgenerator etwa um den Faktor 2 möglich. Die Anzahl der Komponenten des Fehlersyndroms, die von dem zweiten Fehlersyndromgenerator SG2 realisiert werden muss, ist durch die Wahrscheinlichkeit bestimmt, mit der ein beliebiger Fehler in der ECC erkannt werden soll. Je nach erforderlicher Fehlererkennungswahrscheinlichkeit ist damit auch der erforderliche Flächenaufwand festgelegt. In dem realisierten Entwurf wurden 12 Bits Wortbreite für den zweiten Syndromgenerator implementiert. Auf dem betrachteten Chip können mehrere gleiche ECCs für den gleichen linearen Code parallel zur Überwachung der mehreren ECCs benutzt. Dadurch kann ein einziger (zweiter) Syndromgenerator SG2 zur On-line Fehlererkennung aller parallel arbeitenden ECCs verwendet werden, deren korrigierte Daten dann bitweise XOR-verknüpft werden müssen, bevor sie in den Syndromgenerator SG2 eingegeben werden, da die bitweise XOR-Verknüpfung von Codevektoren wieder ein Codewort ergibt.

Aufgrund der gemeinsamen Fehlererkennung für die verschiedenen ECCs ist dann allerdings nicht mehr erkennbar, welche der ECCs fehlerhaft war, wenn ein Fehler in irgendeiner der ECCs aufgetreten ist. Im betrachteten Fall ist der fehlerkorrigierende Code C ein 2-Bit fehlerkorrigierender und 3-Bit fehlererkennender verkürzter BCH-Code mit einbezogener Gesamtparität mit dem Codeabstand  $d = 6$ . Die vom Adressengenerator erzeugten Adressenbits, die nicht abgespeichert werden, sind als Datenbits in die Fehlererkennung bzw. Fehlerkorrektur einbezogen.

Der zweite Syndromgenerator SG2 wurde dabei so implementiert, dass datenabhängig für eingegebene korrekte Codewörter der 12-Stellig Outputvektor zwischen  $0, 0, \dots, 0$  und  $1, 1, \dots, 1$  wechselt, so dass alle Stuck-at-Fehler auf den Ausgängen und interne Fehler des zweiten Syndromgenerators online erkannt werden können.

### **VIII.3.1.1. Beschreibung des betrachteten Codes C**

Wir betrachten einen Code C mit  $k$  Nutzdatenbits  $u = u_1, \dots, u_k$ ,  $l$  Adressenbits  $a_1, \dots, a_l$  und  $m$  Prüfbits  $c = c_1, \dots, c_m$ , die ein Codewort  $v = u, a, c$  eines BCH-Codes C mit einbezogener Gesamtparität der Länge  $n = k + l + m$  bilden. Die Datenbits des Codes sind die Bits  $u, a$ , und die Prüfbits sind die Bits  $c$ . Der Code C ist in der Lage, 1-Bit Fehler und 2-Bit Fehler zu korrigieren und 3-Bit Fehler zu erkennen. Sein Codeabstand ist  $d = 6$ . Die Bits  $u$  und  $c$  werden unter der Adresse  $a$  in einem Speicher gespeichert. Tritt kein Fehler auf, dann gilt:

- Beim Schreiben werden die Bits  $u, c$  unter der Adresse  $a$  in den Speicher geschrieben.
- Beim Lesen werden unter der Adresse  $a$  die Bits  $u, c$  aus dem Speicher gelesen.

Die Prüfbits  $c$  sind aus den Datenbits  $u, a$  zu

$$C = H \cdot (u, a)^T$$

Bestimmt, wobei

$$H = \begin{pmatrix} H_1 \\ H_3 \\ P \end{pmatrix}$$

die H-Matrix des betrachteten (verkürzten) BCH-Codes mit einbezogener Gesamtparität -hier in separierter Form- mit  $P = (1, \dots, 1)$  ist. Die konkrete Form der H-Matrix beeinflusst das beschriebene Vorgehen nur wenig.

Wir betrachten nun Fehler in den Bits  $u, a, c$ . Wie ausgeführt, werden die Bits  $u, c$  unter der Adresse  $a$  in den Speicher geschrieben. Die im Speicher abgespeicherten Daten  $u, c$  können fehlerhaft geworden sein, wenn sie ausgelesen werden. Ebenso kann die Adresse fehlerhaft sein. Anstelle von  $a$  kann die Adresse  $a_0$  auftreten. Die ausgelesenen Daten werden mit  $u_0, c_0$  bezeichnet. Die Daten  $u_0, c_0$  werden unter der Adresse  $a_0$  ausgelesen. Im Falle eines Daten- oder Adressenfehlers gilt:

$$(u, a, c) \neq (u_0, a_0, c_0).$$

Die Aufgabe der ECC ist es, derartige Fehler zu korrigieren bzw. anzuzeigen. 1-Bit Fehler und 2-Bit Fehler in  $u$  und  $c$  werden durch die ECC korrigiert. 3-Bit Fehler in  $u$  und  $c$  werden nur als 3-Bit Fehler angezeigt. Obwohl 1-Bit und 2-Bit Fehler korrigiert werden, werden sie als 1-Bit Fehler bzw. als 2-Bit Fehler von der Einheit DFA1 angezeigt. Fehler in den Adressenbits  $a$  werden nicht korrigiert, sondern nur erkannt.

Für einen angezeigten 3-Bit Fehler ist der konkrete Entwurf des Korrektors zu beachten. So kann durch einen angezeigten 3-Bit Fehler die Korrektur blockiert werden, oder der Korrektor führt eine fehlerhafte Korrektur aus. Zusätzlich zu Datenfehlern und Adressenfehlern können außerdem Fehler in der ECC, d. h. Fehler in dem Syndromgenerator SG1, dem Korrektor KOR1 und in der Schaltung DFA1 zur Fehleranzeige auftreten. Das können sowohl permanente als auch transiente Fehler sein. Die Fehler in der ECC treten im Vergleich zu Speicherfehlern mit sehr kleiner Wahrscheinlichkeit auf, nachdem die ECC beim Start-up getestet wurde.

Es sind die folgenden Situationen zu betrachten:

- Es sind nur Daten- und/oder Adressenfehler vorhanden (mit relativ hoher Wahrscheinlichkeit), die ECC ist korrekt.
- Es sind nur Fehler in der ECC vorhanden (mit relativ niedriger Wahrscheinlichkeit), die Daten und die Adresse sind korrekt.
- Es sind gleichzeitig sowohl Daten- und/oder Adressenfehler als auch Fehler in der ECC vorhanden (mit sehr niedriger Wahrscheinlichkeit, wir sprechen von Fehlern höherer Ordnung). Sowohl die Daten und/oder die Adresse sind fehlerhaft als auch die ECC ist fehlerhaft.

Die Aufgabe der realisierten Lösung ist es, On-line, d. h. ohne Unterbrechung im laufenden Betrieb die Fehler in der ECC zu erkennen. Für diese Lösung gilt:

- Mit sehr hoher Wahrscheinlichkeit werden Fehler in der ECC erkannt, für alle Inputs, die keine Fehler in den Daten und Adressen aufweisen.



- Mit sehr hoher Wahrscheinlichkeit werden Fehler in den Adressen erkannt, selbst wenn die Teilschaltung DFA1 der ECC keinen Adressenfehler anzeigt. Ein Fehler wird dann durch den zweiten Syndromgenerator erkannt.
- Mit sehr hoher Wahrscheinlichkeit werden vorliegende 3-Bit Fehler erkannt, selbst wenn die ECC (fehlerhaft) keinen 3-Bit Fehler anzeigt.
- Liegt ein 1-Bit oder 2-Bit Fehler in den Daten vor, der von der ECC richtig korrigiert wird und wird dieser Fehler (auf Grund der zusätzlich fehlerhaften DFA1) nicht von der DFA1 angezeigt, dann wird dieser Fehler der DFA1 von der vorgeschlagenen Lösung nicht erkannt.

Würden hier keine speziellen Maßnahmen für die Fehlererkennung in der DFA1 realisiert, dann würden im Falle eines Fehlers in der DFA1 die Anzahl der 1-Bit oder 2-Bit Fehler nicht richtig aufsummiert werden kann, obwohl die eigentliche Funktionsweise der funktionellen Schaltung noch korrekt erfolgt. Im Entwurf wurde die DFA1-Einheit verdoppelt.

Systemkritische Fehler werden selbst dann praktisch sicher erkannt, wenn die DFA1 fehlerhaft ist, was aber bei einer verdoppelten DFA1 praktisch nicht auftreten kann.

### VIII.3.1.2. Beschreibung der realisierten Lösung

Abbildung VIII-24 zeigt die ECC mit nachgeschalteten Syndromgenerator SG2, der zur On-line Fehlererkennung von Fehlern in der ECC dient. Die Daten  $u$ ,  $a$  werden in den Coder eingegeben, der nach der Beziehung

$$c = G \cdot (u, a)^T$$

die Prüfbits  $c$  bildet.

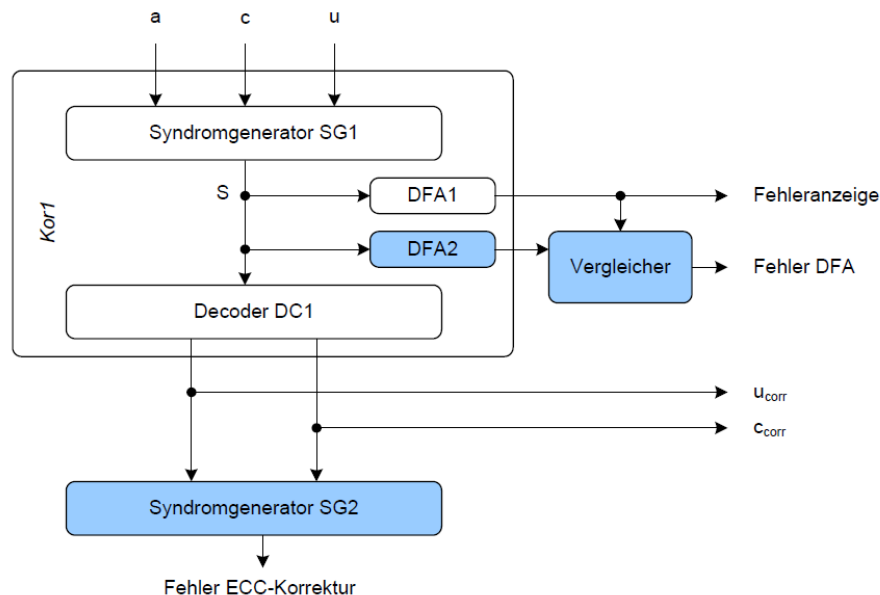


Abbildung VIII-24: Aufbau der ECC und zugehörige Online-Checker

Die Bits  $u$ ,  $c$  werden unter der Adresse  $a$  in den Speicher eingeschrieben. Wenn die Daten aus dem Speicher gelesen werden, können sie korrekt oder fehlerhaft sein. Unter der Adresse  $a$  stehen dann die Werte  $u_0$ ,  $c_0$ . Das Lesen erfolgt unter der

Adresse  $a_0$ . Dabei kann  $a = a_0$  oder, wenn eine fehlerhafte Adresse verwendet wird,  $a \neq a_0$  gelten. Beim Lesen unter der Adresse  $a_0$  werden  $u_0, c_0$  von Speicher ausgegeben. Der Syndromgenerator SG1 bildet das Syndrom  $S(u_0, a_0, c_0)$  nach der Beziehung

$$S(u', a', c') = H \cdot (u', a', c')^T$$

wobei das Syndrom die Form

$$S = s_1, s_3, P$$

hat. (In einem alternativen Entwurf, in dem die Parität z. B. aus  $s_3$  ableitbar ist, wenn die Spalten der H-Matrix so gewählt sind, dass die unteren  $m$  Komponenten der H-Matrix jeweils eine ungerade Anzahl von Einsen aufweisen ergibt sich praktisch nur der Unterschied, dass  $P$  durch einen XOR-Baum aus  $s_3$  bestimmt wird).

Aus dem Syndrom  $S$  wird durch die Schaltung zur Daten-Fehler Anzeige DFA1 bestimmt, ob kein Fehler, ein 1-Bit Fehler, ein 2-Bit Fehler oder ein 3-Bit Fehler angezeigt wird. Es gilt:

kein Fehler wird angezeigt, wenn  $S = 0$  ist,

- ein 1-Bit Fehler wird angezeigt, wenn  $S \neq 0$ , und  $s_1^3 = s_3$  und  $P = 1$
- gilt,
- ein 2-Bit Fehler wird angezeigt, wenn  $S \neq 0$ , und  $s_1^3 = s_3$  und  $P = 0$  gilt,
- ein 3-Bit Fehler wird angezeigt, wenn  $S \neq 0$ , und  $s_1^3 = s_3$  und  $P = 1$  gilt.

Das Syndrom  $S$  wird in den Decoder eingegeben, der für  $u_0, a_0$  und  $c_0$  die zu korrigierenden Bits  $K(u_0), K(a_0)$  und  $K(c_0)$  bestimmt. Ist  $K(a_0) \neq 0$ , dann liegt ein Adressenfehler vor. Adressenfehler werden nicht korrigiert.

Aus  $u_0$  und  $K(u_0)$  wird  $u_{corr} = u_0 \oplus K(u_0)$  und aus  $c_0$  und  $K(c_0)$  wird  $c_{corr} = c_0 \oplus K(c_0)$  gebildet.

Der (erste) Syndromgenerator SG1 und der nachgeschaltete Decoder DC1

bilden den Korrektor KOR1. Zur Fehlererkennung der ECC dient der (zweite) Syndromgenerator SG2, in den  $u_{corr}, a_0$  und  $c_{corr}$  eingegeben werden. Die eventuell fehlerhafte Adresse  $a_0$  wird unkorrigiert eingegeben. Der Syndromgenerator SG2 bildet nach der Beziehung

$$T = K \cdot (u_{corr}, a', c_{corr})^T$$

das Fehlersyndrom  $T$ , wobei  $K$  die H-Matrix dieses Syndromgenerators bezeichnet. Die Matrix  $K$  besteht hier aus einer Teilmenge von 12 Zeilen der Matrix  $H$ .  $T$  aus 12 Komponenten des Syndroms  $S$ .

Etwas vereinfacht gilt: Ist  $T=0$ , dann wird kein Fehler der ECC durch das Fehlersignal  $E_2$  angezeigt. Ist  $T \neq 0$ , dann wird ein Fehler der ECC durch das Fehlersignal  $E_2$  angezeigt.

Die Fehleranzeigeschaltung DFA1, welche 1-Bit, 2-Bit, 3-Bit und Adressenfehlern signalisiert wird vollständig mittels DF2 vollständig redundant ausgelegt. Das bewährte Verfahren von Verdopplung und Vergleich erlaubt das Erkennen jedes funktionalen Fehlers sowohl in DF1 als auch in DF2 sofern ein single-point-of-failure Model angenommen wird.

Im Rahmen des Diana-Projekts konnte gezeigt werden, dass mit Hilfe des nachgeschalteten Syndromgenerators SG2 Fehler sowohl in der ECC-Logik als auch im Syndromgenerator SG2 mit sehr hoher Wahrscheinlichkeit erkennbar sind.

### ***VIII.3.1.3. Experimentelle Untersuchungen und Verifikation***

Die Mächtigkeit der On-line Fehlererkennung mittels des nachgestellten Syndromgenerators SG2 wurde durch Simulationen auf Basis eines VHDL-Modells untersucht. Hierzu wurden 100.000 Fehlervektoren mit zufälligem Fehlergewicht in die Fehlererkennungsschaltung auf Basis des Syndromgenerators SG2 eingegeben. Die experimentellen Ergebnisse zeigen, dass die Fehlererkennungsrate deutlich über 99.9% liegt. Somit wird ein Fehlverhalten der Korrekturschaltung KOR1 praktisch immer erkannt.

Um die korrekte Funktionsweise der Fehlererkennungsschaltung auf Basis des Syndromgenerators SG2 sicherzustellen, wurde dieses Modul formal verifiziert. Insbesondere wurde durch dieses Verfahren bewiesen, dass jedes Codeswort des Codes C, welches am Eingang des Syndromgenerators SG2 angelegt wird, zu einer positiven Bewertung der Funktionalität der Korrekturschaltung KOR1 durch die Fehlererkennungsschaltung führt. Ferner wurde gezeigt, dass Fehler bis zu einem Hamming-Gewicht von 3 sowie alle Fehler mit ungeradem Hamming-Gewicht sicher durch den Syndromgenerator SG2 erkannt werden.

### ***VIII.3.1.4. Test der ECC-Monitore beim Startup***

Zusätzlich zu den ECC-Monitoren wurde ein Startup-Test zur Funktionsüberprüfung der redundant ausgelegten Fehlererkennungsschaltung und des nachgeschalteten Syndromgenerators SG2 erarbeitet. Ziel dieses Startup-Test ist es, eventuelle Fehlfunktionen, bedingt durch defekte Hardwareelemente, zu identifizieren. Da derartige Defekte zu einem vollständigen Ausfall des Monitors zur Überwachung der Fehlererkennungsschaltung führen können, ist der Test so ausgelegt, dass er in die Startphase des Systems integriert werden kann. Im zurückliegenden Projektzeitraum wurde dieser Startup-Test definiert und die notwendigen Systemrandbedingungen definiert. Der Test garantiert eine stuck-at Fehlerüberdeckung von >90% bzw. >60% und erfüllt somit die geforderten Safety-Anforderungen.

Obwohl die ECC-Monitore SG2 und DF2 selbstprüfend ausgelegt sind<sup>7</sup> oder ausgelegt werden können, ist aus Safety-Gründen ein weiterer Test der Monitore

---

<sup>7</sup> Der ECC-Monitor SG2 ist so ausgelegt, dass durch Anlegen weniger Codeworte, wie sie laufenden Betrieb ohnehin mehrheitlich auftreten, 90% aller stuck-at-Fehler im ECC-Online-Checker am Ausgang dieses Monitors erkannt werden können. Bei der redundanten Fehlererkennungsschaltung

beim Start des Systems vorgesehen. Dieser Test soll die prinzipielle Funktion der ECC-Monitore sicherstellen. Ein solcher Test muss zum einen Hardware-Fehler in den kritischen Teilen der ECC-Monitore mit einer gewissen Fehlerüberdeckung erkennen und zum anderen die Anbindung der Monitore ans System erfassen. Die folgende Tabelle fasst die erreichte Testüberdeckung der einzelnen Bestandteile der ECC und der ECC-Monitore zusammen. Sowohl die permanente Überwachung im laufenden Betrieb als auch der Startup-Test erfüllt alle Safety-Anforderungen.

Tabelle 2: Testüberdeckung der einzelnen ECC-Blöcke

Block mit single point of failure	Fehlererkennung durch Block	Wann erfolgt die Fehlererkennung	Erreichte Fehlerüberdeckung
<b>Syndrome Generator</b>	ECC online checker	nahezu sofort	>99% (funktional)
<b>ECC (Fehlerkorrektur)</b>	ECC online checker	nahezu sofort	>99% (funktional)
<b>ECC online checker SG2 (ECC Monitor)</b>		Startup Test & laufender Betrieb	>90% (stuck-at)
<b>DF1 Fehlererkennung</b>	EDC online comperator (Verdopplung & Vergleich)	nahezu sofort	100% (funktional)
<b>DF2 redundante Fehlererkennung</b>	EDC online comperator (Verdopplung & Vergleich)	nahezu sofort	100% (funktional)
<b>Verbindung Fehlersignale – System</b>		Startup Test	100% (funktional)
<b>EDC Comperator</b>		Startup Test	>60% (stuck-at)

#### a. Testpattern zum Test des Online Checkers (Syndromgenerator SG2)

Wie bereits im vorherigen Abschnitt erwähnt ist der nachgeschaltete Syndromgenerator SG2 der ECC bereits selbstprüfend ausgelegt. Dies bedeutet, eventuelle Hardwarefehler werden bereits durch die Anwendung normaler Codeworte mit hoher Wahrscheinlichkeit erkannt. Natürlicher Weise sind die im Speicher abgelegten Daten stark benutzerabhängig und somit auch die angewendeten Testpattern im laufenden Betrieb. Obwohl bereits eine sehr geringe Anzahl von ausgewählten Testpattern ausreicht um eine Fehlerüberdeckung von >90% (stuck-at) zu erreichen, kann die effektive Fehlerüberdeckung auf Grund der Benutzerabhängigkeit nicht exakt bestimmt und garantiert werden.

Für den Startup-Test wurde deshalb vorgesehen, 32 dediziert Testvektoren (Codeworte) im Speicher zu hinterlegen. Diese 32 Codeworte garantieren eine Fehlerüberdeckung >90% (stuck-at) im Bezug auf die Hardware des ECC Online Checkers. Da Codeworte mit intakter ECC-Logik niemals ein „fail“ am Ausgang des Online Checkers erzeugen können, wird weiterhin mindestens ein Testvektor

---

(DF1) kann praktisch jedes funktionale Fehlverhalten des Monitors im laufenden Betrieb erkannt werden, sofern ein single-point-of-failure Model angenommen wird (Verdopplung und Vergleich).

benötigt, der ein kein gültiges Codewort darstellt. Dieser stellt sicher, dass auch eine fehlerhafte ECC Ausgabe korrekt durch den Online Checker erkannt und im System registriert wird. Aus Gründen der Zuverlässigkeit wurde vorgeschlagen, dieses Pattern redundant im Speicher zu hinterlegen.

#### **b. Testpattern zum Test der Fehlererkennungsschaltung (DF1, DF2), EDC Comperator und ECC Fehlersignale**

Die ECC Fehlererkennungsschaltung DF1 zur Berechnung der ECC-Fehlersignale ist mit der Schaltung DF2 redundant ausgelegt und wird konstruktionsbedingt im laufenden Betrieb überprüft. Während des Startup-Tests muss der Fokus deshalb auf die Überprüfung des EDC Comperators sowie auf die fehlerfreie Anbindung der Fehlersignale ans System gelegt werden.

Zur Überprüfung der ECC Fehlersignale und deren Systemanbindung wird für jedes Fehlersignal je ein dedizierter Testvektor abgespeichert. Diese Testvektoren sind natürlicher Weise Codeworte, die um einen bestimmten Bit-Fehler modifiziert wurden. Sofern es sich um einen linearen Code handelt, ist das verwendete Basiscodewort von untergeordneter Bedeutung. Da diese Testvektoren nicht durch die ECC korrigiert werden können (da die gespeicherten fehlerbehafteten Codeworte gewollt sind), sollten diese Patten zwingend redundant im Speicher abgelegt werden.

Wie bereits beim Online Checker dargestellt, so wird auch der EDC Comperator, eine fehlerfreie Hardware vorausgesetzt, nie eine Abweichung zwischen den Fehlererkennungsschaltungen DF1 und DF2 detektieren. Um diesen Fall stimulieren und den „fail“ Fall des EDC Comperators zu überprüfen, ist ein Test-Eingangssignal vorgesehen, der es erlaubt, die Ausgaben der Schaltung DF1 zu negieren. Dies führt zu einer garantierten Abweichung der Ausgaben von DF1 und DF2 und simuliert somit einen Fehler in einer der beiden Fehlererkennungsschaltungen. Die Untersuchungen im Projektzeitraum haben gezeigt, dass die oben genannten Testpattern im Verbindung mit dem zusätzlichen Test-Eingang zur Negierung der Ausgänge von DF1 ausreichend sind, um eine Fehlerüberdeckung von >60% (stuck-at, bezogen auf den EDC Comperator) zu erreichen. Somit wird der aus Safety-Sicht geforderte Wert für diesen Schaltungsteil von 60% Fehlerüberdeckung (stuck-at) übertroffen.

Neben den reinen Testpattern für den Startup-Test wurden auch die Systemrandbedingung und Testabläufe für einen solchen Test definiert. Diese Randbedingungen sind von zentraler Bedeutung um einen fehlerfreien und zuverlässigen Testablauf zu garantieren. Sie definieren beispielsweise die Zugriffe auf Speicher, Modi der ECC selbst, Abfolge der Testvektoren oder notwendige Gültigkeitschecks der Testvektoren zur Laufzeit des Startup-Tests.

#### **VIII.3.2. Zusammenfassende Bewertung**

Das vorgeschlagene Verfahren zur Überprüfung der ECC geht von einem der ECC nachgeschalteten Syndromgenerator aus, der überprüft, ob die durch die ECC korrigierten Wörter nach der Korrektur Codewörter sind.

Da die Anzahl der Codewörter wesentlich geringer als die Anzahl aller Wörter ist, ist die Wahrscheinlichkeit, einen Fehler zu erkennen, sehr groß, praktisch gleich 1.

Dadurch, dass der nachgeschaltete Syndromgenerator auch nur eine Teilmenge der Komponenten des Syndroms realisieren kann, das in dem fehlerkorrigierenden Code der ECC verwendet wird, ist ein bezüglich des erforderlichen Flächenaufwandes stufbares Vorgehen möglich. Die Anzahl der verwendeten Komponenten des Syndroms bestimmt exponentiell die Wahrscheinlichkeit, Fehler der ECC zu erkennen. Schon 7 Komponenten des Syndroms T entsprechen einer Fehlerüberdeckung von 99 %, und 10 Komponenten einer Fehlerüberdeckung von 99.9 %.

Die Fehleranzeigeschaltung DFA1 wird durch Verdopplung und Vergleich überprüft. Ein fehlerhaft nicht angezeigter 3-Bit Fehler, der zu einem Abbruch führen soll, wird indirekt überprüft. Ein nicht angezeigter 3-Bit Fehler wird mit hoher Wahrscheinlichkeit durch den nachgeschalteten Syndromgenerator SG2 als Nicht-Codewort erkannt. Ein fehlerhaft nicht angezeigter Adressenfehler wird ebenfalls indirekt durch den nachgeschalteten Syndromgenerator SG2 überprüft, und er wird ebenso mit hoher Wahrscheinlichkeit durch den nachgeschalteten Syndromgenerator als Nicht-Codewort erkannt.

Fehler im nachgeschalteten (zweiten) Syndromgenerator werden erkannt, wenn sie zu einem Syndrom ungleich 0, ..., 0 oder ungleich 1, ..., 1 dieses Syndromgenerators führen. Insbesondere sind die stuck-at-0 Fehler und die Stuck-at-1 Fehler auf den Ausgängen des zweiten Syndromgenerators SG2 im on-line Mode erkennbar.

In regelmäßigen Zeitintervallen, beim Einschalten, wird die ECC durch einen Selbsttest getestet. Vorteile der vorgeschlagenen Vorgehensweise sind

- der geringe, stufbare Hardwareaufwand schon für eine einzelne ECC.
- die Möglichkeit, einen einzigen (zweiten) zusätzlichen Syndromgenerator SG2 für mehrere ECCs auf einem Chip zu verwenden, wenn die korrigierten Signale der verschiedenen ECCs komponentenweise XOR-verknüpft werden.
- die sehr hohe Fehlerüberdeckung für beliebige Fehler.
- im Vergleich zu Verdopplung der ECC und Vergleich ist zunächst der Hardware-Aufwand erheblich geringer und der extrem große (selbstprüfende) und schwer zu testende Vergleich ist nicht erforderlich.
- im Vergleich zu einem praktisch ständig ablaufenden BIST sind die komplizierte Steuerung und die sicher schwer zu testenden Multiplexer nicht erforderlich.

Ferner kann die Funktionalität der ECC-Monitore (Syndromgenerator SG2 sowie redundante Fehlererkennungsschaltung DF1/2) durch einen kurzen Startup-Test effizient nachgewiesen werden. Die hierfür notwendigen Testpattern wurden bestimmt und die Systemrandbedingungen definiert. Es konnte gezeigt werden, dass sich ein solcher Startup-Test mit sehr wenigen abgespeicherten Testpattern realisieren lässt ohne die Safety-Anforderungen zu verletzen. Weiterhin konnte gezeigt werden, dass sich insbesondere der Test des nachgeschalteten Syndromgenerators SG2 nahezu vollständig auf Basis von Codeworten der ECC realisieren lässt. Hierdurch kann auf ein redundantes Speichern der Testmuster verzichtet werden.

#### VIII.4. Ergebnisse zur Aufgabe 2.3 - Test und Diagnose digitaler Logik in der Applikation unter Verwendung von On-Chip Produktionstestlogik und Standard-Hochgeschwindigkeits-Schnittstellen

Diese Aufgabe umfasst zwei Aspekte, welche jeweils das Ziel haben die digitale Logik einzelner Halbleiterkomponenten im Gesamtsystem testen zu können und eine weiterführende Diagnose zu ermöglichen.

Mit Hilfe von Selbsttestmaßnahmen basierend auf pseudo-zufälligen Testmustern soll ein struktureller Test im System durchgeführt werden. Hierbei soll soweit wie möglich die Scantest-Architektur des produktiven Tests wiederverwendet werden. Da bekannt ist, dass die Testabdeckung basierend auf pseudo-zufälligen Testmustern in der Regel deutlich geringer ist als mit deterministischen Tests, sollen zusätzliche DFT-Maßnahmen erarbeitet werden, welche die Testbarkeit der Schaltung erhöhen, um so eine ausreichende Testqualität des Selbsttests zu ermöglichen. Die zu entwickelnde Selbstteststruktur soll hierbei mit einem möglichst geringen zusätzlichen Flächenbedarf auskommen.

Desweiteren soll eine Möglichkeit geschaffen werden, die es erlaubt externe Testdaten dem zu testenden Halbleiterbauelement im System über funktionale Hochgeschwindigkeitsschnittstellen zuzuführen. Hierzu sollen verschiedene Arten von Schnittstellen auf ihre Verfügbarkeit im Gesamtsystem und ihre Eignung für den Testdatenaustausch untersucht werden. Für die geeigneten Schnittstellen sollen spezielle Schaltungsblöcke entwickelt werden, die eine Ankoppelung eines externen Testdatenstroms an die interne Scantest-Architektur erlauben.

Die entwickelte Selbsttest-Architektur und die Anbindung der Testdaten über eine Hochgeschwindigkeits-Schnittstelle sollen implementiert und an einer realen Schaltung validiert werden.

##### VIII.4.1. Selbsttest unter Ausnutzung der Scantest-Architektur

Nach eingehender Untersuchung verschiedener möglicher Selbsttestlösungen wurde primär aus Flächengründen ein Ansatz ausgewählt, der weitestgehend die bestehende Scantest-Architektur des Produktionstests ausnutzt und nur sehr wenig zusätzliche Komponenten benötigt. Neben den verfügbaren Scanketten kann auch die sich auf dem Baustein befindende Logik zur Kompression der Testdaten für den Selbsttest genutzt werden. Die Dekompressor-Schaltung, welche beim verwendeten Testdatenkompressionsverfahren „Embedded Deterministic Test“ (EDT) die vom Tester kommenden komprimierten Testdaten auf die internen Scanketten verteilt, kann beim Selbsttest als Testmustergenerator verwendet werden. Die Struktur des EDT-Dekompressors ähnelt einem LFSR<sup>8</sup>. Für den Selbsttestbetrieb können die Scan-Eingangskanäle konstant gehalten werden und der EDT-Dekompressor liefert durch Anlegen des Taktes einen pseudo-zufälligen Datenstrom, der über ein enthaltenes XOR-basiertes Netzwerk linear unabhängig auf die internen Scanketten verteilt wird.

---

<sup>8</sup> Linear Feedback Shift Register

Neben einem Signaturregister (MISR) zur Kompression der Testantworten muss die Schaltung für den Selbsttest lediglich durch einen BIST-Controller erweitert werden, der die für den Test notwendigen Steuersignale auf dem Baustein bereitstellen kann.

Die prinzipielle Struktur der gewählten Selbsttest-Architektur ist in Abbildung VIII-25 dargestellt. Der obere Teil zeigt die ohnehin existierende Scantest-Architektur des Produktionstests. Der untere Teil stellt die Erweiterungen für den Selbsttest im System dar. Der Flächenbedarf für das zusätzliche Signaturregister und den BIST-Controller ist hierbei gering. Die Funktion des für den Selbsttest entwickelten BIST-Controllers wird in Kapitel VIII.4.3 genauer beschrieben.

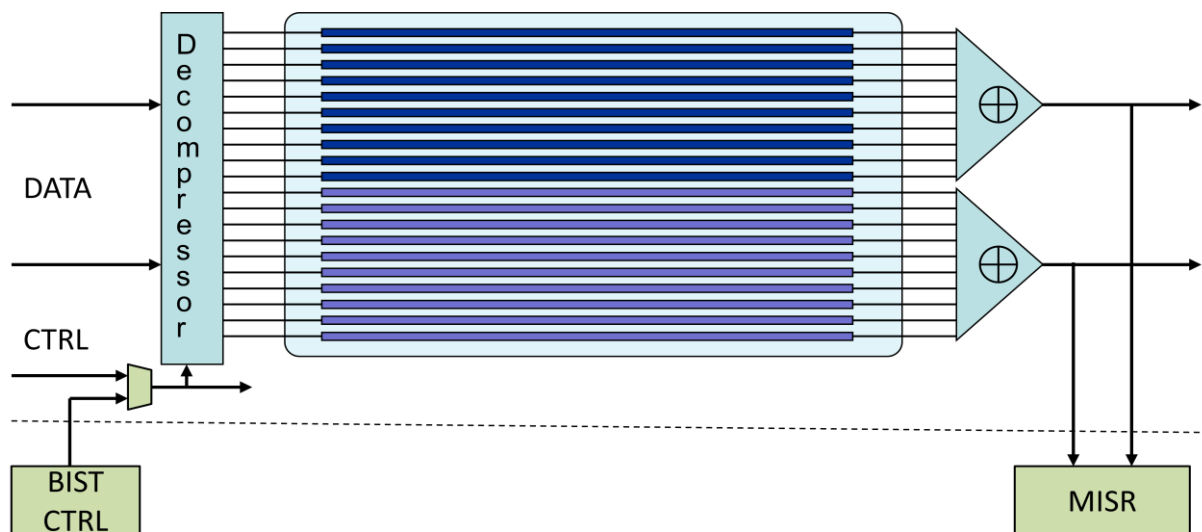


Abbildung VIII-25: Prinzipielle Struktur der Selbsttest-Architektur

#### VIII.4.2. Analyse der Testbarkeit beim Selbsttest mit pseudo-zufälligen Testmustern

Die erreichbare Testabdeckung bei Verwendung von pseudo-zufälligen Testmustern ist in der Regel deutlich geringer als die, welche mit deterministisch berechneten Tests erzielt werden kann. Für die Schaltung eines typischen Microcontrollers wurden umfangreiche Analysen zur Testbarkeit mit pseudo-zufälligen Testmustern durchgeführt und Design-Maßnahmen untersucht, um die Testabdeckung zu erhöhen.

Die Problematik mit pseudo-zufälligen Testmustern bzw. der Unterschied zu berechneten Scan-Tests kann einfach an einem UND-Gatter mit  $n$  Eingängen gezeigt werden. Zur Vereinfachung wird angenommen, dass alle Eingänge dieses UND-Gatters direkt aus Scan-Registern kontrolliert werden. Um eine logische „1“ am Ausgang zu erzeugen, müssen sämtliche Eingänge ebenfalls den Wert „1“ haben. Bei einem berechneten Testmuster ist dies in der Regel kein Problem und die Scankette wird einfach mit der entsprechenden Anzahl von „1“-Werten geladen. Testmustergeneratoren für pseudo-zufällige Tests liefern stets „0“- und „1“-Werte mit gleicher Wahrscheinlichkeit. Das heißt jedes Scan-Register wird jeweils mit einer Wahrscheinlichkeit von 50% mit einer „0“ bzw. „1“ geladen. Die



Wahrscheinlichkeit, dass am Ausgang des UND-Gatters eine „1“ entsteht, reduziert sich somit mit der Anzahl der Eingänge und beträgt  $0,5^n$ . Bei einem UND-Gatter mit 10 Eingängen ergibt sich somit statistisch nur alle 1024 Testmuster eine „1“ am Ausgang. Bei noch mehr Eingängen wird es sehr unwahrscheinlich, dass sich die „1“ überhaupt jemals während des Tests einstellt. Das gleiche Problem gibt es entsprechend auch für ODER-Gatter, bei denen die „0“ am Ausgang unwahrscheinlich wird.

Große UND- bzw. ODER-Strukturen sind in typischen Designs keine Seltenheit. Sie treten immer auf wenn viele Bedingungen gleichzeitig erfüllt sein müssen, um ein Ereignis auszulösen. Diese großen UND- bzw. ODER-Strukturen werden bei der Synthese in der Regel als Baum aus Einzelgattern aufgebaut, da die zur Verfügung stehenden Standardzellen nur eine begrenzte Anzahl von Eingängen haben.

Um die Testbarkeit solcher Strukturen zu verbessern, wurden spezielle VHDL-Funktionen entwickelt, welche die Logik-Bäume bereits im RTL-Code automatisch in kleinere Segmente aufteilen. Die Ausgänge der Segmente werden hierbei über einen XOR-Baum auf einen gemeinsamen Beobachtungspunkt geführt. Durch Testmultiplexer wird die Kontrollierbarkeit der nächsten Baumstufe sichergestellt. Abbildung VIII-26 zeigt einen solchen Logik-Baum mit den zusätzlichen Testmaßnahmen (blau) für eine Partitionierung mit jeweils 8 Eingängen pro Segment. Statistisch lässt sich diese Struktur nun mit 256 Zufallsmustern komplett testen.

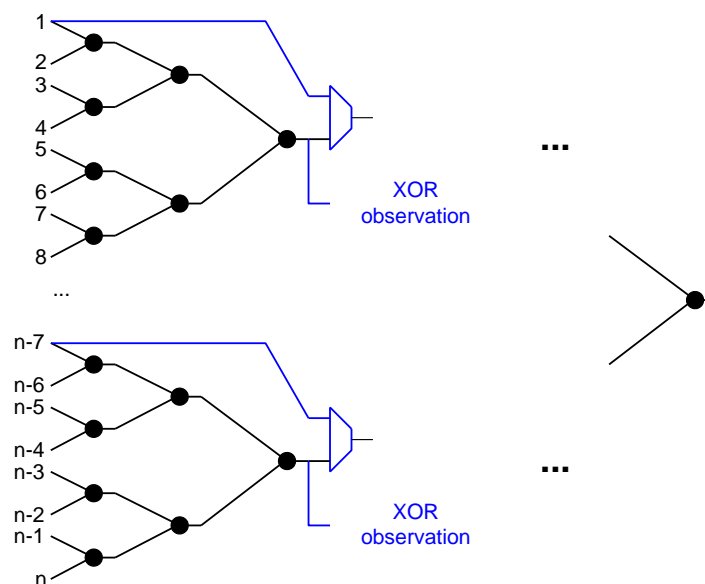


Abbildung VIII-26: Logik-Bäume mit Maßnahmen zur Verbesserung der Testbarkeit

Ähnliche VHDL-Funktionen wurden auch für den Vergleich (kleiner, größer) von breiten Bussen entwickelt. Auch hier wurde die Logik in Segmente aufgeteilt und die Teilergebnisse beobachtbar gemacht. Somit kann auch die Vergleichslogik für die niederwertigen Bits mit pseudo-zufälligen Testmustern getestet werden, obwohl in der Regel schon die höherwertigen Bits einen Unterschied aufweisen.

Als weitere Maßnahme, um die Testbarkeit mit pseudo-zufälligen Testmustern zu erhöhen, wurden die Steuersignale sämtlicher Clock-Gating-Zellen von einer zentralen Quelle kontrollierbar gemacht, so dass während des Selbsttest der Takt meistens an den Registern ankommt. Ohne diese Maßnahme wurde beobachtet, dass der Takt aufgrund komplexer Enable-Bedingungen und mehrstufigen Clock-Gatings nur selten für den Test aktiv ist.

Für den untersuchten Microcontroller wurde der RTL-Code geändert und die neu entwickelten, besser testbaren Funktionen verwendet. Durch diese Maßnahmen konnte die Testabdeckung für 10000 Zufallstestmuster von 76,0% auf 94,5% gesteigert werden (siehe Abbildung VIII-27).

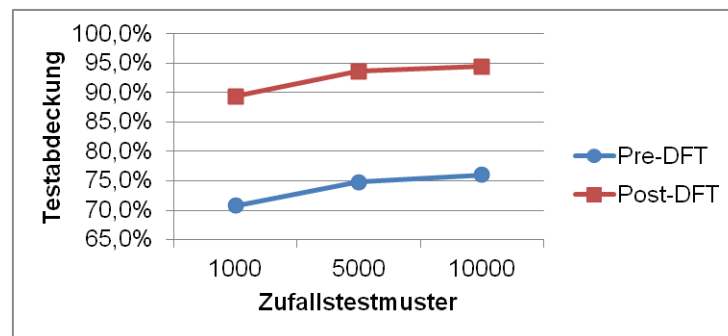


Abbildung VIII-27: Verbesserung der Zufallstestbarkeit durch RTL-Änderungen

### VIII.4.3. Scan-Controller

Bei der Durchführung des Produktionstests werden alle notwendigen Steuersignale des Scantests vom Tester zur Verfügung gestellt. Da beim Selbsttest aber kein Testautomat zur Verfügung steht, müssen sämtliche für den Testablauf notwendigen Steuersignale auf dem zu testenden Baustein selbst erzeugt werden. Hierzu dient der in dieser Aufgabe entwickelte Scan-Controller, welcher als technologieunabhängiger VHDL-Code realisiert wurde.

#### VIII.4.3.1. Schnittstelle des Scan-Controllers

Zu den bereitzustellenden Steuersignalen zählt im Wesentlichen das sogenannte „Scan-Enable“-Signal, welches zwischen dem seriellen Laden bzw. Entladen der Prüfpfade und dem funktionalen Betrieb der Register umschaltet. Der als Testmustergenerator verwendete EDT-Dekompressor benötigt eigene Steuersignale für die Initialisierung. Auch das Signaturregister, das die Testdaten komprimiert, erfordert ein Steuersignal, welches die Signaturbildung kontrolliert.

Neben den Steuersignalen muss der Scan-Controller auch die verschiedenen Takt-Signale für den Testmustergenerator und den Betrieb der Prüfpfade bereitstellen. Diese Takte werden hierbei aus dem für den Selbsttest verwendeten Systemtakt abgeleitet, welcher in der Regel von einer PLL oder einem Oszillator erzeugt wird. Der Scan-Controller liefert zusätzlich zwei Statussignale als Ausgang, die das Ende des Tests bzw. den Abschluss der Initialisierungsphase anzeigen.

Eine kurze Übersicht der vom Scan-Controller bereitgestellten Steuersignale ist in Tabelle VIII.3 dargestellt.

Tabelle VIII.3: Vom Scan-Controller bereitgestellte Steuersignale

scan_enable_o	Scan-Enable-Signal zum Umschalten zwischen Shift- und Capture-Phase
scan_clk_o [lbist_scan_clocks_c:1]	Taktsignal(e) für die Prüfpfade
misr_compress_en_o	Steuersignal für das Signaturregister
edt_clk_o	Taktsignal für den Testmustergenerator (EDT-Block)
edt_update_o	Steuersignal zur Initialisierung des Testmustergenerators
edt_channel_one_o	Steuersignal zur Initialisierung des Testmustergenerators
mask_init_o	Statussignal signalisiert die Initialisierungsphase des Testmustergenerators
lbist_done_o	Statussignal signalisiert das Ende des Selbsttests

Für die Konfiguration des Scan-Controllers müssen einige Konstanten und Generics im entwickelten VHDL-Code definiert werden. Zusätzlich sind einige Signale zur Ansteuerung notwendig.

Mit den Konstanten werden grundsätzliche Parameter des Scan-Controllers konfiguriert. Dies sind die Anzahl der vorgesehenen Taktzyklen in der Capture-Phase, die Anzahl der zu erzeugenden Scan-Taktsignale und die Busbreite des Steuersignals zur Konfiguration der Testmusteranzahl. Zur Konfiguration des Scan-Controllers ist es ebenfalls nötig die Länge der längsten internen Scan-Kette und die Länge des EDT-Maskierungsregisters zu spezifizieren. Anstatt Konstanten werden hierfür Generics verwendet, um den Einsatz von mehreren Scan-Controllern mit verschiedenen Werten in einer Schaltung zu ermöglichen. Dies ist dann relevant wenn verschiedene Schaltungsteile hierarchisch mit einem eigenen Selbsttest getestet werden sollen (siehe Kapitel VIII.4.5).

Neben den typischen Steuersignalen wie Takt und Reset gibt es noch weitere Steuersignale, die eine flexible Konfiguration im Betrieb ermöglichen. So lässt sich das Taktschema in der Capture-Phase, die Anzahl der zu testenden Testmuster und ein spezieller Modus zur Reduktion der Aktivität beim Betrieb der Prüfpfade (siehe Kapitel VIII.4.3.3 ) einstellen.

#### **VIII.4.3.2. Beschreibung der Funktionalität**

Der Kern des Scan-Controllers besteht aus einem Zustandsautomaten, dessen Zustandsübergangsdiagramm in Abbildung VIII-28 dargestellt ist. Ausgehend aus dem Ruhezustand wird der Scan-Controller mit dem Steuersignal „lbist\_mode\_i“ gestartet. Danach folgen drei Zustände, die zur Initialisierung des EDT-Maskierungsregisters dienen. Anschließend wird der eigentliche Selbsttest gestartet. Jedes Testmuster des Selbsttests muss die Prüfpfade laden bzw. entladen und den eigentlichen Test in der Capture-Phase durchführen. Jedes dieser Testmuster wird in vier Phasen durchlaufen: „load\_unload“, „shift“, „shift\_idle“ und „capture“. Während „shift“ werden die Prüfpfade seriell geladen bzw. entladen. In „capture“ erfolgen die eigentlichen Testzyklen. Die Phasen „load\_unload“ und „shift\_idle“ befinden sich

jeweils zwischen „shift“ und „capture“ und dienen lediglich dazu, den zeitlichen Ablauf zu entspannen. Diese vier Zustände werden so oft wiederholt, bis der Testmusterzähler die gewünschte Anzahl von Testmustern erreicht hat. Dann ist das Ende des Selbsttests erreicht.

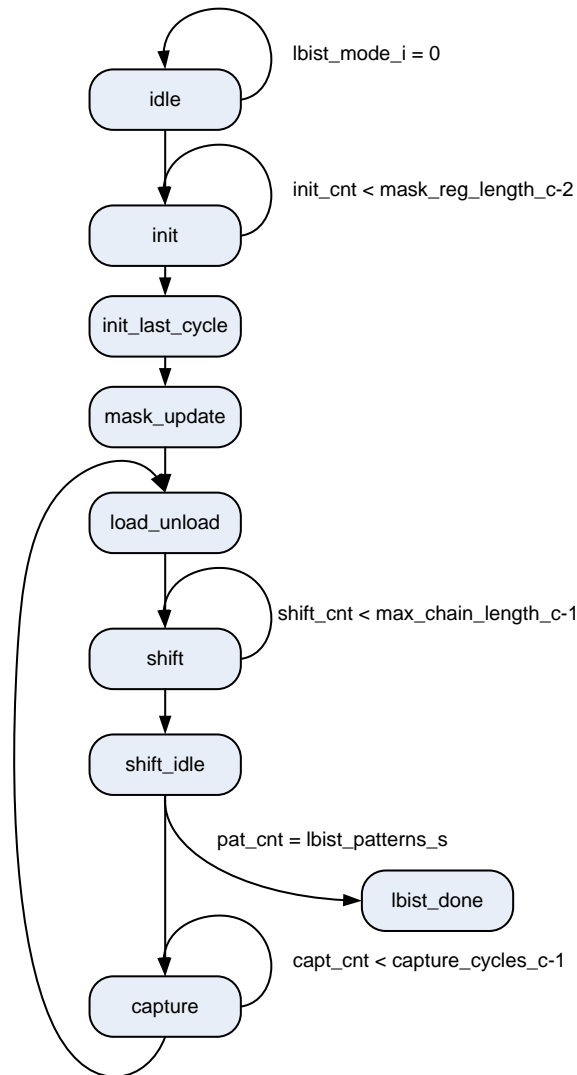


Abbildung VIII-28: Zustandsübergangsdiagramm des Scan-Controllers

### Initialisierungsphase

Die EDT-Logik zur Testdatenkompression verwendet ein Maskierungsregister, das es beim normalen Produktionstest erlaubt bei Bedarf einzelne interne Scan-Ketten auszublenden. Diese Funktionalität kann beim Selbsttest nicht verwendet werden. Daher ist es notwendig das Maskierungsregister vor dem Start des Selbsttests so zu konfigurieren, dass alle internen Scan-Ketten unmaskiert am Signaturregister beobachtbar sind. Das Maskierungsregister muss hierzu komplett mit dem Wert „1“ gefüllt werden. Lediglich das letzte Bit, das seriell über den ersten EDT-Kanal geladen wird, muss den Wert „0“ annehmen. Anschließend muss mit dem „EDT-Update“-Signal der geladene Wert des Maskierungsregisters aktiviert werden.

## Capture Phase

Die Länge der Capture-Phase ist durch die Konstante „capture\_cycles\_c“ definiert. Das für den Test verwendete Taktschema wird durch den Steuereingang „capture\_config\_i“ konfiguriert. Speziell wenn mehrere Gruppen von Prüfpfaden mit unterschiedlichen Takten betrieben werden sollen, kann so der Ablauf in der Capture-Phase gesteuert werden. In Abbildung VIII-29 ist den Signalverlauf für den Fall dargestellt, dass zwei verschiedene Scan-Takte nacheinander in der Capture-Phase erzeugt werden sollen.

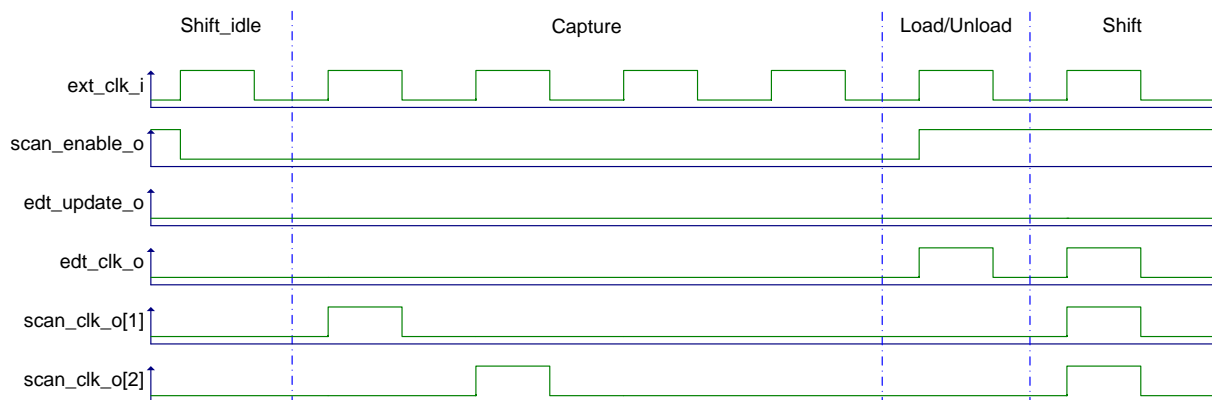


Abbildung VIII-29: Signalverlauf der Capture-Phase: lbist\_scan\_clocks\_c = 2, lbist\_capture\_cycles\_c = 4, capture\_config\_i = („0010“, „0001“)

## Shift-Phase und Signaturbildung

Das Signal zur Steuerung des Signaturregisters „misr\_compress\_en\_o“ ist nur während der Shift-Phase aktiv. Das heißt nur während des Entladens der Scan-Ketten werden die Daten im Signaturregister komprimiert. Die Werte während der Capture-Phase werden nicht berücksichtigt. Das erste Entladen der Scan-Ketten wird nicht bewertet, da hier die Scan-Ketten eventuell noch nicht vollständig initialisiert sind. Nachdem der Testmusterzähler seinen Zielwert erreicht hat, erfolgt ein letztes Entladen der Scan-Ketten. Anschließend erreicht der Scan-Controller den Zustand „lbist\_done“ und das Ende des Tests wird am Ausgang „lbist\_done\_o“ angezeigt. Das Signaturregister behält hierbei seinen endgültigen Wert.

### VIII.4.3.3. Maßnahmen zur Reduktion der Schaltungsaktivität

Die Schiebephase des auf Prüfpfaden basierenden strukturellen Tests ist oft der kritische Fall bezüglich der Leistungsaufnahme. Bereits beim produktiven Test führt dies vermehrt zu Problemen. Es wird erwartet, dass die Situation beim Selbsttest im System noch deutlich kritischer ist. Die Spannungsversorgung erfolgt hier nicht mehr vom sehr leistungsfähigen Tester. Anstatt dessen muss der Test mit der funktionalen Versorgung des Systems betrieben werden, die in der Regel deutlich geringer dimensioniert ist. Zusätzlich führt der Test mit pseudo-zufälligen Testdaten dazu, dass statistisch rund 50% der Scan-Register in jedem Schiebezyklus ihren Wert ändern.

Eine Maßnahme, die oft beim produktiven Test angewendet wird, um die gleichzeitige Schaltungsaktivität zu reduzieren, ist es die Scan-Ketten in verschiedene Gruppen mit verschiedenen Takten einzuteilen und diese leicht versetzt anzulegen (siehe Abbildung VIII-30). Dadurch, dass nicht mehr alle Register zum exakt gleichen Zeitpunkt schalten, kann in vielen Fällen ein kritischer Einbruch der Versorgungsspannung verhindert werden.

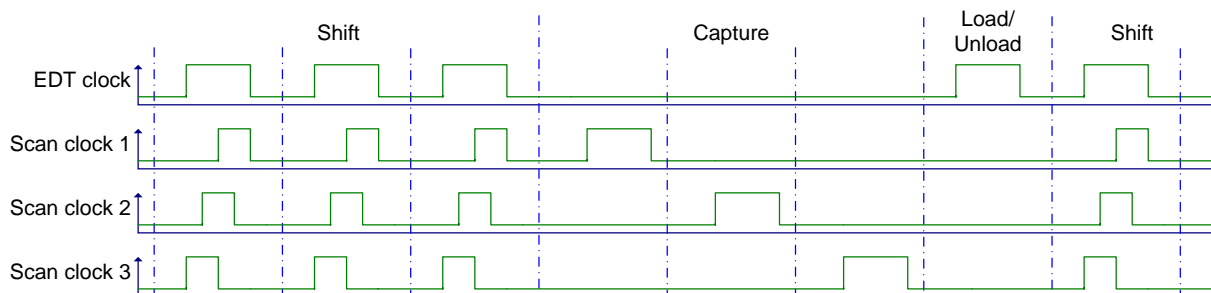


Abbildung VIII-30: Prinzip von verzögerten Scan-Takten, um die gleichzeitige Aktivität beim Schieben der Scan-Ketten zu reduzieren

Beim Selbsttest im System ist dieses Verschieben der Takte gegeneinander nicht so leicht möglich, da die Takte nicht frei vom Tester angesteuert werden können, sondern von der verwendeten Taktquelle des Systems abgeleitet werden müssen.

Der entwickelte Scan-Controller enthält eine Option, die es ebenfalls ermöglicht mehrere Scan-Gruppen nacheinander in der Schiebephase zu takten. Anstatt die Takte gegeneinander zu verschieben, kann der Scan-Controller mehrere Taktpulse pro Schiebecycle liefern. Mit einer zusätzlichen Logik, welche für jede Scan-Gruppe jeweils nur einen der Pulse weitergibt, kann ein sequentielles Taktschema erreicht werden. In Abbildung VIII-31 ist der Signalverlauf für ein Beispiel dargestellt, bei dem in jedem Schiebecycle drei Takte bereitgestellt werden.

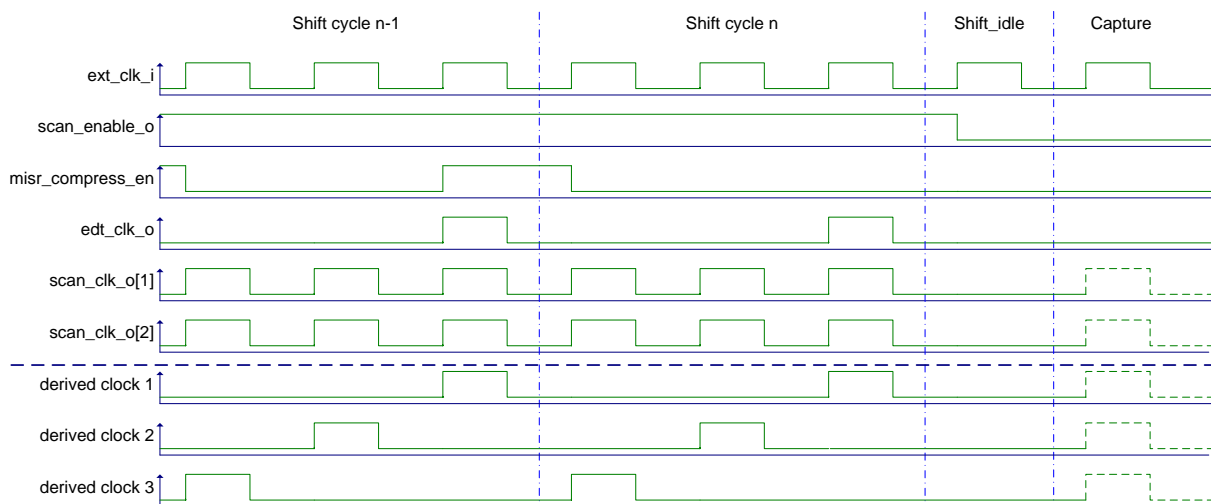


Abbildung VIII-31: Signalverlauf bei Aktivierung der „Split Shift“-Option (split\_shift\_cycles\_j = "010" resultierend in 2 zusätzlichen Takten pro Shift-Zyklus)

#### VIII.4.4. Maßnahmen zur Reduktion der Ausführungszeit des Selbsttests

Die Länge der verwendeten Scanketten ist ein wesentlicher Faktor, der die Laufzeit des Selbsttests bestimmt. Beim Produktionstest ist die Anzahl der Scanketten durch die nutzbaren Pins des Bausteins oder durch die verfügbaren Testerkanäle beschränkt. Selbst mit Testdatenkompression, die es erlaubt mehrere interne Scanketten durch ein reduziertes Testinterface zu betreiben, kann die Anzahl der Scanketten nicht beliebig erhöht werden, da sich zu hohe Kompressionsraten negativ auf die Testmustergenerierung auswirken. Bei dem verwendeten Selbsttest hingegen ist die Anzahl der Scanketten nicht limitiert. Durch die Verwendung von mehr und somit kürzeren Scanketten kann die Testdauer des Selbsttests reduziert werden.

Es wurde eine Erweiterung der Scantest-Architektur untersucht, bei der sich die Konfiguration der Scanketten für den Selbsttest und den produktiven Test unterscheidet. Für den Selbsttest wird eine um Faktoren größere Anzahl von Ketten implementiert als dies für den Produktionstest gewünscht ist. Für den produktiven Test werden diese kurzen LBIST-Ketten wieder aneinandergelängt, um die ursprünglich geplante Anzahl zu erhalten.

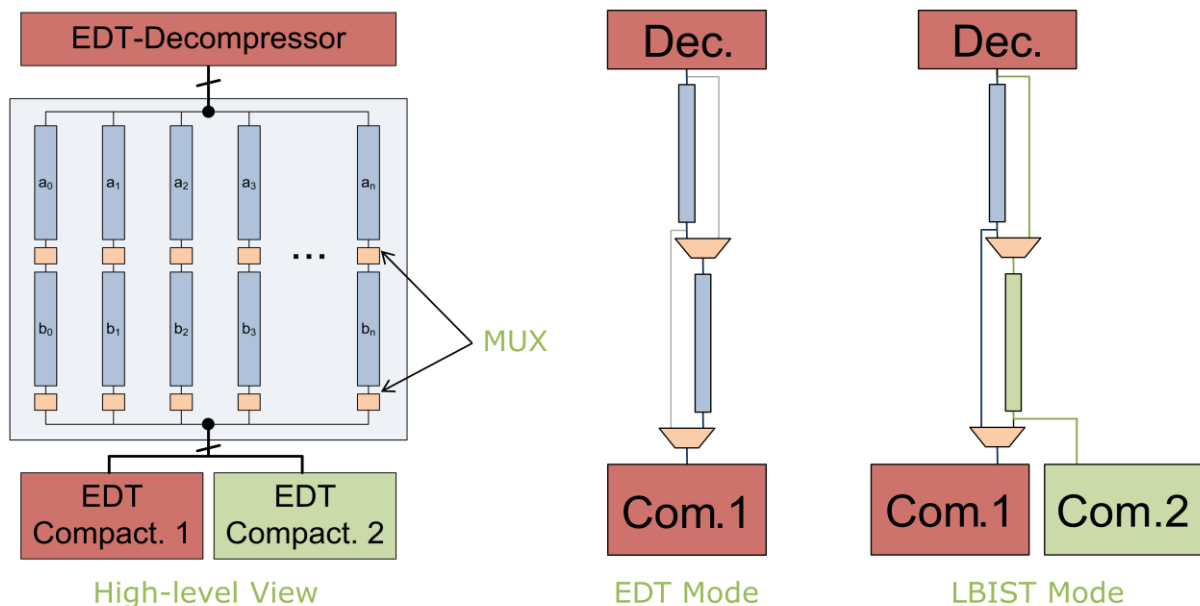


Abbildung VIII-32: Prinzip der konfigurierbaren Scantest-Architektur zur Reduktion der Ausführungsdauer des Selbsttests

In Abbildung VIII-32 ist auf der linken Seite das Prinzip der konfigurierbaren Scantest-Architektur dargestellt. Die Scanketten sind jeweils in zwei Segmente unterteilt, die über Multiplexer konfiguriert werden können. Beim produktiven Test (siehe Abbildung VIII-32 mittig) werden die Segmente zu einer Kette verbunden. Die Anzahl der Scanketten bzw. deren resultierende Länge entspricht der gewünschten produktiven Konfiguration basierend auf der gewählten Kompressionsrate. Beim Selbsttest (siehe Abbildung VIII-32 rechts) werden die beiden Kettensegmente

parallel vom EDT-Dekompressor stimuliert. Hierdurch verdoppelt sich die Anzahl der Ketten und die Länge wird halbiert. Für die Beobachtung der neu gewonnenen Scanketten wird die Logik des EDT-Kompaktors aufgedoppelt. Im LBIST-Mode entspricht die resultierende Architektur somit zwei identischen EDT-Partitionen, die parallel betrieben werden. Da aus Gründen der Flächenoptimierung ein gemeinsamer EDT-Dekompressor für beide Partitionen verwendet wird, werden jeweils zwei Kettensegmente stets mit identischen Werten geladen. Diese Abhängigkeit der Testdaten kann prinzipiell Auswirkungen auf die erzielbare Testabdeckung haben. Um diesen Einfluss zu minimieren, sollten die gemeinsam stimulierten Scansegmente möglichst zu unabhängigen Bereichen der Schaltung gehören. In Abbildung VIII-32 ist das Prinzip der konfigurierbaren Scantest-Architektur mit lediglich zwei Segmenten pro Scankette dargestellt. Die Scanketten können aber auch in mehrere Segmente unterteilt werden, um die Testdauer des Selbsttests weiter zu reduzieren.

Für eine Beispielschaltung wurde dieser Ansatz angewendet und auf seine Durchführbarkeit im Design-Ablauf hin geprüft. Der Einbau der Scanketten und der notwendigen Multiplexer für die zwei Modi kann weitestgehend im bestehenden Design-Flow automatisiert werden. Moderne Synthese-Werkzeuge unterstützen den Einbau der Scanketten für verschiedene Modi. Das heißt die Multiplexer brauchen nicht im RTL-Code beschrieben werden, sondern können automatisch bei der Synthese eingebaut werden. Lediglich die Erweiterung der EDT-Logik um die zusätzlichen EDT-Kompaktoren muss manuell im RTL-Code erfolgen. Der automatisch generierte VHDL-Code des Kompaktors wird hierbei einfach mehrfach instanziiert. Da beim Selbsttest keine Scanketten maskiert werden, brauchen die zusätzlichen Kompaktoren nicht von der EDT-Logik kontrolliert werden. Die vorhandenen Eingänge zur Ansteuerung der Maskierungslogik werden auf konstant inaktiv gelegt. Bei der Synthese kann die nicht genutzte Maskierungslogik in den zusätzlichen EDT-Kompaktoren wegoptimiert werden, um Fläche zu sparen. Die lediglich für den Selbsttest genutzten EDT-Kompaktoren reduzieren sich somit auf einfache XOR-Bäume. Für die Fehlersimulation und Bestimmung der Signatur sind lediglich einige Anpassungen nötig, um die zusätzlichen EDT-Module zu beschreiben. Dass hierbei alle EDT-Module gemeinsam den gleichen EDT-Dekompressor verwenden, führt zu keinen Problemen.

Die Wirksamkeit des neuen Ansatzes wurde für die Schaltung eines typischen Microcontrollers genauer untersucht. Das neue Verfahren wurde mit zwei bis sechs Segmenten pro Kette angewendet und mit der ursprünglichen Scankonfiguration verglichen. Die verschiedenen betrachteten Konfigurationen sind in Tabelle VIII.4 dargestellt. Für den produktiven Test werden immer 100 interne Scanketten und 10 externe Scankanäle verwendet, so dass stets ein unveränderter Kompressionsfaktor von 10 resultiert. Für die verschiedenen Implementierungen  $X_i$  werden die Scanketten beim Selbsttest in  $i$  kürzere Segmente unterteilt. Die Länge der längsten Scankette reduziert sich beim Selbsttest somit um den Faktor  $i$ . Zum Beispiel kann durch die Aufteilung jeder Kette in 6 Segmente die Kettenlänge im LBIST-Betrieb von 244 auf 41 reduziert werden.

Tabelle VIII.4: Scankonfigurationen mit resultierender Scankettenlänge

Version 1.1	Bericht	16. Oktober 2013
-------------	---------	------------------



	Mode	X1	X2	X3	X4	X6
<b>Scanketten</b>	LBIST	100	200	300	400	600
	EDT	100				
<b>Scankanäle</b>	EDT	10				
<b>Längste Scankette</b>	LBIST	244	122	82	61	41
	EDT	244				

Die erreichte Testabdeckung und die resultierende Selbsttestdauer bei Annahme eines 20MHz Taktes sind in

Tabelle VIII.5 dargestellt. Eine Erhöhung der Kettenanzahl führt für eine gegebene Anzahl von Testmustern zu einer leichten Reduktion der Testabdeckung. Aufgrund der deutlich geringeren Kettelänge kann allerdings ein deutlicher Anstieg der Testabdeckung bei gleicher Testdauer beobachtet werden.

Tabelle VIII.5: Testabdeckung und Testzeit für verschiedene Scankonfigurationen

	Testmuster	X1	X2	X3	X4	X6
<b>Testabdeckung (LBIST)</b>	1000	88,99	89,10	88,94	88,98	88,73
	5000	93,69	93,65	93,47	93,37	93,32
	10000	94,52	94,46	94,27	94,25	94,17
	100000	95,90	95,82	95,74	95,66	95,57
<b>Testdauer [ms] (LBIST)</b>	1000	12,20	6,10	4,10	3,05	2,05
	5000	61,00	30,50	20,50	15,25	10,25
	10000	122,00	61,00	41,00	30,50	20,50
	100000	1220,00	610,00	410,00	305,00	205,00

#### VIII.4.5. Hierarchische Selbsttestlösungen

Neben dem Ansatz die gesamte digitale Logik mit einem einzelnen Selbsttest zu versehen, wurden auch hierarchische Selbsttestlösungen untersucht, bei denen die Logik in einzelne Bereiche partitioniert wird, welche dann eigenständig getestet werden können. Der hierarchische Test bietet den Vorteil, dass eine direkte Schlussfolgerung über die fehlerhafte Hierarchie möglich ist. Speziell bei sicherheitsrelevanten Systemen ist dies von Vorteil, da so zwischen Fehlern in mehr oder weniger kritischen Schaltungsteilen unterschieden werden kann. Dies kann bei der Bewertung des Fehlers und den notwendigen Folgeaktionen relevant sein.

Eine Grundvoraussetzung für den hierarchischen Selbsttest ist die Partitionierung der Scantest-Architektur. Jede einzeln getestete Hierarchie benötigt eigenständige Scanketten und einen eigenen EDT-Block. Dies kann für den produktiven Test mit geringer Anzahl von Testpins zu Restriktionen führen, da jede Hierarchie ein eigenes Testinterface benötigt. Wenn nicht genug Pins zur Verfügung stehen, müssen beim

Produktionstest die einzelnen Bereiche sequentiell getestet werden. Da beim Selbsttest keine externen Pins verwendet werden, können hier weiterhin alle Hierarchien gleichzeitig getestet werden, solange keine anderen Randbedingungen wie z.B. die Leistungsaufnahme dagegen sprechen. Für den Selbsttest gibt es keine speziellen Anforderungen an die Scankettenlängen der einzelnen Testhierarchien. Für den produktiven Test hingegen sollte darauf geachtet werden, dass alle Module, die parallel getestet werden sollen, eine ähnliche Kettenlänge aufweisen, da die längste Kette die Testdauer bestimmt.

Für den hierarchischen Test müssen die einzelnen Hierarchien gegeneinander isoliert werden, um Wechselwirkungen zwischen die Testpartitionen auszuschließen und unabhängige Tests zu gewährleisten.

#### **VIII.4.6. Ansteuerung und Auswertung des Selbsttests im System**

Der Kapitel VIII.4.3 beschriebene Scan-Controller ist so generisch und unabhängig von der zu testenden Schaltung, dass er prinzipiell in jeder Schaltung zur Ansteuerung der Scan-Ketten beim Selbsttest verwendet werden kann. Neben diesem universellen Scan-Controller ist stets auch noch ein LBIST-Controller notwendig, der sich um das Starten und Auswerten des Selbsttests im System kümmert. Dieser LBIST-Controller hängt stark vom Konzept des Selbsttests ab und muss für jedes Produkt individuell entwickelt werden.

Eine wesentliche Fragestellung ist wie und wann der Selbsttest gestartet werden soll. Zum einen ist es möglich den Test von außen vom System zu starten. Hierzu bieten sich typische Diagnoseschnittstellen wie zum Beispiel JTAG an. Die Ansteuerung kann aber auch mit Hilfe des speziell entwickelten Interfaces (siehe Kapitel VIII.4.7) über funktionale Schnittstellen wie CAN oder FlexRay erfolgen. Das Ziel eines solchen von extern kontrollierten Selbsttests kann es zum Beispiel sein, fehlerhafte Komponenten in der Werkstatt zu identifizieren. Zum anderen ist es besonders für sicherheitsrelevante Komponenten oft gefordert, dass sich diese beim Einschalten oder periodisch während des Betriebs selbst überprüfen. Hier muss die Ansteuerung des Selbsttests in die Startsequenz des Bausteins integriert werden und meist unabhängig vom umgebenen System erfolgen. Ähnliches gilt für die Auswertung des Testergebnisses. Über eine Diagnose- oder funktionale Schnittstelle kann das Testergebnis nach außen ans System übergeben werden. Es ist ebenfalls möglich, dass das Testergebnis in speziellen Registern abgelegt wird, die dann per Software ausgewertet werden. Allgemein stellt sich die Frage wie mit einem Ausfall umgegangen wird, welcher zum Beispiel beim Systemstart erkannt wird. Der Startvorgang könnte komplett abgebrochen werden, oder der Ausfall könnte lediglich zur weiteren Behandlung an das umgegebene System gemeldet werden. Unabhängig von der gewählten Methode zum Starten und Auswerten des Selbsttests, gibt es eine Vielzahl von Detail-Problemen, die jeweils produktspezifisch gelöst werden müssen.

Während beim produktiven Test in der Regel die gesamte Schaltung mit Hilfe von Prüfpfaden getestet wird, müssen beim strukturellen Selbsttest einige Besonderheiten beachtet werden. Die für die Ansteuerung des Selbsttests notwendige Logik muss von den Scan-Ketten ausgeschlossen werden. Dies führt dazu, dass sich die Scan-Konfiguration des Selbsttests leicht von der produktiven

Scan-Architektur unterscheiden muss. Diese Sonderbehandlung einzelner Designhierarchien erschwert den Einbau der Prüfpfade. Ähnliches gilt auch für die Takt- und Reset-Signale. Beim produktiven Scan-Test wird zentral auf die Scan-Takte und Resets umgeschaltet. Beim Selbsttest muss der LBIST-Controller weiterhin mit dem funktionalen Takt betrieben werden, um die Kontrolle über den Test zu behalten. Da nach Beendigung des Selbsttests sämtliche Register mit pseudo-zufälligen Werten geladen sind, muss die Schaltung anschließend zurückgesetzt werden, um zurück in den normalen Betrieb zu gelangen. Gesonderte Maßnahmen sind im LBIST-Controller notwendig, um das Signaturregister von diesem Reset auszuschließen, so dass die Signatur nach dem Neustart erhalten bleibt und ausgewertet werden kann. In der Regel ist auch ein Mechanismus gewünscht, der den Selbsttest beim Startvorgang umgehen kann. Ansonsten würde ein Designfehler in der Selbsttestlogik dazu führen, dass sich die Schaltung überhaupt nicht starten lässt, was ein sehr hohes Designrisiko bedeuten würde. All diese Punkte müssen produktspezifisch bei der Spezifikation des LBIST-Controllers berücksichtigt werden.

#### **VIII.4.7. Test-Zugang über serielle Standard-Schnittstellen**

Um einen Testzugang zu einzelnen Bausteinen des Gesamtsystems zu erhalten, wurde ein Interface entwickelt, welches über funktionale Standard-Schnittstellen wie CAN und FlexRay einen Zugriff auf die interne Scantest-Architektur des zu testenden Halbleiterbauelements erlaubt. Das Universal Scan-Test Interface (USIF) stellt hierbei die Brücke zwischen den diversen Standard-Schnittstellen und der internen Test-Technologie dar (siehe Abbildung VIII-33). Zunächst wird die Anbindung an CAN, FlexRay und USB unterstützt. Da das USIF unabhängig von konkreten Standard-Schnittstellen konzeptioniert ist, kann es relativ einfach auf andere Schnittstellen erweitert werden. Neben der Übertragung von Scantest-Daten ist auch die Übertragung von JTAG-Informationen über das USIF möglich. Die Anbindung der JTAG-Schnittstelle an das funktionale Bussystem erlaubt es neben dem Scantest auch andere Tests wie zum Beispiel Selbsttests für Speicher und Logik vom System aus zu starten und auszuwerten.

Für eine benutzerfreundliche Konfiguration und Ausführung des Scan-Tests auf dem angeschlossenen Steuergerät wie zum Beispiel PC oder Notebook, wurde eine graphische Benutzerschnittstelle entworfen und implementiert.

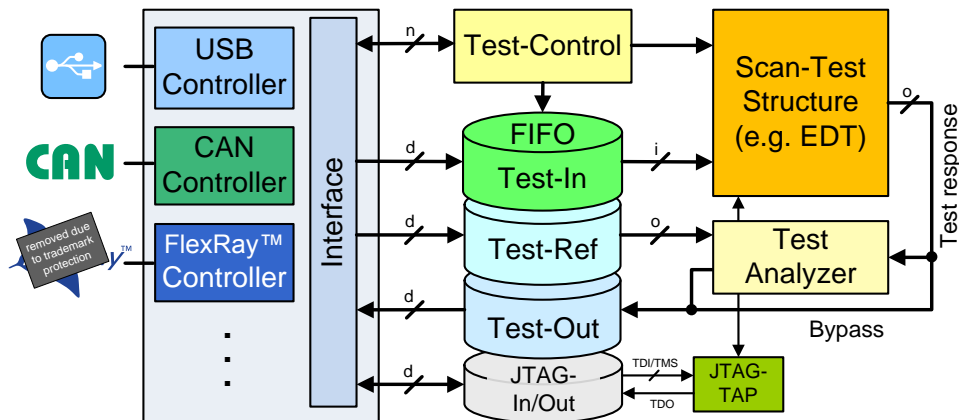


Abbildung VIII-33: Aufbau des Universal Scan-Test Interface

#### VIII.4.7.1. Steuerung des Testzugangs

Das Test-Interface und somit der Testablauf wird mittels definierter Steuerdaten kontrolliert. Sobald eine der integrierten Schnittstellen empfangende Daten meldet (Daten im Empfangspuffer verfügbar), wird der "virtuelle" Eingangsport des USIF-Controllers auf die jeweilige Schnittstelle geschaltet. Das bedeutet, in dieser Komponente ist nur eine allgemeine Schnittstelle, mit Ein- und Ausgangsport, sichtbar, ohne Informationen (Architektur, Protokoll, usw.) der zugrundeliegenden Kommunikations-Controller verwalten zu müssen. In Abhängigkeit der Quelle der Eingangsdaten, die im Source-Register gespeichert wird, werden mittels Schalter die Ports der USIF-Komponente mit den Ports der jeweiligen Schnittstelle verbunden. Somit ist das Test-Interface unabhängig von speziellen Schnittstellen gestaltet. In Abbildung VIII-34 ist die Zustandsmaschine des zentralen Controllers für den Testzugang dargestellt.

Das erste Steuerdatenwort (Abbildung VIII-35), gelesen im Zustand *init*, muss im *Test-Control*-Feld die Codierung für den Start der Kommunikation oder für die Initiierung eines Startup-Test enthalten. Im Falle eines Startup-Tests wird sofort die Prozedur des BISTs mit on-chip Steuerinformationen ausgeführt. Soll die Kommunikation gestartet werden, wird zunächst der Wert im *Controller-Address*-Feld mit dem gespeicherten Wert im *Source-Register* abgeglichen, um einen korrekten Kommunikationsbeginn zu verifizieren. Daraufhin folgt die Zugangskontrolle, um die internen Strukturen vor unbefugtem Zugriff zu schützen. Ist der Zugang durch die Login-Prozedur zugelassen, können Schreib- und Lesezugriffe für die Steuerung eines umfangreichen Scan-Tests erfolgen. Der Scan-Test kann vor oder nach der Übertragung der Testdaten gestartet werden. Liegen die Testdaten bereits vor, kann sofort begonnen werden, ansonsten wird je nach Verfügbarkeit der nötigen Testdaten der Scan-Test fortgeführt.

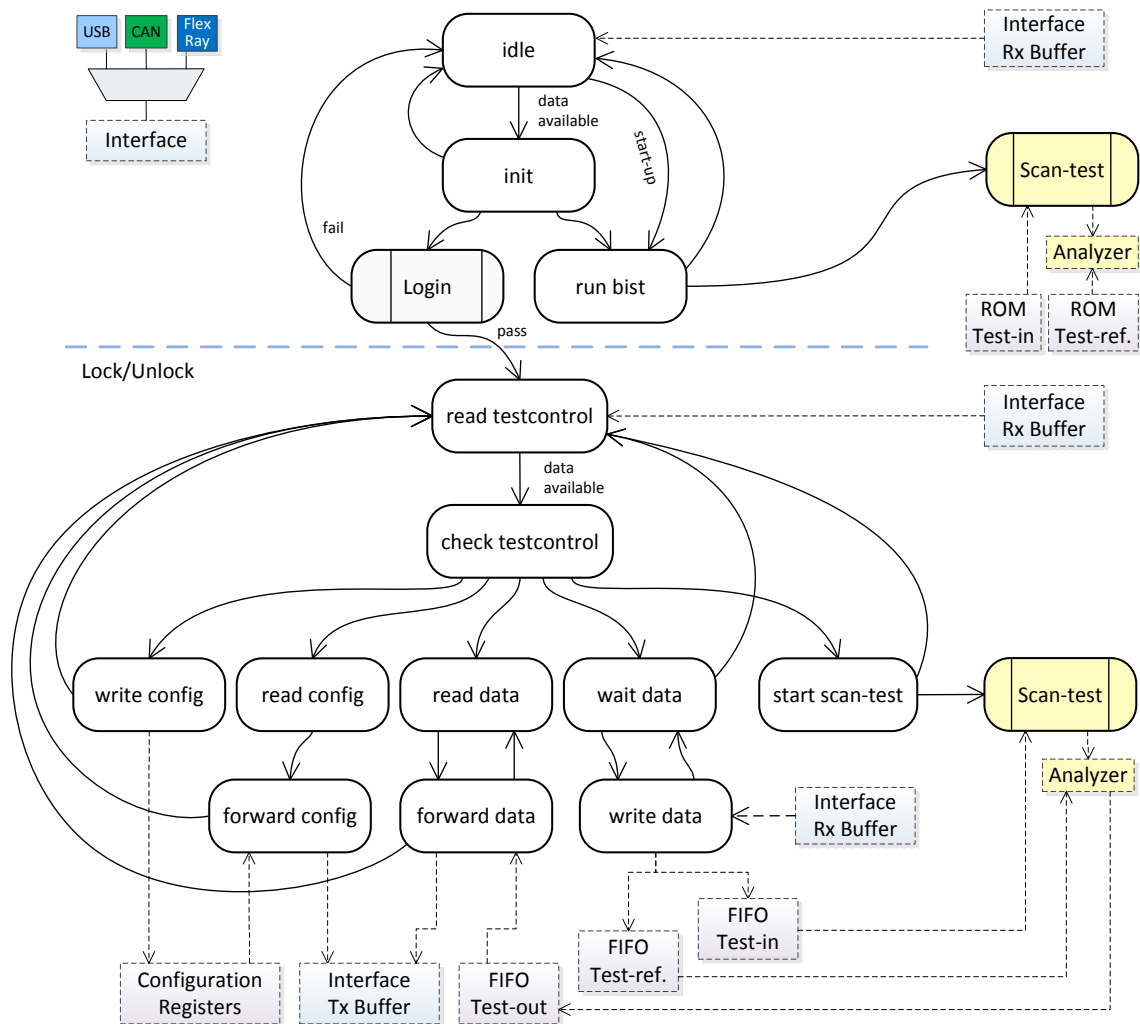


Abbildung VIII-34: USIF-Kontrollfluss

#### VIII.4.7.2. USIF-Datenformat

Zur Steuerung des *Universal Scan-Test Interface* (USIF) wurde ein konsistentes Datenformat entwickelt. Dabei sind sämtliche Steuerdatenwörter einheitlich, mit einem *Control*-Feld, einem (unterteilten) Adressfeld und einem optionalen Datenfeld, aufgebaut. Die Größe eines Steuerdatenwortes entspricht der Datenwortbreite der USIF-Architektur, also der Testdatenspeicher und Sende- und Empfangs-FIFOs der Kommunikations-Controller. Diese sind generisch ausgelegt und in der aktuellen Variante 32 Bit. Es kann also in jedem Takt ein Datenwort aus dem Empfangsspeicher des adressierten Kommunikations-Controller (falls Daten bereits empfangen) ausgelesen und verarbeitet werden.

Das Steuerdatenwort des USIF-Controllers ist wie folgt codiert:

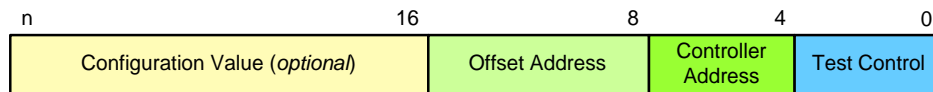


Abbildung VIII-35: USIF-Datenformat

Im *Test-Control*-Feld ist die auszuführende Operation definiert. Hiermit geschieht die Steuerung des USIF-Kontrollflusses. Das Feld *Controller-Address* gibt die Komponente des Test-Interfaces an, in der eine Schreib-/ Lesezugriff erfolgen soll. Das *Offset-Address*-Feld enthält die interne Adresse der spezifizierten Komponente, um auf ein Konfigurationsregister oder ein Speicherblock zuzugreifen.

Falls ein Konfigurationszugriff erfolgt, enthält das Konfigurationsfeld den Wert für das durch Controller-Adresse und Offset-Adresse (Controller-interne Adresse) spezifizierte Register. Im Falle mehrerer kleiner Parameter kann das Konfigurationsfeld auch eine Kombination mehrere in einer Adresse "gebündelte" Konfigurationswerte enthalten. Übersteigt hingegen die Breite eines Konfigurationsregisters die des Konfigurationsfeldes, muss es in mehrere Adressbereiche aufgeteilt werden. Dementsprechend werden auch mehrere Schreib-/Lesezugriffe zur Adressierung jedes dieser Bereiche benötigt.

Im Falle von zu übertragenden Testdaten ist *Test-Control* für den Schreibzugriff gesetzt. Es ist das FIFO der Testeingaben oder der Referenzwerte adressiert. Durch das optionale Feld ist nun die Anzahl der darauffolgenden Datenwörter bestimmt (Abbildung VIII-36). Somit können die Daten mit voller Bandbreite übertragen werden, bis der angegebene Wert mittels eines Zählers erreicht ist. Das Steuerdatenwort stellt den Header eines Datenblocks dar. Jeder zu übertragene Datenblock muss durch solch einen Header angeführt werden.

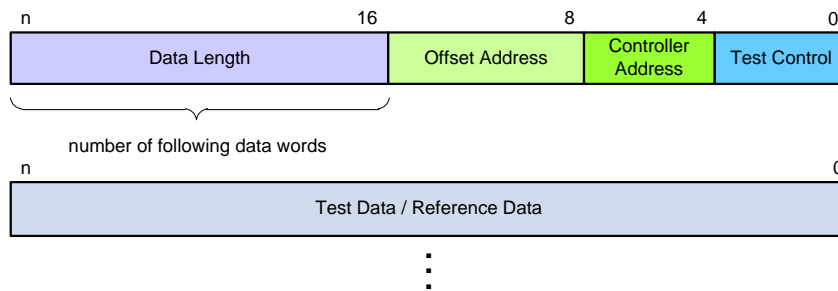


Abbildung VIII-36: USIF-Datenformat zur Datenübertragung

Die Anfrage von Testdaten erfolgt über den *Test-Control*-Code für den Lesezugriff. Die Übertragung verläuft äquivalent zur Schreibanweisung. Im Feld für die Datenlänge ist die Anzahl der zu lesenden Datenwörter anzugeben. Soll der gesamte Speicher der Testausgaben ausgelesen werden, gibt es die Möglichkeit die Datenlänge auf null zu setzen. Somit ist die Obergrenze für die Anzahl der zu lesenden Datenwörter unterdrückt und der Lesezugriff ist erst beendet, wenn der adressierte Speicher über keine weiteren Daten verfügt.

### VIII.4.7.3. Anbindung der EDT-Logik

Die EDT-Logik (Abbildung VIII-37) wird im Produktionstest durch ein ATE<sup>9</sup> gesteuert. Für das USIF-Konzept, die Diagnose über eine Anwendungsschnittstelle, muss diese Funktionalität von einer Komponente on-chip übernommen werden. Hierzu wurde ein Controller zur Steuerung des EDT-Blocks implementiert. Der EDT-Logik-Controller bildet entsprechend der gelesenen EDT-Kontrollinformationen die Testphasen *load/unload*, *shift* und *capture* nach. Diese Kontrollinformationen werden zusammen mit den Eingaben für die EDT-Kanäle aus der EDT-Pattern-Datei extrahiert und übertragen.

Der EDT-Logik-Controller muss mit doppeltem Scan-Takt betrieben werden. Es werden zuerst die Kontrollinformationen aus dem Test-In-FIFO gelesen und anhand derer die darin enthaltenen Eingangswerte der Kontrollports (*scan\_en*, *edt\_update*, *edt\_bypass*) gesetzt oder die Scan-Clock an die EDT-Clockports (*clk*, *edt\_clk*) durchgeschaltet. Anschließend werden mit steigender Flanke der Scan-Clock die Testdaten aus dem Test-In-FIFO direkt an die EDT-Kanäle (*edt\_channels\_in*) angelegt. Somit sind die Kontrolleingänge rechtzeitig zum Scan-Takt eingestellt, um je nach Testphase die EDT-Logik zu initialisieren, die Testdaten in die EDT-Kanäle zu schieben oder ein funktionales Takt auszuführen. Die EDT-Logik wird also komplett durch die Daten aus der Pattern-Datei getrieben.

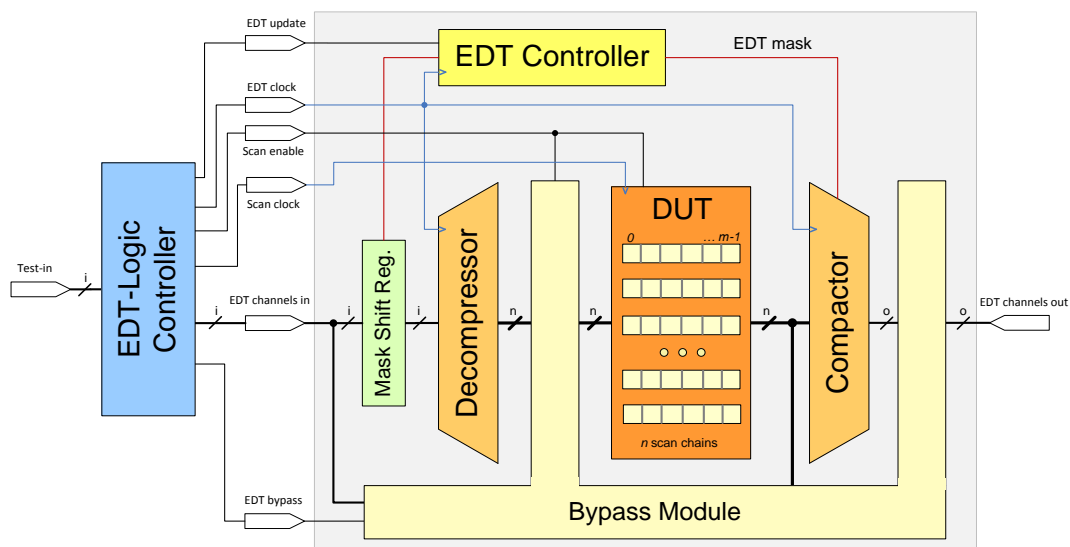


Abbildung VIII-37: Ansteuerung der EDT-Logik

### VIII.4.8. Testanalyse

Für die Auswertung der Testantworten wurde die Analyzer-Komponente entworfen und in die USIF-Architektur integriert. Es werden folgende Modi der Testanalyse unterstützt:

<sup>9</sup> Automatic Test Equipment

- *LogAll*, Ausgabe aller Testergebnisse,
- *LogFaults*, Ausgabe der fehlerhaften Testergebnisse mit zugehörigen Zeitstempel,
- *PassFail*, on-chip Auswertung der Testergebnisse mittels off-chip Referenzdaten,
- *BIST*, on-chip Auswertung der Testergebnisse mittels on-chip Referenzdaten.

Im *LogAll*-Modus werden sämtliche Ergebnisse der Test-Technologie (EDT, Scan-Controller) aufgezeichnet und müssen kontinuierlich aus dem Test-Out-FIFO ausgelesen werden. Der Test stoppt sobald kein Speicherplatz im Test-Out-FIFO vorhanden ist und kann erst wieder mit freigewordenem Speicher fortgesetzt werden.

Sollen hingegen nur die fehlerhaften Testantworten gespeichert werden, ist der Modus *LogFaults* zu wählen. Es findet somit bereits eine Filterung der Ausgaben on-chip statt. Hierbei müssen zusätzlich zu den Teststeuerinformationen die Referenzdaten zu den Testergebnissen über den Testzugang übertragen und im Test-Ref-FIFO abgelegt werden. Ein Fehler in der zu testenden Schaltung ist erkannt, wenn eine Testantwort nicht dem zugehörigem Referenzwort, das zeitgleich aus dem Test-Ref-FIFO geladen wird, entspricht. Zu jeder abweichenden Testantwort muss zusätzlich ein Zeitstempel gespeichert werden. Der Zeitstempel setzt sich aus der Nummer des aktuellen Testmusters, Pattern-Nr.  $p$ , und der bisherigen Anzahl an Shift-Takten der aktuellen Shift-Phase, Shift-Nr.  $s$ , zusammen. Zu jedem Fehler ist somit ersichtlich, in welcher Testantwort welchen Testmusters dieser erkannt wurde. Die Zeitstempel werden in einen separaten Speicher (TS-FIFO), das synchron zum Test-Out-FIFO betrieben wird, geschrieben. Das heißt, im *LogFaults*-Modus wird mit jedem Schreibzugriff auf das Test-Out-FIFO zeitgleich das Wertepaar  $(p,s)$  aus einem Counter in das TS-FIFO übernommen. Die beiden Speicher werden separat ausgelesen, Testantwort und Zeitstempel sind aber über die Speicheradresse, beim FIFO-Prinzip also über den Zeitpunkt des Schreibzugriffs, eindeutig einander zuzuordnen.

Im *PassFail*-Modus wird der Test abgebrochen und ein entsprechendes Bit im Statusregister gesetzt, sobald ein Fehler erkannt wird. Zudem wird die verfälschte Testantwort mit zugehörigem Zeitstempel gespeichert, um diese optional auslesen zu können. Diese Information kann für eine darauffolgende optimierte Diagnose genutzt werden, in der der Suchraum bereits eingeschränkt werden kann.

Der *BIST*-Modus wird während des Startup-Tests (bzw. Shutdown-Test) aktiviert und entspricht dem *PassFail*-Modus, nur dass hier der Analyzer vom Testausgabespeicher (Test-Out-FIFO) entkoppelt ist, also auch nicht die abweichende Testantwort oder den Zeitstempel speichert. Die Referenzdaten, um die Testantworten auszuwerten, werden hierbei aus einem on-chip Testspeicher (Test-Ref-ROM) geladen.

Die Analyzer-Komponente wurde außerdem um einen Analysefilter erweitert, um die Möglichkeit zu bieten nicht sämtliche Testantworten auswerten zu müssen und somit den Referenzdatensatz und, im Falle der Log-Modi, den Datensatz an zu speichernden Testantworten zu reduzieren. Bevor ein Scan-Test gestartet wird, kann der Analyzer mit einer Sample-Rate  $r$  und einer Basis  $b$  konfiguriert werden. Ein



Modulo- $r$ -Counter wird mit jedem Scan-Takt in der Shift-Phase, also mit jeder neuen Testantwort, inkrementiert. Die Basis  $b$  ( $b < r$ ) gibt den Startwert an, ab welcher Testantwort mit der Auswertung begonnen werden soll. Ein neuer Referenzwert wird nur geladen und mit der aktuellen Testantwort verglichen, wenn die nächste gültige Shift-Nummer erreicht ist, also Counterwert  $s \bmod r = b$ . Es findet also vor der Analyse eine Filterung auf jede  $r$ -te Testantwort statt.

#### VIII.4.8.1. Clock Domain Crossing

Die USIF-Architektur bildet mit den verschiedenen Kommunikationscontrollern ein Multiple-Clock System. Es kann als sogenanntes GALS-System<sup>10</sup> betrachtet werden, welches das Problem der Kommunikation und Datenaustausches zwischen asynchronen Einheiten in sich birgt.

Der Datenaustausch kann über Zwischenpuffer synchronisiert werden. Hierzu sind die Sende- und Empfangspuffer der Kommunikationscontroller als Dual-Clock-FIFOs ausgelegt (Abbildung VIII-38). Da Schreib- und Leseport durch den jeweils angeschlossenen Controller getaktet sind, ist das Timing des Schreib- und Lesezugriffs unabhängig voneinander. Somit ist keine weitere Administration von Nöten. Die FIFOs sind zudem noch mit unterschiedlich großen Lese- und Schreibports ausgestattet. Dadurch kann die Datenübergabe zwischen Komponenten unterschiedliche Bitbreiten ohne zusätzlichen Aufwand realisiert werden. So werden z.B. die Testeingaben von der Schnittstelle als 32bit-Wörter in das Test-In-FIFO abgelegt. Der Scan-Controller liest aus diesem die Daten aber nur als 16bit-Wörter entsprechend seines Eingangsports wieder aus.

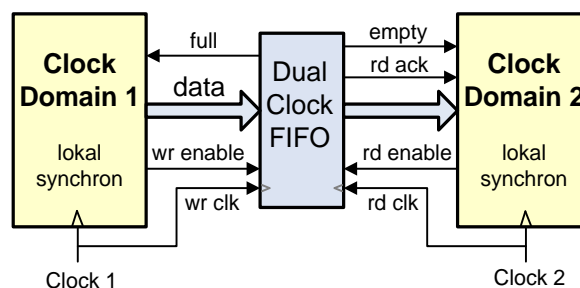


Abbildung VIII-38: Clock Domain Crossing via Dual-Clock-FIFOs

Aber der Kontrollpfad muss auch synchronisiert werden, um den korrekten Empfang von Kontrollsignalen oder den Datenaustausch zwischen Registern der unterschiedlichen Clock Domains sicherstellen. Das kann durch Handshakeverfahren geschehen. Das 2-Phasen Handshake Protokoll ist schneller, beruht aber auf einer komplexeren Schaltung. Das 4-Phasen Handshake Protokoll hat zwei redundante Phasen, wurde aber bevorzugt, da es durch eine simplere Schaltung umgesetzt werden kann und stets einen definierten Ausgangszustand besitzt.

<sup>10</sup> Globally Asynchronous Locally Synchronous

Die Kommunikation findet zwischen Talker und Listener im sogenannten Push-Pull-System statt. Im 4-Phasen-Handshake-Protokoll dienen die ersten beiden Phasen der Synchronisation und die dritte und vierte Phase des Zurücksetzens der Handshake-Signale auf Talker- und Listener-Seite (Abbildung VIII-39). Der Talker signalisiert mit einer Anfrage ( $req=1$ ), dass Daten übergeben werden sollen (*push*) oder entgegengenommen werden können (*pull*). Sobald der Listener die Anfrage quittiert ( $ack=1$ ), werden die Daten transferiert. Nach abgeschlossener Datenübertragung können Anfrage- und daraufhin auch Bestätigungssignal wieder in den Ausgangszustand gesetzt werden ( $req=0 \rightarrow ack=0$ ). Da das Handshakesignal *req* Lese- oder Schreibanfrage bedeuten kann, ist von einem bidirektionalen Datentransfer die Rede.

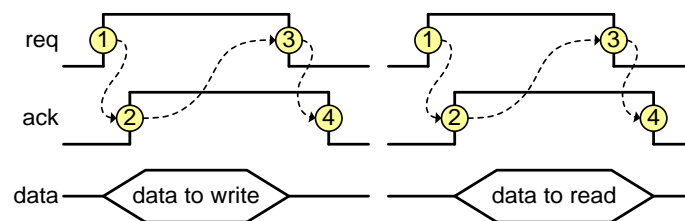


Abbildung VIII-39: 4-Phasen-Handshake

#### VIII.4.8.2. USIF-Applikation

Die Schnittstelle zum Anwender bildet eine Applikation auf PC-Seite (Java), die es ermöglicht, Testdaten über den USB-Anschluss zu senden und wieder zu empfangen. Eine Benutzeroberfläche stellt die Eingabemasken zu Konfiguration der Controller und zur Eingabe der Testdaten bereit. Die zu übertragenden Daten werden eingelesen und die Verbindung zum USIF wird hergestellt. Um die USB-Geräte, die an dem Host des Betriebssystems angeschlossen sind, zu identifizieren und mit der entsprechenden Gegenseite zu kommunizieren, setzt die Applikation auf eine USB-API mit zugehörigem Treiber auf. Die Wahl fiel auf die offene USB-Bibliothek *Libusb* (lizenziert unter GNU Lesser General Public License version 2.1).

Alle am USB-Host angeschlossenen USIF-Geräte (durch die Geräte-Identifikatoren VID und PID gefiltert) werden erkannt und konfiguriert. Für jeden Schreib- und Lesezugriff wird der entsprechende Kommunikationskanal (Downstream/ Upstream Pipe) zu den USB-Endpunkten (Empfangs-/Sendepuffer), die für den Bulk-Transfer ausgelegt sind, geöffnet. Dabei werden Testdaten aus einer eingelesenen Datei in den Empfangspuffer (OUT-Endpoint) des Geräts geschrieben. Daten für den Lesezugriff liegen im Sendepuffer (IN-Endpoint) des Geräts. Da es sich bei USB um einen Master-Slave-Bus handelt, kann das Gerät nicht selbständig verfügbare Daten melden, sondern kann nur auf entsprechende Anfrage reagieren. Um nicht ständig Daten aus einem leeren IN-Endpoint anzufragen und so NAK-Antworten zu produzieren (welche zu Exceptions in der Libusb-API führen), werden zusätzliche aktuelle Geräteinformationen auf Anwenderseite benötigt. Hierzu wurde das Gerät um ein Interrupt-Endpoint ergänzt, um stets aktuelle Status-Informationen bereitzustellen. Ein Thread der Anwendung liest nun regelmäßig (Polling im Intervall

von 1 ms) die Status-Informationen, in denen u.a. Zustand des Tests und Anzahl der im Lesebuffer (IN-Endpoint) verfügbare Daten stehen, aus. Wenn gültige Daten anliegen, kann eine Leseanfrage erfolgen, um diese senden zu lassen und bei korrektem Empfang in einer Datei zu speichern.

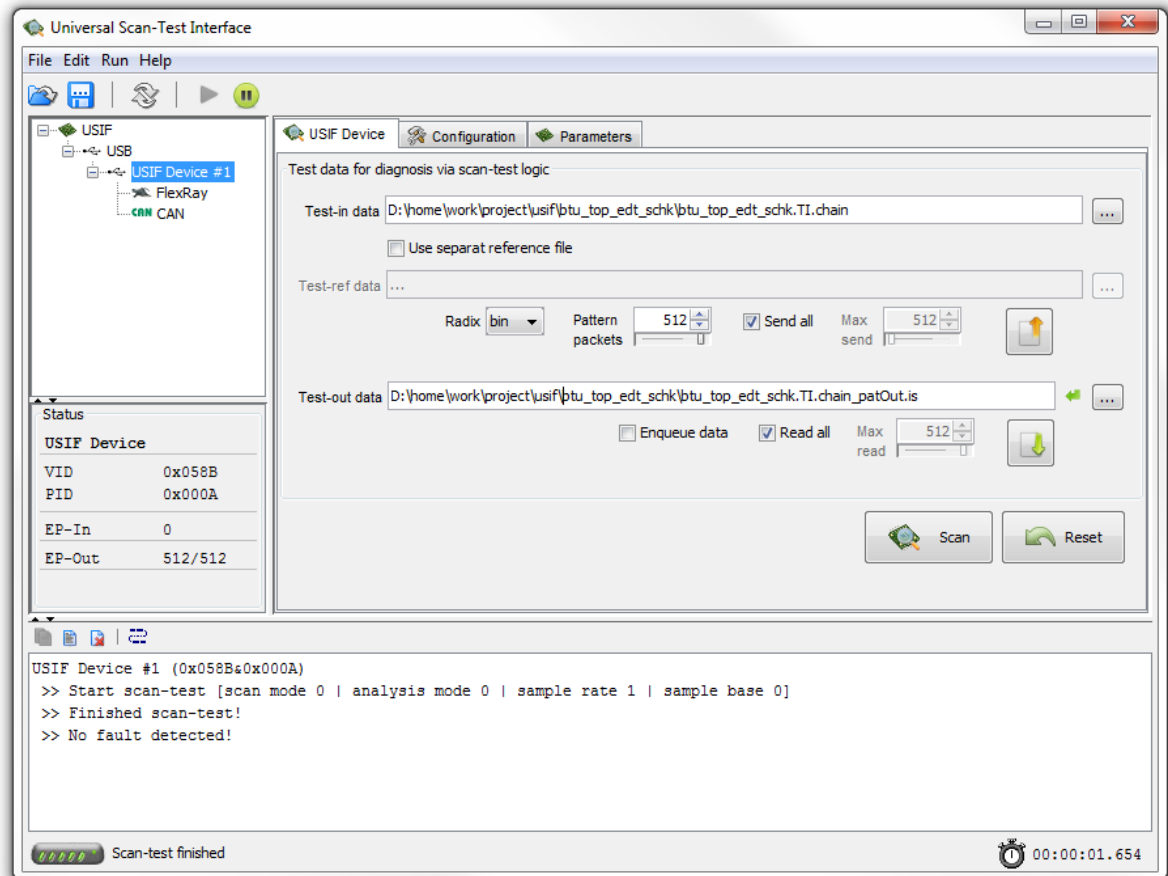


Abbildung VIII-40: Kommunikationsseite der USIF-Applikation

Um einen kompletten Testlauf zu automatisieren wurde die Scan-Funktion implementiert. Hierzu müssen zunächst die nötigen Eingaben für die einzulesenden Testdaten und abzuspeichernden Testantworten erfolgen. Wird ein *USIF-Device* angewählt, erscheint die Passwordeingabe, um sich für den Zugriff zu authentifizieren. Das Passwort wird verschlüsselt (AES128) über die USB-Schnittstelle gesendet. Ist das Passwort korrekt, wird dies vom USIF-IC bestätigt und die Applikation öffnet die zugehörige Kommunikationsseite (*USIF Device*, Abbildung VIII-40). Hier kann die Pattern-Datei angegeben werden. Enthält diese keine Informationen zu den erwarteten Testantworten, kann für eine Testauswertung zusätzlich eine Datei mit den Referenzwerten zu den Testausgaben spezifiziert werden. Zu diesen Eingaben ist unter anderem der Radix, zur korrekten Interpretation der Daten (binär, hexadezimal), und die maximale Paketgröße, in der der Testdatensatz für die Übertragung aufgeteilt wird, einzutragen. Die Paketgröße

darf die Kapazität des Test-In- bzw. Test-Ref-FIFOs nicht übersteigen, was aber durch die Applikation sichergestellt wird.

Um einen Scan-Test zu konfigurieren, wurde eine weitere Seite erstellt (*Configuration*, Abbildung VIII-41). Über eine Eingabemaske können die nötigen Parameter spezifiziert werden. Darüber hinaus ist es möglich die Scan-Test-Parameter aber auch weitere USIF-Register (siehe USIF Memory Map) über den Konfigurations-Editor zu setzen. Ist der Editor aktiviert und darin ein Register eines Scan-Test-Parameters adressiert, wird der in der Eingabemaske eingestellte Wert überschrieben. Im Editor sind zu übermittelnden Daten im USIF-Datenformat (Instruktion, Adresse, Wert) binär oder hexadezimal anzugeben.

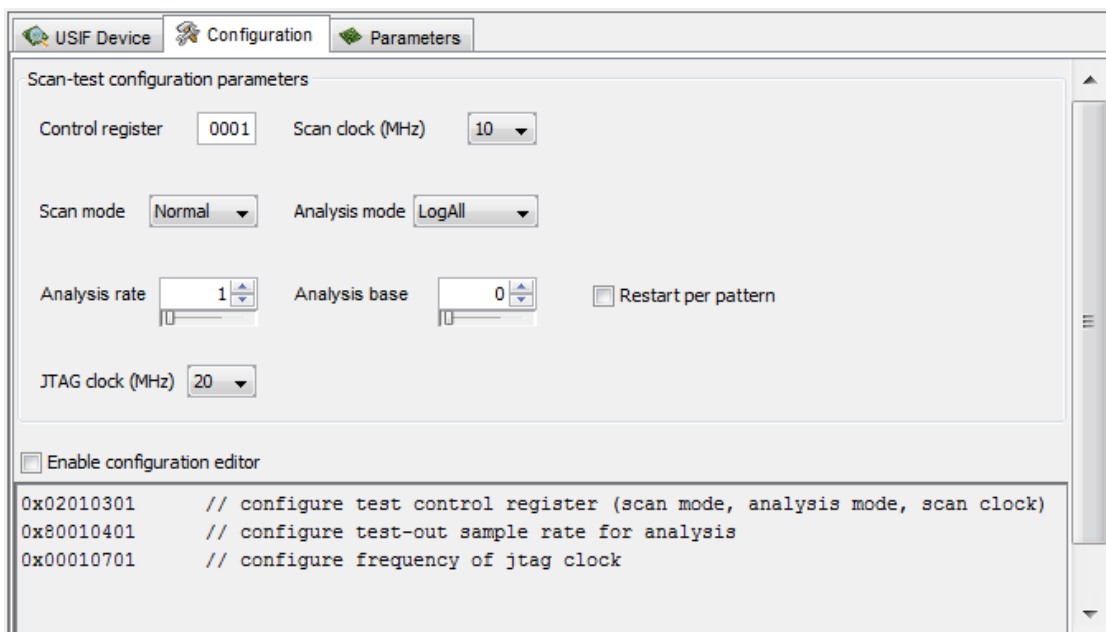


Abbildung VIII-41: Konfigurationseite der USIF-Applikation

Tabelle VIII.6: Konfigurationsparameter für den Scan-Test

<i>Control Register</i>	Das erste Bit (lsb) erzwingt ein Reset der Scan-Test-Komponenten (inkl. Test-FIFOs) vor der Konfiguration. Die weiteren Bits sind bisher nicht genutzt.
<i>Scan Mode</i>	Der Scan-Modus <i>Normal</i> bedeutet, dass die Scan-Test-Architektur mit den komprimierten Testdaten angesteuert wird. Mittels des Modus <i>Bypass</i> können die unkomprimierten Testmuster in die Scan-Ketten geschoben werden.
<i>Analysis Mode</i>	Der Modus <i>LogAll</i> ermöglicht die Aufzeichnung aller Testausgaben (in Abhängigkeit vom Rate-Base-Tupel). Im <i>LogFaults</i> -Modus werden nur die Testantworten gespeichert, die ungleich ihren Referenzwerten sind, also durch die Fehler erkannt wurden. Durch den <i>PassFail</i> -Modus wird der Test abgebrochen, sobald ein Fehler erkannt wird.
<i>Scan Clock</i>	Konfiguration des Scan-Taktes, optional 10, 20 oder 50 MHz.

<i>Analysis Rate</i>	Die Analyserate $r$ gibt an, dass nur jede $r$ -te Testantwort auszuwerten und ggf. abzuspeichern ist ( $1 \leq r < 256$ ).
<i>Analysis Base</i>	Die Basis $b$ gibt an, ab welcher Testantwort mit der Auswertung begonnen wird (falls $b \geq r$ , wird $b = b \bmod r$ gesetzt).
<i>Restart per Pattern</i>	Der Parameter gibt an, ob die Rate über ein Pattern hinaus weitergezählt (0), oder jeweils mit der $b$ -ten Testantwort eines Pattern Neubegonnen wird (1).
<i>JTAG Clock</i>	Konfiguration des JTAG-Taktes, optional 10, 20, 30 oder 50 MHz.

### Extraktion der Testmuster

Für die Testmusterformate STIL1450, STIL2005 und CTL wurde eine Funktion zur Extraktion der Testdaten implementiert. Da diese Formate einen äquivalenten Aufbau besitzen, kann hierfür eine gemeinsame Funktion, die die Dateien scannt und die nötigen Testdaten extrahiert, genutzt werden. In der aktuellen Version handelt es sich hierbei um einen Pattern-Filter, der lediglich die Eingaben für die EDT-Kanäle (*edt\_channels\_in*) und die Referenzausgaben (*edt\_channels\_out*) aus der textuellen Testablaufbeschreibung extrahiert. Die EDT-Kontrolldaten zur Steuerung der Testphasen (*load/unload*, *shift capture*) werden durch die Funktion hinzugefügt und mit jedem Testpattern übertragen. Die Makro- und Prozeduraufrufe in diesen Pattern-Dateien werden also nicht behandelt.

Um sämtliche Testinformationen der Pattern-Datei direkt zu entnehmen (ohne Konfiguration über Makros und Prozeduren), wurde ein weiteres Format hinzugezogen. Hier erwies sich das TDL-Format als gut lesbar. Es handelt sich um ein statisches Format, in dem sämtliche EDT-Eingaben (inklusive der Kontrollinformationen) direkt nacheinander folgend definiert sind. Es wird zuerst die Initialisierungen der EDT-Ports gelesen, um dann die Testdaten für die EDT-Kontrollports und EDT-Kanäle zu interpretieren und als Bitstrom zu speichern.

### Scan-Funktion

Mittels der Scan-Funktion ist der Testablauf, die paketweise Übertragung der Testeingaben und -ausgaben über die USB-Schnittstelle und die Testauswertung, automatisiert. Zunächst wird der Testdatensatz aus der eingelesenen Datei je nach Format extrahiert und in eine Liste von Datenworten für die Testeingaben und gegebenenfalls in eine separate Liste für die Referenzwerte abgespeichert. Als Erstes werden dem USIF-IC die angegebenen Konfigurationsdaten übertragen und darauf bereits der Test-Controller aktiviert. Dieser startet aber erst, sobald die nötigen Testdaten verfügbar sind.

Aus der Liste der Testeingaben wird je nach spezifizierter maximaler Paketgröße ein Teildatensatz entnommen und in das Test-In-FIFO übertragen. Dem Datenpaket wird hierzu ein zusätzlicher Header, zur Interpretation durch den USIF-Controller (Schreibinstruktion und entsprechender FIFO-Eingangsport als Zieladresse), vorangestellt. Vor dem Versenden wird die Paketgröße des Datensatzes gegebenenfalls auf den Umfang des verfügbaren freien Speicherplatzes im USB-

Empfangspuffer verringert. Das Datenpaket wird schließlich durch den USB-Host wiederum zerlegt, in Frames verpackt und versendet.

Im Falle der Analyzer-Modi *LogFaults* oder *PassFail* muss zudem ein Paket aus der Liste der Referenzdaten in das Test-Ref-FIFO geschrieben werden. Daraufhin startet der Test-Controller den Scan-Test, also dekomprimiert die Testdaten zu Testmustern, schiebt diese in die Scan-Ketten und kompaktiert die Testantworten. Die Testantworten werden je nach Analysemodus und Analyserate entweder direkt oder nach Filterung durch den Analyzer in das Test-Out-FIFO geschrieben. Der Test befindet sich im Wartezustand, wenn das Test-In-FIFO leer oder, falls Analysemodus nicht *LogAll*, das Test-Ref-FIFO leer oder das Test-Out-FIFO voll ist.

Die Test- und FIFO-Zustände (*Test run*, *Fault detected*, *Test-In full*, *Test-Ref full*, *Test-Out empty*) werden über eine Statusbotschaft mittels Polling des USB-Statusendpunktes (Interrupt-Transfer im Intervall von 1ms) regelmäßig abgefragt. Wird im *PassFail*-Modus ein gefundener Fehler signalisiert, ist der Scan-Test beendet und die Scan-Funktion terminiert. Optional kann die Testantwort, durch die der Fehler erkannt wurde, mit zugehörigem Zeitstempel ausgelesen werden.

Ansonsten liest die Applikation das Test-Out-FIFO komplett aus, sobald verfügbare Testantworten gemeldet werden. Hierzu wird dem USIF-Controller die entsprechende Leseanweisung übermittelt (Leseinstruktion und Ausgangsport des Test-Out-FIFOs als Quelladresse). Im *LogFaults*-Modus werden darauf auch die zugehörigen Zeitstempel aus dem TS-FIFO gelesen.

Die nächsten Testdatenpakete werden gesendet sobald Speicherplatz im Test-In bzw. Test-Ref-FIFO frei ist. Dies könnte theoretisch sofort ohne weitere Verzögerung erfolgen, da die Geschwindigkeit der Datenverarbeitung durch die Applikation und die Übertragung über die USB-Verbindung im Verhältnis zur Verarbeitung der Testdatenpakete auf dem IC sehr langsam erfolgt. Da aber auf die Statusmeldungen reagiert wird und diese im festen Intervall abgefragt werden, können Wartezeiten entstehen.

Nach Übermittlung des gesamten Testdatensatzes ist der Scan-Test abgeschlossen. Es werden die restlichen Testantworten ausgelesen und der Testantwortensatz mit zugehörigen Zeitstempeln in die spezifizierte Ausgabedatei geschrieben. Im Falle des *LogAll*-Modus werden die Zeitstempel durch die Applikation eingefügt und zudem, falls Referenzdaten vorhanden sind, die fehlerhaften Testantworten markiert.

#### **VIII.4.9. Implementierung und Verifikation für Beispielschaltungen**

Die entwickelte Selbsttest-Lösung und die weiterführenden Konzepte wurden anhand von verschiedenen exemplarischen Beispielschaltungen verifiziert. Der zur Steuerung des Selbsttests notwendige Scan-Controller wurde hierzu in den RTL-Code der Schaltungen integriert und mit der EDT-Logik des Produktionstests verbunden. Für die die Ansteuerung und Auswertung des Selbsttests wurden verschiedene Konzepte implementiert. Neben der direkten Ansteuerung über eine existierende JTAG-Schnittstelle wurde auch die Integration des Selbsttest in die funktionale Startsequenz des Bausteins realisiert. Die so erweiterte Schaltung wurde synthetisiert und mit Scan-Ketten versehen. Auf der resultierenden Gatter-Netzliste wurde mit Hilfe des kommerziellen ATPG-Werkzeugs eine Fehlersimulation zur

Bestimmung der Testabdeckung durchgeführt und der resultierende Wert des Signaturregisters bestimmt. Außerdem wurde mit den Design-Werkzeugen eine Vielzahl von Checks durchgeführt, welche die Funktionsweise und den korrekten Anschluss der Selbsttestlogik prüfen. In einer abschließenden Logik-Simulation wurde der komplette Ablauf des Selbsttests simuliert. Der aus der Simulation ermittelte Wert des Signaturregisters stimmte hierbei mit dem vorausgerechneten Wert des ATPG-Werkzeugs überein. Desweiteren wurde der Signalverlauf der relevanten Selbsttestsignale überprüft und mit dem erwarteten Verhalten verglichen.

Um die Funktionsweise des seriellen Testinterfaces zu verifizieren und die Ansteuerung durch die USIF-Applikation zu erproben, wurde ein prototypisches Gesamtdesign mit Hilfe programmierbarer Logik-Bausteine (FPGA<sup>11</sup>) auf einem Entwicklungsboard implementiert. Hiermit konnte die Anbindung einer typischen EDT-Scantest-Architektur eines realen Microcontroller-Bausteins an die seriellen Standard-Schnittstellen CAN und FlexRay unter Verwendung des entwickelten Universal Scan-Test Interfaces erfolgreich demonstriert werden. Hierbei konnte der komplette Ablauf ausgehend von der Testmustererzeugung bis hin zur Ausführung und Auswertung des Tests durchlaufen und verifiziert werden.

Mit Hilfe der entwickelten Methoden und Schaltungen kann somit ein struktureller Selbsttest für einen Halbleiterbaustein im System durchgeführt werden. Zur weiterführenden Diagnose kann über funktionale Schnittstellen auf die Scantest-Architektur des Bausteins zugegriffen werden und berechnete Testmuster ausgeführt und ausgewertet werden.

#### VIII.4.10. Literatur

- [1] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, T. Kun-Han, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eider, Q. Jun, „Embedded Deterministic Test for Low Cost Manufacturing Test“, International Test Conference (ITC), 2002, pp. 301-310
- [2] P. H. Bardell, W. H. McAnney: "Parallel Pseudorandom Sequences for Built-In Test", Proc. Int. Test Conf., 1984, pp. 302-308
- [3] P. Bardell, W. H. McAnney, J. Savir: "Built-in Test for VLSI", John Wiley & Sons, New York, 1987
- [4] U. Gätzschnann, C. Galke, and H. T. Vierhaus, "Ein flexibles Verfahren zur Testdaten-Kompaktierung und -Dekompaktierung für den Scan-Test", 16. ITG-GI-Workshop Test und Zuverlässigkeit von Schaltungen und Systemen (ZuE), 2004
- [5] R. Frost, D. Rudolph, C. Galke, R. Kothe, and H. Vierhaus, "A configurable modular test processor and scan controller architecture", On-Line Testing Symposium, IEEE International, pp. 277–284, 2007
- [6] R. Kothe and H. T. Vierhaus, "Test data and power reductions for transition delay tests for massive-parallel scan structures", Digital Systems Design, Euromicro Symposium on, pp. 283–290, 2010

---

<sup>11</sup> Field Programmable Gate Array

- [7] S. C. Talbot and S. Ren, "Comparision of fieldbus systems CAN, TTCAN, FlexRay and LIN in passenger vehicles", International Conference on Distributed Computing Systems Workshops, pp. 26–31, 2009
- [8] W. Zimmermann and R. Schmidgall, "Bussysteme in der Fahrzeugtechnik: Protokolle und Standards", ATZ-MTZ Fachbuch, Vieweg, 2007
- [9] D. Paret and R. Riesco, "Multiplexed networks for embedded systems: CAN, LIN, Flexray, Safe-by-Wire", Wiley, 2007
- [10] W. Lawrenz, "CAN Controller Area Network: Grundlagen und Praxis", Hüthig Verlag, 2011
- [11] F. Consortium, "FlexRay Protocol Specification v2.1 Rev A", December 2005
- [12] Rausch, Mathias, „FlexRay – Grundlagen, Funktionsweise, Anwendung“, Carl Hanser Verlag München, 2008
- [13] Reineke, O., „Design und Realisierung einer FlexRay-Steuergeräteplattform auf PowerPC-Basis“, FH NTA, 2005
- [14] Anandarajah, M., „USB Controller IP For FPGA Designs“, University of Queensland, Oct 2001
- [15] J. Axelson, "USB 2.0 - Handbuch für Entwickler, vol. 3", Hüthig Jehle Rehm, 2007
- [16] I. USB Implementers Forum, "USB Specification 2.0 Revision 2.0", April 2000
- [17] Sauter, B., „USB-Stack for Embedded-Systems“, FH Augsburg, Juli 2007

### **VIII.5. Ergebnisse zur Aufgabe 2.4: Entwicklung einer Methodik, um die DC eines FSBST für einen Microcontroller Core nachzuweisen, sowie deren Anwendung**

#### **VIII.5.1. Inhalte der Aufgabe 2.4**

In der Aufgabe 2.4 wurde eine Methodik mit dem Ziel entwickelt, die Fehlerabdeckung (**Diagnostics **Coverage **DC****) eines **Funktionalen **Software **Basierten **Selbst **Tests **FSBST****** für einen Microcontroller Core nachzuweisen.********

Für die Anwendung der Methodik wurden Microcontroller der Firma Infineon aus der TriCore Familie benutzt.

Die Aufgabe 2.4 wurde in zwei Unterpakete gegliedert.

In der Aufgabe 2.4.1 wurde im Wesentlichen die Methodik entwickelt, um die Fehlerabdeckung (**Diagnostics **Coverage **DC****) eines **Funktionalen **Software **Basierten **Selbst **Tests **FSBST****** für einen realen **Micro-**Controller **MC**** Core nachzuweisen. Die Entwicklung des FSBST und die praktischen Nachweise fanden am Beispiel eines TriCore Microcontroller der 32-bit Generation von Infineon Technologies AG statt. Die marktübliche Produktbezeichnung lautet **TC1387**, dieses Produkt ist am Markt frei erhältlich.**********

In der Aufgabe 2.4.2 wurden zwei verschiedene Hardware basierende Methoden analysiert, bezüglich des Ziels, den **Funktionalen **Software **Basierten **Selbst **Tests **FSBST****** zu verbessern und zu ergänzen. Im Vordergrund standen dabei Verbesserungen bezüglich der Ausführungszeit, der **Diagnostic Coverage DC** und des Nachweises der Wirksamkeit des Selbsttests.******





### **VIII.5.2. Fehlermodelle und Fehlersimulation**

Die Fehlersimulation ermöglicht eine sehr systematische, zielgerichtete, genaue und planvolle Untersuchersuchung der Auswirkungen von Fehlern auf das Verhalten der TriCore CPU. Sie erlaubt es, reale physikalische Fehler in den Microcontroller Core einzubauen (z.B. Störungen der Versorgungsspannung oder der Taktversorgung, Fehlereinbau über Fehlerprüfketten oder ähnliches). Daher wurde die Fehlersimulation ausgiebig während der Entwicklung des **Funktionellen Software Basierten Selbst Tests FSBST** Testprogramms genutzt. Ebenso wurde diese Methode für die Bestimmung der erreichten Testabdeckung (diagnostic coverage DC) des Microcontrollers genutzt.

Dieses Kapitel beschreibt wie die Fehlersimulation für die Entwicklung des FSBST verwendet wird. Auf die hier beschriebenen Einstellungen werden wir uns in nachfolgenden Kapiteln beziehen, um die einzelnen Tests genauer spezifizieren zu können. In diesen Kapiteln wird die erreichte Testabdeckung bestimmt.

#### **VIII.5.2.1. Fehlersimulation auf Register-Ebene oder Gatter-Ebene**

Während des Entwicklungsprozesses wird die TriCore CPU auf verschiedenen Ebenen durch Register-Transfer-Logic (RTL) oder durch Gatterebene (GL) Netzlisten dargestellt. Diese Netzlisten bestehen aus hunderttausenden von einfachen Logikgattern (z.B. UND, ODER, MUX, ...) und – im Falle von Register-Transfer-Logic RTL – von entsprechenden Schaltungsblöcken (z.B. 32bit MUX, 8bit Addierer, ...), welche durch Netze verbunden sind (z.B. Leitungen). Diese Netzlisten sind eine genaue Abbildung der CPU in ihrem physikalisch realisierten Aufbau:

- Schaltungsblöcke werden umgesetzt durch entsprechende Logikgatter oder Standardzellen aus der Zellbibliothek während der Logiksynthese. Diese werden danach für das physikalische Chip-Layout genutzt (Place & Route P&R Schritt).
- Einfache Logikgatter stellen bereits Standardzellen aus der Zellbibliothek dar. Diese werden danach für das physikalische Chip-Layout genutzt (P&R Schritt).
- Netze werden umgesetzt als Verbindungen oder VIAs und danach für das physikalische Chip-Layout genutzt (P&R Schritt).

Dank der exakten Umsetzung der Netzlisten in das physikalische Layout auf dem Chip sind Register-Transfer-Listen oder Gatterebene Netzlisten sehr gut für die Fehlersimulation geeignet, um die Auswirkungen von physikalischen Fehlern zu ermitteln und die Fähigkeit der Testmuster zu ermitteln, diese Fehler zu entdecken. Ein weiterer Vorteil dieser Methode (z.B. im Vergleich zur Simulation auf Transistorebene) ist die hohe Leistungsfähigkeit der heutigen Industrie Standard Werkzeuge für die Simulation auf Register-Transfer-Listen oder auf Gatterebene Netzlisten. Diese Tools ermöglichen dem Halbleiter-Hersteller eine hohe Anzahl an Simulationsdurchläufen innerhalb einer vorgegebenen begrenzten Testzeit.

#### **VIII.5.2.2. Fehlereinbau und Fehlererkennung durch Fehlersimulation**

Die IEC61508 Norm beschreibt mehrere unterschiedliche Fehlermodelle, die für eine CPU betrachtet werden müssen. Diese Fehlermodelle sind abhängig von der

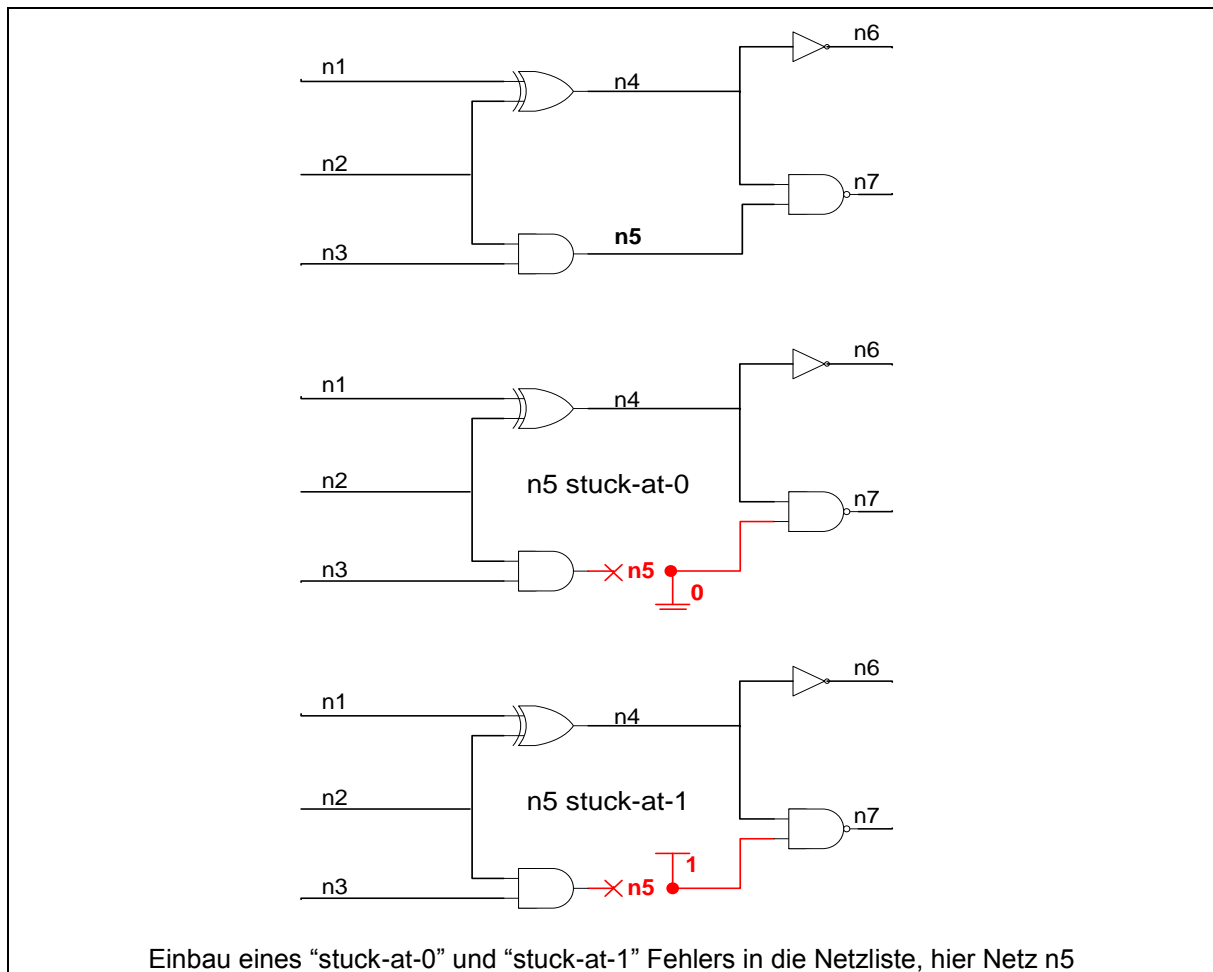
geforderten Sicherheitsstufe (safety integrity level SIL) der CPU (siehe Referenz [1], Teil 2, Tabelle A.1).

niedrige Testabdeckung (DC) (60%) für SIL1	mittlere Testabdeckung (DC) (90%) für SIL2	hohe Testabdeckung (DC) (99%) für SIL3
Stuck-at fault model: <ul style="list-style-type: none"> <li>• Stuck-at faults</li> </ul>	DC (direct current) fault model: <ul style="list-style-type: none"> <li>• Stuck-at faults</li> <li>• Stuck-open faults</li> <li>• Open circuits</li> <li>• Short circuits</li> <li>• High-Z connections</li> </ul>	DC (direct current) fault model: <ul style="list-style-type: none"> <li>• Stuck-at faults</li> <li>• Stuck-open faults</li> <li>• Open circuits</li> <li>• Short circuits</li> <li>• High-Z connections</li> </ul>

Fehlermodelle der CPU, abhängig von der geforderten Sicherheitsstufe (SIL)

„stuck-at“ Fehler lassen sich für die Fehlersimulation leicht modellieren. Um einen „stuck-at“ Fehler in die CPU Core Netzliste einzubauen, muss die Netzliste wie folgt manipuliert werden.

1. Auswahl eines Netzes, wo der „stuck-at“ Fehler eingebaut werden soll (z.B. Netz n5)
2. Aufbrechen des ursprünglichen (fehlerfreien) Netzes, d.h. den Ausgang der treibenden Zelle vom Eingang der empfangenden Zellen lösen
3. Anlegen einer logischen “0” oder “1” an den Eingang der empfangenden Zelle / Zellen



### Einbau eines "stuck-at" Fehlers

Wenn nun sowohl die Original Netzliste als auch die manipulierten Netzlisten simuliert werden, kann deren Verhalten unterschiedlich sein, abhängig von den beiden Aspekten:

- **Steuerbarkeit (controllability)** – beinhaltet der Simulationslauf Eingangsmuster (Stimuli), die bewirken, dass das fehlerhafte Netz invers zum eingepprägten Wert gesteuert wird ? (z.B. in obigem Bild mit Netz n5 „stuck-at-0“: Eingänge n2="1" und n3="1" so dass n5="1" durch das UND-Gatter gesteuert wird, was invers zum eingepprägten „stuck-at-0“ ist)
- **Beobachtbarkeit (observeability)** – beinhaltet der Simulationslauf Eingangsmuster (Stimuli), die bewirken, dass sich der falsche Wert bis zu einem Ausgang oder internen Register fortpflanzt, welches man beobachten kann ? (z.B. in obigem Bild mit Netz n5 „stuck-at-0“: Eingänge n2="1" und n1="1", so dass n4="1" durch das XNOR Gatter gesteuert wird. Dies erlaubt dem fehlerhaften Wert verursacht durch den „stuck-at-0“ Fehler in n5, sich durch das NAND Gatter zum Ausgang n7 fortzupflanzen (fehlerhaft n7="1" statt fehlerfrei n7="0")

Für einen erfolgreichen Test muss ein Testmuster erzeugt werden, welches sowohl die Steuerbarkeit als auch die Beobachtbarkeit des eingebauten Fehlers im

beobachteten Netz ermöglicht (hier „stuck-at-0“ im Netz n5). Ein anderes Testmuster muss erzeugt werden, um den eingebauten inversen Fehler (hier „stuck-at-1“ im Netz n5) im gleichen Netz erkennbar zu machen. Schließlich sind viele weitere Testmuster erforderlich, um alle eingebauten „stuck-at-0“ und „stuck-at-1“ Fehler in allen anderen Netzen der Netzliste erkennbar zu machen. Offensichtlich entwickelt sich dies zu einer sehr schnell anwachsenden Aufgabe anhand folgender Parameter:

- **Testaufwand:** größere Netzlisten mit einer steigenden Anzahl von Gattern und Netzen – mehr Testmuster sind nötig, um alle Netze abzudecken
- **Steuerbarkeit:** reduzierte Steuerbarkeit aufgrund der begrenzten Anzahl von Eingängen und steuerbaren Registern – die Steuerbarkeit wird zunehmend schwieriger
- **Beobachtbarkeit:** weniger Beobachtungspunkte im Verhältnis zur Anzahl der Gatter oder Netze aufgrund der begrenzten Anzahl der beobachtbaren Register – die Beobachtbarkeit wird zunehmend schwieriger
- **Tiefer geschachtelte Logik** mit mehreren Logikebenen zwischen den möglichen Steuerungspunkten und den Beobachtungspunkten – die Steuerbarkeit und die Beobachtbarkeit wird zunehmend schwieriger

Für den TriCore CPU Core mit mehreren hunderttausend Logikgattern, der begrenzten Anzahl an möglichen Steuerungseingängen und nur ein paar hundert Registern, die für Beobachtungszwecke im normalen Betrieb zur Verfügung stehen, bedeutet dies in der Tat eine sehr große Herausforderung. Die Antwort hierauf ist eine große Sammlung von hoch optimierten Testmustern, die zusammen das FSBST Testprogramm darstellen. Das FSBST Testprogramm kombiniert tausende von einzelnen Testmustern, von denen jedes einzelne jeweils viele Netze parallel steuert und beobachtet. Die Ausführung eines kompletten FSBST Testprogramms umfasst in der Größenordnung von  $10^5$  CPU Taktzyklen.

#### ***VIII.5.2.3. Simulierte Ausführung eines FSBST Testmusters auf der TriCore CPU Netzliste***

Wie im vorherigen Kapitel beschrieben, kann der Fehlereinbau und die Fehlersimulation genutzt werden, um nachzuweisen, dass ein spezifischer Fehler in einem einzelnen Netz durch ein einziges Testmuster entdeckt werden kann. Die gleiche Methode kann man anwenden, um zu berechnen, welcher Anteil aller möglichen Fehler (z.B. „stuck-at-0“ oder „stuck-at-1“ Fehler) in allen relevanten Netzen innerhalb der TriCore CPU durch ein vollständiges FSBST Testprogramm erkannt werden können. Dies geschieht in 4 Schritten:

1. Vorbereitung der Fehlerlisten
2. Fehlerfreier Testlauf mit der Original-Netzliste („golden simulation run“)
3. Mehrfache Simulation mit Fehlereinbau in die Netzliste (nur ein Fehler pro Durchlauf)
4. Statistische oder deterministische Berechnung der Erkennungsrate aus den Ergebnissen aus Schritt 3

Die Schritte 1 bis 4 werden in den folgenden Kapiteln näher beschrieben.

#### **VIII.5.2.4. Vorbereitung der Fehlerlisten**

Bevor mit dem Fehlereinbau und dem Fehlersimulationslauf begonnen wird, ist es vorteilhaft, die Anzahl der Fehler, die simuliert werden muss, zu reduzieren. Dies reduziert den Simulationsaufwand und die Simulationszeit. Nicht alle Netze sind relevant für den sicherheitskritischen Betrieb der CPU in der Systemumgebung. Daher kann das Element „Sicherheitsfunktion“ zum Beispiel wie folgt bestimmt werden:

- Fest verbundene Eingänge während des normalen Betriebs
- Feste Registerwerte während des normalen Betriebs
- Logik die während des normalen Betriebs unbenutzt ist (z.B. Test oder Debug Logik)
- Logik die durch andere Tests abgedeckt wird (z.B. Speicher Interface)
- Andere Methoden

Diese „sicheren Fehler“ werden identifizieren mittels Review durch TriCore CPU oder System Design Experten. Diese Teile werden aus der Fehlerliste entfernt (Netzlisten und zu simulierende Fehlerlisten). Im Folgenden werden nur die relevanten Netze und Fehler simuliert, die nicht bereits als „sichere Netze“ oder „sichere Fehler“ identifiziert wurden.

#### **VIII.5.2.5. Fehlerfreier Testlauf (golden simulation run)**

Während des fehlerfreien Testlaufs (golden simulation run) wird das FSBST Testprogramm auf der Original-Netzliste der TriCore CPU betrieben. Die beobachteten Testergebnisse (Registerinhalte zu bestimmten Testzeitpunkten, zusammengefasst in einer 32-bit CRC Prüfsumme) stellen das normale Verhalten der CPU ohne jeden Fehler dar. Die Ergebnisse des „golden simulation run“ werden gespeichert als Referenzwert, um sie gegen die Ergebnisse von späteren Simulationsläufen mit Fehlereinbau in der Netzliste zu vergleichen.

#### **VIII.5.2.6. Mehrfache Simulation mit Fehlereinbau**

Wenn die beiden Ergebnisse sich in mindestens einem Punkt unterscheiden, war das FSBST Testprogramm in der Lage, den eingebauten Fehler erfolgreich zu entdecken. Um dies zu überprüfen, wird ein einzelnes Netz ausgewählt und ein „stuck-at-0“ oder „stuck-at-1“ Fehler wird in das Netz eingebaut (nur ein einziger Fehler gleichzeitig für jeden Simulationslauf). Das Ergebnis des FSBST Testlaufs auf der fehlerhaften Netzliste wird gespeichert und verglichen mit dem Referenzwert aus dem „golden simulation run“.

Wenn die beiden Ergebnisse sich in mindestens einem Punkt unterscheiden, war das FSBST Testprogramm in der Lage, den eingebauten Fehler erfolgreich zu entdecken. Das gleiche gilt, wenn der eingebaute Fehler zu ernsthafteren Vorgängen während des FSBST Testlaufs führt (z.B. trap, program counter Sprung, kompletter CPU freeze), die sicher durch andere Einrichtungen erkannt werden (z.B. trap handler, Fenster Watchdog), dann gilt der Fehler als erkannt. Nur wenn sich das FSBST Testprogramm völlig normal verhält und exakt das gleiche Ergebnis wie beim „golden simulation run“ erzeugt, gilt der Fehler als nicht entdeckt.

Dieser oben beschriebene Ablauf, ein Netz auswählen, einen Fehler einbauen, das komplette FSBST Testprogramm laufen lassen, das Testergebnis gegen die Referenz aus dem „golden simulation run“ vergleichen, entscheiden ob der Fehler erkannt wurde oder nicht, muss mehrfach wiederholt werden. Wenn eine vollständige Simulation durchgeführt werden soll, müssen alle Netze des TriCore CPU Cores (mehrere hunderttausend Netze) zweimal ausgewählt werden, um jeweils einen „stuck-at-0“ und „stuck-at-1“ Fehler einzubauen. Dies führt zu fast einer Million Simulationsdurchläufe, jeder Simulationslauf erstreckt sich über  $10^5$  CPU Taktzyklen. Selbst wenn ein statistischer Ansatz gewählt würde, der weniger als 10% aller Netze und Fehler abdeckt, wäre der Simulationsaufwand immer noch riesengroß. Die Simulation würde eine Serverfarm für mehrere Tage oder Wochen vollständig auslasten (abhängig davon ob die Simulation auf Register-Transfer-Level oder auf Gatterebene durchgeführt wird).

#### **VIII.5.2.7. Berechnung der Testabdeckung (diagnostic coverage)**

Nachdem alle Simulationsläufe vollständig durchgeführt wurden, kann die Erkennungsrate für das FSBST Testprogramm berechnet werden. Hierfür werden die Ergebnisse des Simulationslaufs in 4 Gruppen eingeteilt:

- **DD:** Anzahl der Simulationsläufe, bei dem ein gefährlicher Fehler entdeckt wurde
- **DU:** Anzahl der Simulationsläufe, bei dem ein gefährlicher Fehler nicht entdeckt wurde
- **SD:** Anzahl der Simulationsläufe, bei dem ein sicherer Fehler entdeckt wurde
- **SU:** Anzahl der Simulationsläufe, bei dem ein sicherer Fehler nicht entdeckt wurde

Ein Fehler wird als gefährlich betrachtet, wenn er in ein Netz eingebaut ist, das für den normalen Betrieb der CPU notwendig ist, somit zum Element „Sicherheitsfunktion“ gehört. Ein Fehler wird als sicher betrachtet, wenn er in ein Netz eingebaut ist, das nicht für den normalen Betrieb der CPU notwendig ist, somit nicht zum Element „Sicherheitsfunktion“ gehört (z.B. Test oder Debug Logik). Diese Unterscheidung ist notwendig, sie erfordert ein detailliertes Review und Expertenwissen für die TriCore CPU Entwicklung.

Netze oder Fehler, die als sicher gelten, können von der Fehlerliste und den Simulationsläufen ausgeschlossen werden. Die Unterscheidung zwischen entdeckbar und nicht entdeckbar ist für sichere Fehler entbehrlich. Beide Fehlergruppen SD und SU können in der Variable S zusammengefasst werden.

$$S = SD + SU$$

S stellt die Anzahl der Simulationsläufe dar, deren Fehler als „sicher“ eingestuft sind.

In der Norm IEC61508 werden zwei verschiedene Fehlererkennungsraten definiert, die wir nutzen können, um die FSBST Testmuster und die Simulationsergebnisse auf Basis der oben genannten Fehlerkategorien zu bestimmen:

$$(1) \quad DC = \frac{DD}{DD + DU} \quad \text{Testabdeckung (Diagnostic Coverage DC)}$$

$$(2) \quad SFF = \frac{SD + SU + DD}{SD + SU + DD + DU} \qquad SFF = \frac{S + DD}{S + DD + DU}$$

Anteil "sichere Fehler" (Safe Failure Fraction SFF)

Abhängig vom geforderten Sicherheitsziel (SIL level) muss das FSBST Testprogramm eine niedrige (60%), mittlere (90%) oder eine hohe (99%) Testabdeckung erreichen.

#### VIII.5.2.8. DC-Fehlermodelle nach IEC61508 für SIL2 und SIL3

Wie in der Tabelle von VIII.5.2.2 gezeigt, ist es für höhere Sicherheitsstufen (SIL level) notwendig, über die einfachen „stuck-at“ Fehler hinaus zu schauen. Das DC Fehlermodell, welches die Norm IEC61508 für SIL2 und SIL3 vorschreibt, beinhaltet die folgenden 5 Fehlermodelle:

- Stuck-at Fehler
- Stuck-open Fehler
- Unterbrechung (Open circuit)
- Kurzschluss (Short circuit)
- Hochohmige Verbindung (High-Z)

Es ist nur sehr schwer oder gar nicht möglich, die anderen Fehler in einer ähnlichen Weise, wie oben für die „stuck-at“ Fehler beschrieben, zu simulieren. Dies hat folgende Gründe:

„**Stuck-open**“ und **Unterbrechung** bewirken, dass ein Ende des Netzes (Ausgang oder Eingang) unterbrochen ist. Die Spannung des Netzes wird nicht mehr durch den Ausgang der treibenden Zelle bestimmt, sondern wird durch viele Faktoren bestimmt und kann jeden analogen Wert annehmen. Der Eingang der von diesem „schwebenden Netz“ getrieben wird, verhält sich nicht mehr wie ein typisches digitales Logikgatter. Dieses analoge Verhalten kann nicht mehr mit einem digitalen Logik-Simulator nachgebildet werden, wie er für die Simulation auf Register-Transfer-Level oder Gatterebene Netzliste verwendet wurde. Stattdessen wäre ein analoger Fehlersimulator auf Transistor-Ebene erforderlich. Leider ist dies aufgrund des Umfangs und der Komplexität des TriCore CPU Designs sowie der nötigen Simulationszeit nicht realisierbar (siehe oben, Simulationszeit bei rein digitaler Simulation).

Um **Kurzschlüsse** in einer sinnvollen und aussagekräftigen Weise zu modellieren, wäre es notwendig, Informationen über die physikalische bzw. geographische Lage des Netzes zu bekommen, um benachbarte Netze für potenzielle Kurzschlüsse zu finden. Selbst wenn es möglich wäre, aus dem realen physikalischen Chip Layout diese Informationen zu extrahieren, um die Informationen in die Netzliste für eine Fehlersimulation zurückzuspiegeln, würde der Zeitaufwand für diese Prozedur mehrere praktische Grenzen sprengen. Abgesehen davon, jedes Netz hat normalerweise mehrere andere benachbarte Netze (Parallel-Strukturen oder Kreuzungen) und wäre daher potenzielle Gegenstücke für Kurzschlüsse. Dies bedeutet, dass die Anzahl aller möglichen Kurzschlüsse zwischen benachbarten Elementen ein Vielfaches der Anzahl aller „stuck-at“ Fehler in einer Netzliste ist. Dies würde, selbst bei Beschränkung auf eine statistisch geringe Aussagekraft von 10%,



jede praktisch realisierbare Grenze aufgrund der benötigten Anzahl an Simulationsläufen und der benötigten Simulationszeit sprengen (siehe oben, Simulationszeit-Betrachtung).

**Hochohmige Verbindungen (High-Z)** in einem Netz führen zu einer Verringerung der Flankensteilheit oder zu Laufzeitverzögerungen oder zu veränderten oder floatenden Spannungspegeln am Eingang. In CMOS Designs wie beim TriCore CPU Core sind floatende Spannungspegel nur eine zeitlich vorübergehende Erscheinung. Für eine korrekte Modellierung wäre auch hier eine analoge Simulation notwendig. Diese analoge Simulation ist aus Aufwandsgründen ausgeschlossen, wie bereits oben beschreiben im Falle des offenen Eingangs. Laufzeitverzögerungen könnten auf Gatterebene als Netzliste simuliert werden, jedoch nur auf einem hohen Abstraktionsgrad. Trotzdem wäre auch hierfür der Simulationsaufwand enorm hoch. Darüber hinaus macht es die analoge Natur einer ansteigenden Impedanz (von 0 Ohm bis unendlich d.h. Unterbrechung) erneut unmöglich, eine umfassende oder auch nur eine statistisch aussagekräftige Simulation in einem zeitlich realisierbaren Rahmen durchzuführen.

#### ***VIII.5.2.9. Fehlersimulation mit Mehrfach-Erkennung (N-Detect Fault Simulation)***

Aufgrund aller oben genannten realen Begrenzungen wird eine andere Art der Simulation ausgewählt, die **Mehrfach-Erkennung Fehlersimulation (N-Detect Fault Simulation)**. Angetrieben durch eine ähnliche Motivation, die Automatische Testmuster Generierung (ATPG) über die reine Erkennung von „stuck-at“ Fehlern auf Gatterebene zu erweitern, wurden in der Literatur (siehe Referenzen [3] und [4] und [5]) seit einiger Zeit N-detect Testmuster diskutiert und bereits erfolgreich in Testeinrichtungen der Produktion eingesetzt.

In beiden Fällen ist die grundlegende Idee hinter der N-Detect Fehlersimulation und N-Detect Testeinrichtung die folgende: Wenn ein Testmuster einen bestimmten „stuck-at“ Fehler mehr als einmal erkennt, (z.B. N-mal), verbessert es nicht die Fehlererkennungsrate (da der „stuck-at“ Fehler bereits beim ersten Testlauf erkannt wurde), aber er kann die Wahrscheinlichkeit erhöhen, andere Fehler im gleichen Netz oder in der gleichen Logikzelle zu finden.

Für einen N-Detect Testlauf im Produktionstest ist der Hauptgrund, Fehler innerhalb der Logikzelle (z.B. auf Transistor-Ebene) dadurch zu finden, dass man mehrere verschiedene Testmuster anlegt, die alle den gleichen „stuck-at“ Fehler auf Gatterebene finden. Der Hauptgrund für unsere N-Detect Fehlersimulation geht in eine etwas andere Richtung, kommt aber quantitativ zum gleichen Ergebnis (siehe nachfolgende Tabelle). Für eine N-Detect Fehlersimulation ist es entscheidend, den gleichen „stuck-at“ Fehler mehrfach, d.h. zu verschiedenen Zeitpunkten während des Testlaufs, zu erkennen, um andere Fehlermodelle im gleichen Netz auf Gatterebene zu erkennen. Näheres hierzu in den folgenden Kapiteln.

### Vergleich von N-Detect Fehlersimulation und N-Detect Produktionstest

	<b>N-Detect Fehlersimulation</b>	<b>N-Detect Produktionstest</b>
Zusatzaufwand	Mehrfache Erkennung von „stuck-at“ Fehlern zu verschiedenen Zeitpunkten im Testlauf	Mehrfache Erkennung von „stuck-at“ Fehlern mit verschiedenen Testmustern im Testlauf
Gewinn an Testabdeckung	Testabdeckung von „stuck-at“, „open“, „short“ Fehlern im gleichen Netz auf Gatterebene	Testabdeckung von „stuck-at“, „open“, „short“ Fehlern auf tieferer Transistorebene

#### **VIII.5.2.10. Abdeckung von „stuck-open“ und „open“ Fehlern durch N-Detect Simulation**

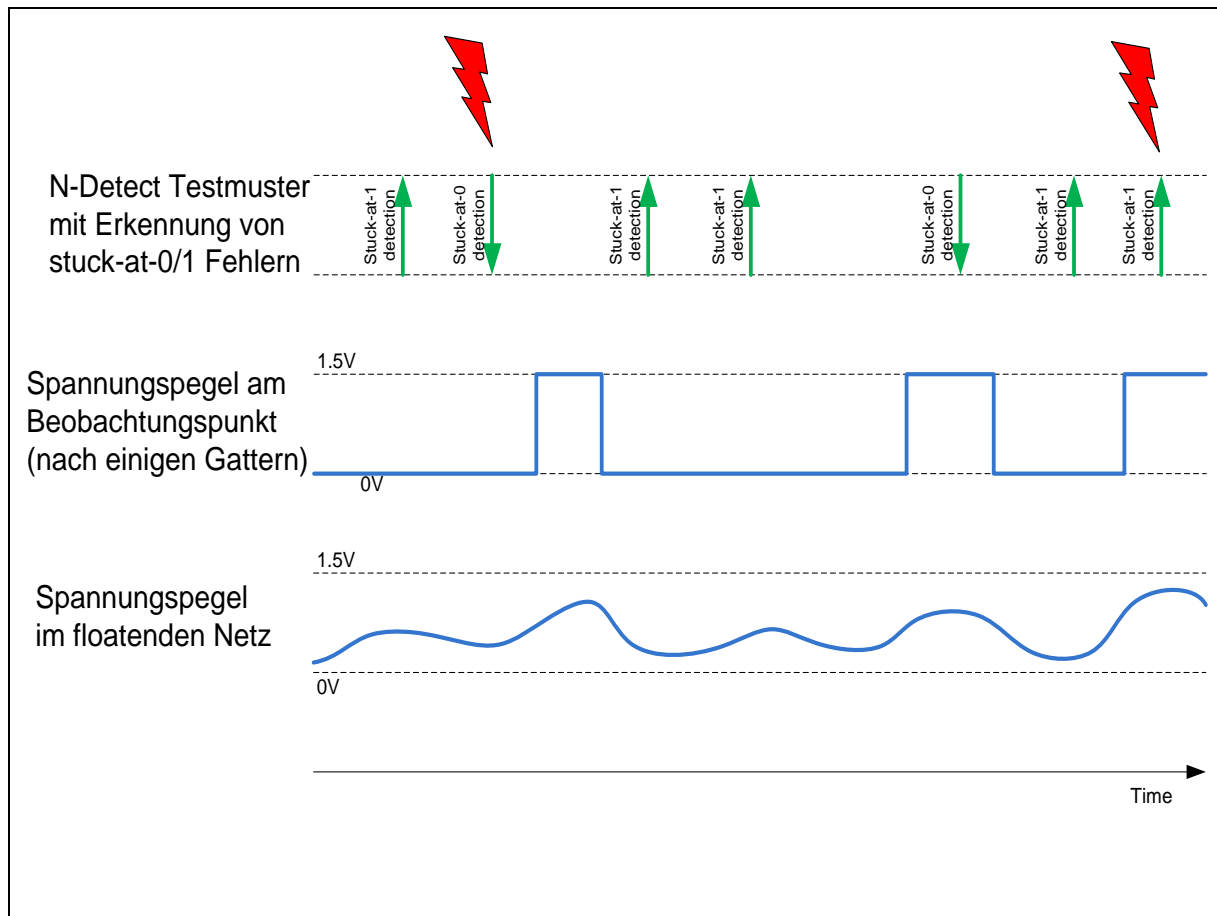
Wie oben beschrieben, bewirken „stuck-open“ und „open“ Fehler (nachfolgend nur noch mit „open“ Fehler bezeichnet) ein offenes Netz, welches die empfangende Zelle (Eingang) „treibt“. Der Spannungspegel in diesem offenen Netz ist grundsätzlich unbekannt, wird jedoch eine der drei möglichen Zustände einnehmen:

Fall 1: Driften in Richtung low level und dort bleiben (z.B. hochohmige Entladung im Netz)

Fall 2: Driften in Richtung high level und dort bleiben (z.B. hochohmige Aufladung im Netz)

Fall 3: Dauerhaftes Floaten der Spannung zwischen low level und high level (z.B. verursacht durch kapazitive Kopplung zwischen benachbarten Netzen)

In einer digitalen Schaltung wie die TriCore CPU nimmt selbst ein floatendes Netz nicht immer einen sauberen low level oder high level ein (z.B. 0,0V oder 1,5V). Die hoch verstärkende Charakteristik von CMOS Gattern bewirkt dass die empfangende Zelle zügig diese Zwischenzustände am Eingang in saubere Logikpegel am Ausgang umwandelt. Das bedeutet im ersten und zweiten Fall, wo das offene Netz in Richtung low level oder high level driftet, der „open“ Fehler sieht am Ausgang genau aus wie ein permanenter „stuck-at-0“ oder „stuck-at-1“ Fehler. Wenn das offene Netz tatsächlich dauerhaft nach oben und nach unten floatet (unteres Signal), wie im dritten Fall, dann sieht es, nach einem oder mehreren Logikgattern weiter im Signalpfad, wie ein zufällig hin- und her- schaltender digitaler Ausgang aus. Dieser Ausgang (Beobachtungspunkt) kann digital erkannt werden (mittleres Signal).



### Erkennung eines "open" Fehlers durch ein N-Detect Testmuster

Der Spannungspegel im floatenden Netz (unteres Diagramm) wird umgewandelt in einen zufällig schaltenden Ausgang in einem Netz ein paar Logikebenen weiter im Signalpfad. Dieser Ausgang dient als Beobachtungspunkt (mittleres Diagramm). Ein Testmuster, das „stuck-at-0“ und „stuck-at-1“ Fehler im fehlerhaften Netz erkennen kann, wird mehrfach eingespeist und erkennt manchmal (wenn das richtige Testmuster zum richtigen Zeitpunkt eingespeist wird) den fehlerhaft schaltenden Ausgang (oberes Diagramm) (hier „roter Blitz“).

Die Fälle 1 und 2 der „stuck-open“ und „open“ Fehler sind direkt mit Testmustern erkennbar, welche „stuck-at-0“ und „stuck-at-1“ Fehler erkennen können (kein N-Detect nötig). Der Fall 3 (mit der Wirkung eines zufällig hin- und her- schaltender Ausgangs einige Logikgatter später) kann mit steigender Wahrscheinlichkeit erkannt werden, je öfter das Testmuster eingespeist wird (siehe Abbildung oben, nicht bei jedem Testdurchlauf wird der zufällige Fehler erkannt).

Wenn wir annehmen, dass es keine starke Korrelation gibt zwischen dem floatenden Netz (Fehlernetz) und den Netzen zur Einspeisung der Testmuster (Steuerung) und zur Entdeckung des Fehlers (Beobachtbarkeit), dann ist die Wahrscheinlichkeit für die Entdeckung bzw. Nicht-Entdeckung des „zeitlich zufälligen“ „stuck-at-0“ oder „stuck-at-1“ Fehlers jeweils 50%. Diese Entdeckungs-Wahrscheinlichkeit gilt für jeden Durchlauf des Testmusters, selbst wenn es sich immer um das gleiche Testmuster handelt. Es ist nur nötig, Testmuster einzuspeisen, die „stuck-at-0“ und „stuck-at-1“

Fehler erkennen können, jedoch ist es nicht nötig, unterschiedliche Testmuster einzuspeisen. Allein die zeitlich zufällige Einspeisung der Testmuster und somit Entdeckung zu zeitlich zufälligen Zeitpunkten genügt.

Allgemein gilt: Bei N-maliger zeitlich zufälligen Einspeisung und Entdeckung ergibt sich die Wahrscheinlichkeit der Nicht-Entdeckung zu  $(50\%)^N$  oder der Entdeckung zu  $1 - (50\%)^N$ . Wird das gleiche Testmuster N-malig zeitlich zufällig eingespeist, so errechnet sich folglich:

$$(3) \quad p_{float} = 1 - 2^{-N} \quad \text{Wahrscheinlichkeit der Entdeckung eines „floating open“ Fehlers}$$

Wesentlich ist, dass die Erkennungs-Wahrscheinlichkeit mit der Anzahl der Testdurchläufe steigt, solange sie zeitlich zufällig zum Fehler eingespeist und ausgewertet werden. Bitte beachten, die Formel (3) gilt nur für „floating open“ Fehler der Kategorie 3.

Alle anderen „floating open“ Fehler (Kategorie 1 und 2) werden zu 100% beim ersten Durchlauf des geeigneten Testmusters erkannt (vorausgesetzt das eingespeiste Testmuster kann sowohl einen „stuck-at-0“ wie einen „stuck-at-1“ Fehler erkennen). In der Realität erwarten wir, dass die meisten „floating-open“ Fehler in die Kategorie 1 oder 2 fallen. Trotzdem (weil es keine gesicherten Daten zum Thema gibt) nehmen wir eine Gleichverteilung aller 3 Kategorien an. Diese Annahme ist pessimistisch und somit zulässig, die Realität ist wahrscheinlich besser. Somit kann die Gesamtwahrscheinlichkeit für die Entdeckung aller „stuck-open“, „ und „open“ Fehler berechnet werden zu:

$$(4) \quad p_{opens} = \frac{1}{3} \cdot d_{SA0} + \frac{1}{3} \cdot d_{SA1} + \frac{1}{3} \cdot (1 - 2^{-N}) \quad \text{Entdeckungs-Wahrscheinlichkeit für „open“}$$

$$(5) \quad d_{SA0} = (\text{stuck-at-0 Fehler erkannt}) ? 1:0 \quad \text{Entdeckung für „stuck-at-0“ Fehler}$$

$$(6) \quad d_{SA1} = (\text{stuck-at-1 Fehler erkannt}) ? 1:0 \quad \text{Entdeckung für „stuck-at-1“ Fehler}$$

Anmerkung: Entdeckungs-Wahrscheinlichkeit für „stuck-at-0/1“ ist entweder 100% oder 0%.

#### **VIII.5.2.11. Abdeckung von „short“ Fehler durch N-Detect Fehlersimulation**

Ähnlich wie bei den offenen Schaltkreisen (siehe vorheriges Kapitel) ist der Spannungspegel oder der logische Wert in einem Netz mit einem Kurzschluss generell unbekannt, aber es wird einen der folgenden fünf Zustände annehmen:

1. Kurzschluss nach GND, daher hat dieses Netz einen „stuck-at-0“ Fehler
2. Kurzschluss nach VDD, daher hat dieses Netz einen „stuck-at-1“ Fehler
3. Kurzschluss zum Nachbarnetz mit einer stärkeren treibenden Zelle im Nachbarnetz, daher folgt dieses Netz dem logischen Zustand des Nachbarnetzes
4. Kurzschluss zum Nachbarnetz mit einer schwächeren treibenden Zelle im Nachbarnetz, daher folgt das Nachbarnetz dem logischen Zustand dieses Netzes

5. Kurzschluss zum Nachbarnetz mit einer gleich starken treibenden Zelle, daher schaltet dieses Netz und das Nachbarnetz unvorhersehbar hin und her, ist eventuell zeitweilig floatend in Zwischenzuständen zwischen den sauberen Spannungspegeln

Die Fälle 1 und 2 sind per Definition „stuck-at-0“ oder „stuck-at-1“ Fehler und können direkt mit den entsprechenden Testmustern erkannt werden. Tatsächlich müssen diese Fehler nicht als Kurzschlüsse, sondern als „stuck-at-0“ oder „stuck-at-1“ Fehler gezählt werden.

Die Fälle 3 und 4 und 5 sind sehr ähnlich, sie sind alle vergleichbar mit den „floating open“ Fehlern aus dem vorhergehenden Kapitel. Fall 3 ist identisch zu Fall 4, lediglich aus der umgekehrten Sichtweise, wo das betrachtete Netz das stärkere (dominante) Netz ist und das Nachbarnetz das schwächere (unterlegene) Netz ist. Fall 5 kann man betrachten als laufend wechselnd zwischen Fall 3 und Fall 4, mit zeitweiligen Phasen von Zwischenzuständen zwischen den Spannungspegeln, wie unter „floating open“ im Vorgängerkapitel beschrieben. In der Literatur wird der Fall 5 manchmal als „wired AND“ oder „wired OR“ modelliert, was eine vereinfachte Sichtweise darstellt, aber in beiden Fällen zum gleichen Ergebnis führt.

Während beim „floating-open“ Fehler aus Fall 3 im Vorgängerkapitel die Ursache für das „zufällige“ Schalten eine Unterbrechung und eine kapazitive Kopplung des fehlerhaften Netzes mit dem Nachbarnetz ist, handelt es sich hier um direktes Treiben bedingt durch einen Kurzschluss. In beiden Fällen ist die Wirkung die gleiche – das „zufällige“ Schalten kann entweder direkt im Netz oder einige Gatter später im Signalpfad erkannt werden. Die Wahrscheinlichkeit der Erkennung steigt auch hier mit der Anzahl der zeitlich zufällig eingespeisten Testmuster für „stuck-at-0“ und „stuck-at-1“ Fehler wie oben beschrieben. Die Erkennungs-Wahrscheinlichkeit für Kurzschlüsse, gleich welcher Art, durch Testprogramme, welche „stuck-at-0“ und „stuck-at-1“ erkennen, ist somit bei N-maliger Wiederholung:

$$(7) \quad p_{shorts} = 1 - 2^{-N} \quad \text{Erkennungs-Wahrscheinlichkeit für Kurzschlüsse (Fall 1 bis 5)}$$

#### **VIII.5.2.12. Abdeckung von hochohmigen Verbindungen (High-Z)**

Leider sind N-Detect Fehlersimulationen nicht geeignet, erhöhte zeitliche Verzögerungen oder längere Durchlaufzeiten verursacht durch hochohmige Verbindungen (High-Z) zu erkennen. Stattdessen können Versuche mit reduzierter Versorgungsspannung und / oder mit erhöhter Taktfrequenz durchgeführt werden, um erhöhte Verzögerungen oder längere Durchlaufzeiten nachzubilden. Gleichzeitig kann die Wirksamkeit des eingesetzten FSBST Testprogramms nachgewiesen werden, solche Fehler zu erkennen, wenn diese auftreten.

#### **VIII.5.2.13. Berechnung der Testabdeckung (Diagnostic Coverage DC)**

Auf Grundlage der Darstellungen und Diskussionen aus den vorhergehenden Kapiteln ist es möglich, eine Fehlersimulation für das FSBST Testprogramm durchzuführen, ganz ähnlich zu den Berechnungen für die „stuck-at-0“ und „stuck-at-1“ Fehler in den früheren Kapiteln. Es werden ebenso die Gesamtanzahl  $N_i$  an „stuck-at“ Fehlerentdeckungen für jedes Netz  $i$  gespeichert Aus der Fehlerabdeckung

(diagnostic coverage DC) für Unterbrechungen und Kurzschlüsse kann man unter Verwendung der früheren Formeln (4) und (5) und (6) und (7) berechnen:

$$(8) \quad DC_{opens} = \frac{1}{S} \cdot \sum_{i=1}^S \left( \frac{1}{3} \cdot d_{SA0} + \frac{1}{3} \cdot d_{SA1} + \frac{1}{3} \cdot (1 - 2^{-N_i}) \right) \quad \text{DC für Unterbrechung}$$

$$(9) \quad DC_{shorts} = \frac{1}{S} \cdot \sum_{i=1}^S (1 - 2^{-N_i}) \quad \text{DC für Kurzschluss}$$

S Gesamtanzahl der Netze bzw. Fehler, die getestet wurden

$N_i$  Anzahl  $N_i$  der „stuck-at“ Fehlererkenntnisse für jedes einzelne Netz  $i$

$d_{SA0}$  = (stuck-at-0 Fehler erkannt) ? 1:0 Entdeckung für „stuck-at-0“ Fehler

$d_{SA1}$  = (stuck-at-1 Fehler erkannt) ? 1:0 Entdeckung für „stuck-at-1“ Fehler

Anmerkung: Entdeckungs-Wahrscheinlichkeit für „stuck-at-0/1“ ist entweder 100% oder 0%.

Die durchschnittliche Anzahl von „stuck-at“ Fehlerentdeckungen pro Netz berechnet sich zu:

$$(10) \quad N_{avg} = -\log_2 \left( 1 - \frac{1}{S} \cdot \sum_{i=1}^S (1 - 2^{-N_i}) \right) \quad \text{Durchschnittliche Anzahl an Fehlerentdeckungen}$$

Diese Zahl wird benutzt, um die Qualität des N-Detect FSBST Testprogramms quantitativ zu bestimmen.

### VIII.5.3. Betrachtete Testfälle

Dieses Kapitel definiert die Testfälle, die durchzuführen sind, um die Wirksamkeit und Fehlerfreiheit des eingesetzten FSBST Testprogramms nachzuweisen.

#### VIII.5.3.1. Funktionale Tests

Die funktionalen Tests laufen während dem normalen Betrieb der CPU (zyklischer Betrieb).

##### a. Überprüfung einer Konstanten, die vom FSBST erzeugt wird

Test Zweck: Überprüfung, ob der FSBST eine einzige und konstante 32-bit Signatur erzeugt, wenn er während des Normalbetriebs ausgeführt wird

Test Beschreibung: Zyklische Ausführung des FSBST als Teil der on-Chip Sicherheits-Software, integriert zusammen mit einer Referenz-Software oder der Applikations-Software (vom Kunden)

- Zyklische Ausführung des FSBST
- Laufender Vergleich des FSBST Ergebnis mit der 32-bit CRC Prüfsumme
- Dauerlauf für mind. 1 Stunde (d.h. 600.000 Zyklen FSBST)

##### b. Überprüfung der Nicht-Beeinflussung von bzw. durch andere Speicherbereiche

Test Zweck: Überprüfung, ob der FSBST nicht vom Rest des Systems beeinflusst wird / diesen nicht beeinflusst und dass der FSBST nur den zugewiesenen Speicherbereich benutzt. Dies stellt sicher, dass Daten in anderen Speicherbereichen des Systems durch den FSBST nicht zerstört oder verändert werden.

Test Beschreibung: Überwachung der Speicherzugriffe während dem „golden simulation run“ (Referenz).

- „golden simulation run“ auf der TriCore RTL Netzliste (ohne Fehlereinbau)
- Überwachung aller Speicherzugriffe während dem Simulationslauf
- Aktivieren einer Falle für einen Speicherzugriff außerhalb des erlaubten Bereichs
- Abspeichern einer Protokolldatei für alle ausgeführten Speicherzugriffe und Überprüfung gegen die Speicheraufteilung

##### c. Überprüfung der Nicht-Beeinflussung von bzw. durch andere Ressourcen

Test Zweck: Überprüfung, ob der FSBST nicht vom Rest des Systems beeinflusst wird / diesen nicht beeinflusst und dass der FSBST nur die ihm zugewiesenen Systemressourcen benutzt. Dies stellt sicher dass der Systemzustand nach dem Durchlauf des FSBST oder wenn der FSBST unterbrochen wird identisch mit dem Systemzustand vor dem Start ist.

Test Beschreibung: Vergleich des Systemzustands vor und nach dem „golden sample run“

- „golden simulation run“ auf der TriCore RTL Netzliste (ohne Fehlereinbau)
- Abspeichern des kompletten Systemzustands (alle internen Register) nach dem kompletten Durchlauf des „golden simulation run“
- Vergleich der gespeicherten Inhalte aller internen Register vor und nach dem FSBST Testlauf

### **VIII.5.3.2. Funktionstest für Betrieb während Hochlauf und Herunterfahren**

Die Anforderungen an den FSBST Testprogramm während des Hochlaufs und des Herunterfahrens des Systems sind identisch zu den Anforderungen bei normalem Betrieb. Daher sind keine speziellen Funktionstests für den Hochlauf und das Herunterfahren nötig.

### **VIII.5.3.3. Nicht-funktionale Anforderungen**

Diese Testgruppe überprüft alle nicht-funktionalen Anforderungen, die messbar sind. Alle nicht-funktionalen Anforderungen, die nicht messbar sind, werden durch Reviews oder ähnlichen Maßnahmen abgedeckt und sind nicht Bestandteil des FSBST Testprogramms.

#### **a. Überprüfung der maximalen Ausführungszeit FSBST von 500µs**

Test Zweck: Überprüfung, ob der FSBST nach max. Ausführungszeit von 500µs abgeschlossen ist, bei maximaler CPU Geschwindigkeit und mit Befehls-Cache eingeschaltet.

#### **b. Überprüfung der maximalen Interrupt Latenzzeit von 50µs**

Test Zweck: Überprüfung, ob die Interrupt Verzögerung, verursacht durch den FSBST, < 50µs ist.

### **VIII.5.3.4. Berechnung der Testabdeckung (Diagnostic Coverage DC)**

Die folgenden Tests haben das Ziel, die Testabdeckung des FSBST zu berechnen. Gemäß der Norm IEC61508 kann man die Fehlerabdeckung für eine Zentrale Prozessor Einheit CPU wie die TriCore CPU durch eine Fehlersimulation messen, unter folgenden Bedingungen:

- Es ist verbindlich vorgeschrieben, alle Fehler aus dem DC Fehlermodell zu betrachten, sofern eine mittlere oder hohe Fehlerabdeckung durch den Test beansprucht wird.
- Es wird empfohlen, die Fehlerabdeckung durch eine Fehlersimulation basierend auf der Netzliste auf Gatterebene zu berechnen, um die bestmögliche Übereinstimmung mit der physikalisch realisierten Implementierung zu bekommen.

Mit diesen beiden Anforderungen befassen sich die folgenden drei Schritte:

#### **a. Fehlerabdeckung für „stuck-at“ Fehler auf Register-Transfer-Level im Vergleich zur Fehlersimulation**

Fehlersimulation nur für „stuck-at“ Fehler basierend auf Register-Transfer-Level (RTL) Netzlisten ist einfach und läuft sehr schnell. Die RTL Netzliste bietet den zusätzlichen Vorteil, dass sichere Netze bzw. sichere Fehler von Experten des CPU Designs leicht identifizierbar sind und somit aus der Fehlerliste ausgeschlossen werden können. Daher wird zuerst eine gründliche Fehlersimulation für „stuck-at“ Fehler basierend auf der RTL Netzliste durchgeführt, um eine erste Zahl für die Fehlerabdeckung des FSBST zu erhalten.



### **b. Fehlerabdeckung für „stuck-at“ Fehler auf Gatterebene durch RTL Netzlisten im Vergleich zur Gatterebene Korrelation**

Wie in früheren Kapiteln beschrieben, ist es unmöglich, nicht alle sicheren Netze auf einer Gatterebene Netzliste zu identifizieren. Daher ist eine Simulation auf Gatterebene Netzliste stets weniger optimiert als eine Simulation auf RTL Netzliste (d.h. es bleiben mehr unentdeckbare aber sichere Netze bzw. sichere Fehler auf der Fehlerliste und werden simuliert). Dies führt unvermeidlich bei den Gatterebene Netzlisten zu geringeren Fehlerabdeckungen, was trotzdem nicht die Realität widerspiegelt.

Aus diesem Grund wird nicht versucht, die Fehlerabdeckung der TriCore CPU auf Basis der Gatterebene Netzliste direkt zu messen. Stattdessen werden zwei gleichwertige RTL Netzlisten und Gatterebene Netzlisten simuliert, beide auf Basis von zwei gleichwertig optimierten Fehlerlisten für beide Netzlisten (d.h. für die RTL Netzliste nicht optimal). Diese Korrelation benachteiligt nicht die Gatterebene Netzliste sondern führt zu einem realistischen Vergleich der FSBST Leistungsfähigkeit der beiden Netzlisten. Auf Basis dieser Korrelation wird die ursprüngliche Fehlerabdeckung aus Schritt 1 für „stuck-at“ Fehler auf RTL Ebene extrapoliert für „stuck-at“ Fehler auf Gatterebene Netzliste.

### **c. Fehlerabdeckung für DC Fehlermodelle auf Gatterebene im Vergleich zur N-Detect Fehlersimulation**

Wie in früheren Kapiteln beschrieben, wird zum Schluss eine zusätzliche N-Detect Simulation durchgeführt, um die Fehlerabdeckung durch den FSBST für „stuck-open“, „open“ und „short“ Fehler zu erhalten. Mit diesen zusätzlichen Maßnahmen wird die Fehlerabdeckung von reinen „stuck-at“ Fehlern auf die DC-Fehlermodelle auf Gatterebene ausgeweitet.

### **d. Bestimmung der Fehlerabdeckung auf Basis der RTL Netzliste**

Test Zweck: Bestimmung der Fehlerabdeckung durch das FSBST Testprogramm auf Basis der RTL Netzliste der TriCore CPU, wenn ausschließlich „stuck-at“ Fehler betrachtet werden.

Erwartetes Ergebnis: Fehlerabdeckung (RTL, „stuck-at“ Fehler) > 90%

### **e. Bestimmung der Fehlerabdeckung auf Basis der Gatterebene Netzliste**

Test Zweck: Überprüfung, ob die Fehlerabdeckung durch das FSBST Testprogramm vergleichbar gut ist wie auf RTL Basis oder auf Gatterebene Basis der TriCore CPU. Somit sind die Ergebnisse aus dem Test FSBST\_DC\_RTL auch übertragbar auf Gatterebene Basis der TriCore CPU, wenn ausschließlich „stuck-at“ Fehler betrachtet werden.

Erwartetes Ergebnis: Die Fehlerabdeckung auf Gatterebene Basis sollte gleich oder besser sein als die Fehlerabdeckung auf Basis der RTL Netzliste.  
Fehlerabdeckung (Gatterebene, „stuck-at“ Fehler) > 90%

#### **f. Bestimmung der Fehlerabdeckung auf Basis des DC-Fehlermodells**

Test Zweck: Nachweis, dass die Ergebnisse des Tests FSBST\_DC\_CORR übertragbar sind auf die Gatterebene Netzliste der TriCore CPU, wenn alle DC Fehlermodelle betrachtet werden

Erwartetes Ergebnis: Fehlerabdeckung (Gatterebene, DC-Fehlermodell) > 90%

#### **VIII.5.4. Ergebnisse und Erfahrungen aus der Aufgabe 2-4-1**

In diesem Kapitel werden die Ergebnisse und Erfahrungen aus Aufgabe 2-4-1 mit den Entwicklungen der Automobilindustrie der letzten drei Jahre reflektiert.

##### **VIII.5.4.1. Die Basisnorm IEC 61508 und die Sektornorm ISO 26262**

Infineon arbeitete im Bereich Microcontroller seit 2007 mit der IEC 61508 Norm Titel „Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme“. Diese Basisnorm wurde erstmals 1998 veröffentlicht und fand in der letzten Dekade eine hohe Akzeptanz in vielen Anwendungsfeldern. Der probabilistische Ansatz bei der Fehler und Gefahrenanalyse erwies sich als äußerst erfolgreich.

Die Methodik der IEC 61508 wurde in den Anwendungsfeldern von sogenannten Sektornormen übernommen und um anwendungsfeldspezifische Erweiterungen ergänzt.

##### **VIII.5.4.2. Die Sektornorm ISO26262**

Für den Bereich Automotive trat nach längerem Vorlauf am 14.11.2011 die ISO 26262 in Kraft. Die ISO 26262 stellt die Anpassung der IEC 61508 an die Bedürfnisse der Automobilindustrie dar. Beide Standards beinhalten Metriken zur Bewertung der Hardware bezüglich ihrer Systemsicherheit. Während die IEC 61508 die Berechnung des Anteils ungefährlicher Ausfälle, Safe Failure Fraction (SFF) genannt, zur Beurteilung der Qualität der Diagnoseabdeckung fordert, verlangt die ISO 26262 die Berechnung der Einzelfehlermetrik, die so genannte Single Point Faults Metric (SPFM) und die der Latent Fehler Metrik (Latent Fault Metric, LFM). Die Metriken berechnen den Anteil der sicheren Fehler und der gefährlichen detektierten Fehler im Verhältnis zu allen Fehlern.

In selben Maße wie die gesamte Automobilindustrie hat sich Infineon in den letzten drei Jahren im Bereich Microcontroller immer mehr auf die ISO 26262 fokussiert was sich an folgenden tiefgreifenden Maßnahmen erkennen lässt:

- Die Entwicklungsprozesse wurden entsprechend adaptiert.
- Die Organisation um ein Funktional Safety Management und ein Funktional Safety Engineering erweitert.
- Der Microcontroller wird nach ISO 26262 Teil 10 als sogenanntes Safety Element out of Context (SEooC) entwickelt.
- Das SEooC besteht aus einer Hardware- und einer Software-Komponente.

##### **VIII.5.4.3. Die Verbesserung der Diagnostic Coverage DC**

Die Integrität der CPU Recheneinheit sicherzustellen ist eines der zentralen Elemente des Microcontroller Sicherheitskonzeptes. Der zunächst von Infineon entwickelte Befehlssatztest basierte auf der klassischen Verifikation einzelner Prozessorinstruktionen in Software und war optimiert nur die Instruktionen zu testen welche in der Sicherheitsapplikation verwendet wurden. Ein spezieller Werkzeug (OTCG = Opcode Test Code Generator) extrahierte aus einem Objektcode der Applikation alle verwendeten Instruktionen und generierte aus Testvorlagen jeder Instruktion einen Gesamttest. Diese Vorgehensweise war notwendig, weil die TriCore

Prozessorarchitektur über 330 Instruktionen kennt und ein Gesamttest aller Instruktionen zu viel CPU Last erzeugt hätte. Heutige Compiler nutzen jedoch nicht alle Instruktionen, so dass mit dem OTCG ein Test erstellt werden konnte, der eine geringere CPU Last zur Folge hat. Bis vor kurzem wurde im Allgemeinen davon ausgegangen, dass für einen solchen Test ein DC von 90% angenommen werden kann. Bereits zu Beginn der genauen Analyse die auch im Zusammenhang mit der Aufgabe 2-4-1 auf Basis eines RTL Model durchgeführt wurde, stellte sich jedoch heraus das diese Art von Prozessortest einen deutlich geringen DC von etwa 60% erreicht.

Infineon entwickelte daraufhin einen neuen Prozessortest, den sogenannten Software Based Self Test (SBST) mit dem ein DC von über 90% erreicht werden kann. In Zusammenspiel mit weiteren Hardwaremaßnahmen lassen sich heute über 99% DC erreichen. SBST ist dabei der Infineon spezifische Name des Softwarepaketes das einen FSBST durchführt, so wie es in Aufgabe 2-4-1 ausführlich beschrieben ist. SBST ist somit ein Produktname für einen FSBST.

Der Deckungsgrad des SBST konnte mit der Methodik aus Aufgabe 2-4-1 vollständig nachgewiesen werden.

Die unzureichende DC von Selbsttests ähnlich dem OTCG wurde von Zertifizierungsstellen wie z.B. TÜV auch für andere Prozessoren in der Automobilindustrie erkannt. Die Methodik aus Aufgabe 2-4-1 wird deshalb inzwischen allgemein für die Zertifizierung gefordert und soll von allen Halbleiterherstellern der Automobilindustrie angewendet werden. Sie stellt damit den unter Sicherheitsaspekten maßgebenden aktuellen Stand der Technik dar.

Wie in 2-4-1 beschrieben hat Infineon diese Methodik bei der Single-Core, non-lockstep Architektur des TC1387 entwickelt und anschließend auf alle Produkte der AUDO-Future und AUDO-Max Familien mit der Prozessorarchitektur TC1.3.1 und TC1.6 angewendet.

#### ***VIII.5.4.4. RTL Netzliste als Entwicklungsbasis des FSBST***

Die Entwicklung der TriCore CPU findet auf der logischen RTL Ebene statt. Somit liegt das Logikdesign zunächst als RTL Darstellung vor. Die RTL Netzliste wird von den Schaltungsentwicklern „von Menschen-Hand“ erzeugt. Eine Netzliste auf Gatterebene wird aus der RTL Netzliste durch ein automatisches Logik-Synthese-Tool maschinell generiert.

Die sicherheitskritischen Pfade der TriCore CPU können von erfahrenen Schaltungs-Experten manuell identifiziert werden. Im Vergleich hierzu besteht die Maschinen erzeugte Netzliste auf Gatterebene aus vielen logischen Instanzen und Netzen, welche die eingebauten Funktionen des FSBST Programms und deren Auswirkungen nicht klar zeigen.

Auch die nötigen Simulationslaufzeiten für RTL und Gatterebene unterscheiden sich deutlich. Während ein kompletter Simulationslauf zur Bestimmung der „stuck-at“ Fehlererkennung in ca. 3 Tagen durchläuft, benötigt der gleiche Simulationslauf auf Gatterebene über 3 Wochen. Sowohl die Simulationslaufzeit als auch die logische Transparenz der Darstellung sprechen für die weitere Entwicklung und Optimierung des FSBST Testprogramms auf RTL Basis.

#### **VIII.5.4.5. FSBST Entwicklung als Kontinuierlicher Verbesserungs-Prozess (KVP)**

Betrachten wir die FSBST Entwicklung als einen kontinuierlichen Verbesserungs-Prozess:

1. Erfahrene Logik-Entwickler, in Zusammenarbeit mit Experten für Funktionale Sicherheit, sprechen das logische Design durch und identifizieren Hierarchie-Ebenen, welche die sicherheitskritischen Komponenten beinhalten. Eine erste Signal-Liste wird für das gesamte Design erstellt. Der sicherheitskritische Anteil wird extrahiert, dieser enthält immer noch nicht-sicherheitskritische Module. Diese Teile werden identifiziert und von der Master-Fehlerliste ausgeschlossen. Dieser Vorgang wird mehrfach wiederholt.
2. Logik-Entwickler markieren Signale mit einem beliebigen konstanten Wert oder Signale, die ignoriert werden können, weil sie nicht sicherheitsrelevant sind. Aus dieser Liste werden „stuck-at“ Fehler generiert, zum Ausschluss aus der Master-Fehlerliste.
3. Die Durchleuchtung der Firmware-Umgebung führt zu einer Liste von statischen Registern, welche zu einem Satz an Signalen auf Gatterebene führt, die garantiert konstant sind. Einige dieser Signale finden sich auf RTL Ebene wieder, diese bilden wiederum einen neuen Satz von „stuck-at“ Fehlern, die wir im Test ignorieren können.
4. Logik-Entwickler und Testingenieure durchleuchten und verbessern jeden einzelnen Schaltungsblock hinsichtlich der Fehlerabdeckung. Diese Ingenieure erzeugen und verbessern gezielt die Fehlerabdeckung einzelner Schaltungsbereiche. Diese optimierten Testmuster werden in das komplette FSBST Testprogramm integriert.
5. Ein Testmuster-Generator mit zufälligen Testmustern wird gezielt in jeden Schaltungsblock eingespeist, um neue erfolgreiche Testmuster-Abschnitte zu erzeugen, die wiederum in das komplette FSBST Testprogramm integriert werden können.
6. Zufällig erzeugte Testmuster für zufällige „stuck-at“ Fehler lässt man nun laufen, als Iterationsschleife während der FSBST Testentwicklung, um die Verbesserung / Verschlechterung der Testabdeckung durch diese zufälligen Testmuster zu bestimmen.
7. Aus diesem Regressionstest werden die erfolgreichen Testmuster extrahiert, mit den zugehörigen Netzen und Werten, die für die „stuck-at“ Fehlersimulation eingespeist wurden. Der Teil des Testmusters, der den Fehler zuerst entdeckt hat, wird extrahiert.
8. Andere sekundäre Testmuster-Statistiken werden ausgewertet, zusammen mit der Fehlerabdeckung pro Instanz und der Anzahl der entdeckten Fehler pro Schaltungsteil.
9. Nicht-entdeckte „stuck-at“ Fehler werden in einen Satz von formalen Eigenschaften zusammengefasst, um zu beweisen, dass sie konstant auf einem bestimmten „stuck-at“ Wert sind. Dies führt dazu, dass diese Fehler aus der Fehlerliste ausgeschlossen werden.
10. Die in jedem Entwicklungsschritt gefundenen und extrahierten erfolgreichen Testmuster werden in das existierende FSBST Testprogramm integriert.

Dieser Iterations-Ablauf wird so oft wiederholt, bis die geforderte Fehlerabdeckung erreicht wird.

#### **VIII.5.4.6. Die Fehlerdetektionszeit und Diagnosezyklus**

Ein wichtiges Maß welches indirekt aus den Normen IEC 61508 und ISO 26262 hervorgeht, ist die Zeit innerhalb derer in einem System ein Fehler erkannt werden muss.

Diese Fehlerdetektionszeit hängt sehr stark von der jeweiligen sicherheitskritischen Applikation ab, hat aber einen wesentlichen Einfluss auf die verwendete Methode zur Fehlererkennung.

ISO 26262 gibt Vorschriften, wie sichergestellt werden muss, dass auftretende Fehler keine der beschriebenen Schadensklassen verursachen können. Fehler müssen in einem beschriebenen Umfang vor dem Eintreten eines gefährlichen Ereignisses diagnostiziert und Gegenmaßnahmen ergriffen werden. Der Umfang der Diagnose richtet sich nach der Wahrscheinlichkeit mit der ein Fehler auftreten kann und der mit dem Fehler verbundenen Schadensklasse.

Mit ISO 26262 können nun die Diagnoseschritte in zwei Gruppen eingeteilt werden:

Die erste Gruppe von Tests beinhaltet alle jene, die nur einmal pro Fahrzyklus durchgeführt werden müssen.

Für die Auswirkung auf die Rechenleistung eines Systems entscheidender ist die zweite Gruppe von Tests. Das sind Diagnosen die zyklisch während des Betriebs eines Fahrzeugs durchgeführt werden müssen. Die Zykluszeit muss so bemessen werden, dass bei auftretenden Fehlern rechtzeitig reagiert werden kann.

Der Zyklus entspricht der Fehlerdetektionszeit und ist im Wesentlichen bestimmt durch die von der Applikation vorgegebenen maximalen Reaktionszeit und der Zeit die notwendig ist bis Gegenmaßnahmen wirken.

Die maximale Reaktionszeit  $t_R$  minus der Zeit für Gegenmaßnahmen  $t_G$  bestimmt den maximalen Zeitabstand  $t_{ZY}$  zwischen zwei Tests.

$$t_{ZY} = t_R - t_G$$

Ein fiktives Systembeispiel: Es dauert 1ms bis ein Kurzschluss ein Bauteil so überhitzt dass ein Brand entstehen kann. Es dauert 200µs um den Strom für das Bauteil abzuschalten. Es muss demnach alle 800µs der FSBST gestartet werden, um einen eventuellen Brand für den Fehlerfall Kurzschluss verhindern zu können. Die Zykluszeit beträgt 800µs.

#### **VIII.5.4.7. Reduktion der Rechenleistung durch Diagnose**

Die geforderte DC beeinflusst die Dauer der Diagnose. Entscheidend ist dabei die Zeit, für die die applikative Operation für den Test unterbrochen werden muss.



#### Zykluszeit und Fehlerdetektionszeit

Dauer der Unterbrechung und Zykluszeit haben daher einen erheblichen Einfluss, ob eine Diagnosemethode verwendet werden kann, denn die Rechenleistung einer CPU wird durch die Ausführung der Diagnose reduziert.

Der Verlust an Rechenleistung kann durch eine Erhöhung des Prozessortaktes bis zu einer bestimmten Grenze ausgeglichen werden.

Die typische Laufzeit des SBST bewegt sich in der praktischen Anwendung, bei maximalem Prozessortakt zwischen 400µs und 1ms.

Beispiel: Würde bei dem oben erwähnten Bauteil der Fehlerfall Kurzschluss mittels SBST ermittelt und hat der SBST eine Laufzeit von 400µs, so bleiben für die Applikation bei einer Zykluszeit von 800µs nur 400µs übrig. Die Rechenleistung reduziert sich um 50%.

In der praktischen Anwendung des SBST hat sich daher gezeigt, dass für alle Applikationen mit langer Diagnosezykluszeit die Laufzeit des SBST keine Rolle spielt. In diesen Applikationen wird der Microcontroller TC1387 gerne genommen.

Wie aus dem Bild und dem Beispiel hervorgehen, kommt der Einsatz von SBST während des laufenden Betriebs immer dann an seine Grenzen, wenn die geforderte Fehlerdetektionszeit und damit die Testzykluszeit so kurz werden, dass die verbleibende Rechenzeit und Rechenleistung für die eigentliche Applikation nicht mehr ausreichen.

Jene Applikationen mit sehr kurzer Diagnosezykluszeiten und hohen Anforderungen an die Rechenleistung kamen daher mit diesem Ansatz des TC1387 weniger zurecht.

In diesen Applikationen hat sich gezeigt, dass ein Doppelprozessor im Lockstep-Betrieb entscheidende Vorteile hat.

#### **VIII.5.4.8. Doppelprozessor im Lockstep-Betrieb**

In der Vorhabensbeschreibung zu DIANA vom 17. Juli 2009 wurde folgende Aussage gemacht:

*„Traditionelle Ansätze zur Detektierung von permanenten Fehlern im Microcontroller Core bauen auf eine redundante Ausführung von Instruktionen auf. Dabei wird meist dieselbe Instruktion auf zwei identischen Cores ausgeführt und das Ergebnis von einer Kontrolllogik verglichen. Weichen die Ergebnisse beider Cores voneinander ab, geht der Microcontroller in einen sicheren Zustand über. Dieses Verfahren hat unter anderem die folgenden Nachteile: Die zwei Cores benötigen nicht nur zusätzliche Chipfläche, sondern erhöhen auch den Stromverbrauch, was den Einsatz nur eingeschränkt möglich macht. Fehler die gleichzeitig auf beide Cores wirken,*

sogenannte „Common-Cause-Failures“ (z.B. Schwankungen in der Versorgungsspannung) können nicht erkannt werden. Ein alternativer Ansatz um permanente Fehler im Core zu erkennen ist eine funktioneller Software basierter Selbsttest (FSBST). Hierbei werden die Instruktionen, die von der sicherheitskritischen Anwendung verwendet werden, identifiziert und innerhalb eines Templates, das z.B. repräsentative Daten enthält, auf einem Instruction Set Simulator (ISS) ausgeführt und eine Referenzsignatur auf das Ergebnis gebildet. Zur Laufzeit des Anwendung im Embedded System wird der FSBST zyklisch für alle identifizierten Templates auf dem Core ausgeführt, die Signatur gebildet und gegen die Referenzsignatur verifiziert.“

Die Ergebnisse aus Aufgabe 2-4-1 und die praktischen Erfahrungen zeigen, dass der Lockstep-Betrieb auch große Vorteile haben kann auch und gerade für den Fall, wenn FSBST und Lockstep kombiniert werden.

Aus diesem Grund wurden in Aufgabe 2-4-2 auch die Optionen von Lockstep im Zusammenspiel mit FSBST dahingehend untersucht, unter welchen Bedingungen die oben erwähnten Nachteile behoben oder relativiert werden können.

#### **VIII.5.4.9. Relativierung der Nachteile von Lockstep-Systemen**

##### **Strombedarf:**

In der Vorhabensbeschreibung wurde beschrieben, dass der Betrieb eines zweiten Cores den Strombedarf verdoppelt. Bei genauerer Untersuchung stellte sich heraus dass die zweite CPU nicht automatisch den Strombedarf verdoppelt. Es wird ein nicht zu vernachlässigender Anteil des Strombedarfs für den Zugriff auf Speicher und den Betrieb des Bussystems verwendet, und nachdem diese Zugriffe nicht doppelt ausgeführt werden, benötigt die zweite CPU nur etwa 70% des Stroms. Je nach Ausführung der LCSU wird allerdings diese Einsparung wieder aufgehoben.

Im Vergleich zum Ansatz aus Aufgabe 2-4-1 mit einfachen CPUs ohne Lockstep, fällt allerdings ein anderer Aspekt ins Gewicht. Besonders in Systemen mit sehr kurzer Fehlerdetektionszeit und hoher Testzyklusrate, kann der Anteil der Rechenleistung, die für den SBST aufgewendet wird, auch in die Nähe der applikativen Rechenleistung kommen oder sie sogar überschreiten. Die Halbierung des Energiebedarfs mit nur einer CPU wird in solchen Fällen durch die Erhöhung der Taktrate wieder aufgehoben. In dieser Gruppe von Applikationen finden sich auch jene Systeme, die heute einen Lockstep bevorzugen.

##### **Common-Cause-Failures:**

Die Gruppe von Fehlern mit gleicher Ursache stellt eine generelle Gefahr für sicherheitsrelevante Systeme dar. Der Microcontroller TC1387 hat hier einige große Vorteile, dennoch können nicht alle Common-Cause-Failures erkannt werden und das Ergreifen von Gegenmaßnahmen ist in manchen Fällen genauso schwierig wie mit einem Lockstep-System.

Software und Tools: Der größte Vorteil des TC1387 ist, dass es zwei diverse CPUs in dem Silizium gibt, Tricore und PCP, die sich gegenseitig überprüfen können. Nachdem die Mehrzahl der Fehler eines Systems immer noch im Bereich der Software liegen, bringt eine Architektur bei der zwei vollkommen unterschiedliche



Software-Tools verwendet werden, hier den großen Vorteil, dass Fehler die durch Tools wie Compiler verursacht werden, in dieser Architektur mit hoher Wahrscheinlichkeit erkannt werden. Lockstep-Systeme müssen sich auf intensive Tests von Software und Tools verlassen. Die Detektion solcher Fehler kann allerdings nur in übergeordneten Schritten erfolgen und nicht innerhalb weniger Takte.

Versorgungs- und einstrahlungsbedingte Fehler: Bei diesen Fehlerquellen kippt eine Information innerhalb einer CPU. In einem diversen System, verursachen solche Ereignisse unterschiedliche Fehler, die im übergeordneten Rahmen erkannt werden. Bei Lockstep-CPU's müssen besondere Maßnahmen ergriffen werden, um sicherzustellen, dass nicht beide CPUs denselben Fehler erhalten. Folgende Maßnahmen wurden untersucht und werden später genauer erläutert:

- a) Lockstep-Core mit Zeitversatz
- b) Diverser Lockstep-Core
- c) Inverser Lockstep-Core mit Zeitversatz

#### **VIII.5.5. Ergebnisse aus der Aufgabe 2.4; HW und SW basierter Built-In Self-Test für Prozessorkerne**

In der Aufgabe 2.4.2 wurden verschiedene Hardware basierende Selbsttests Methoden (HW-BIST) analysiert, deren Ziel es ist **F**unktionalen **S**oftware **B**asierten **S**elbst **T**ests **FSBST** zu verbessern und zu ergänzen. Im Vordergrund stand dabei die Ausführungszeit, die **D**iagnostic **C**overage **DC** und der Nachweis der Wirksamkeit des Selbsttests.

##### **Methoden des HW-BIST**

Die Untersuchungen wurden auf zwei verschiedene Methodengruppen zur Implementierung eines HW-BIST konzentriert.

Der sogenannte Scan Test untersucht ganz im Sinne der übergeordneten Aufgabe von AP 2, wie sich Schaltungsmaßnahmen, welche für den Fertigungstest eingebaut wurden, auch zur Diagnose während des Betriebs eines Fahrzeugs anwenden lassen.

Die zweite untersuchte Methodengruppe, konzentriert sich darauf, wie ein FSBST im Zusammenspiel mit einem Doppelprozessorsystem im Lockstep-Betrieb kombiniert werden kann.

##### **VIII.5.5.1. Scan Test**

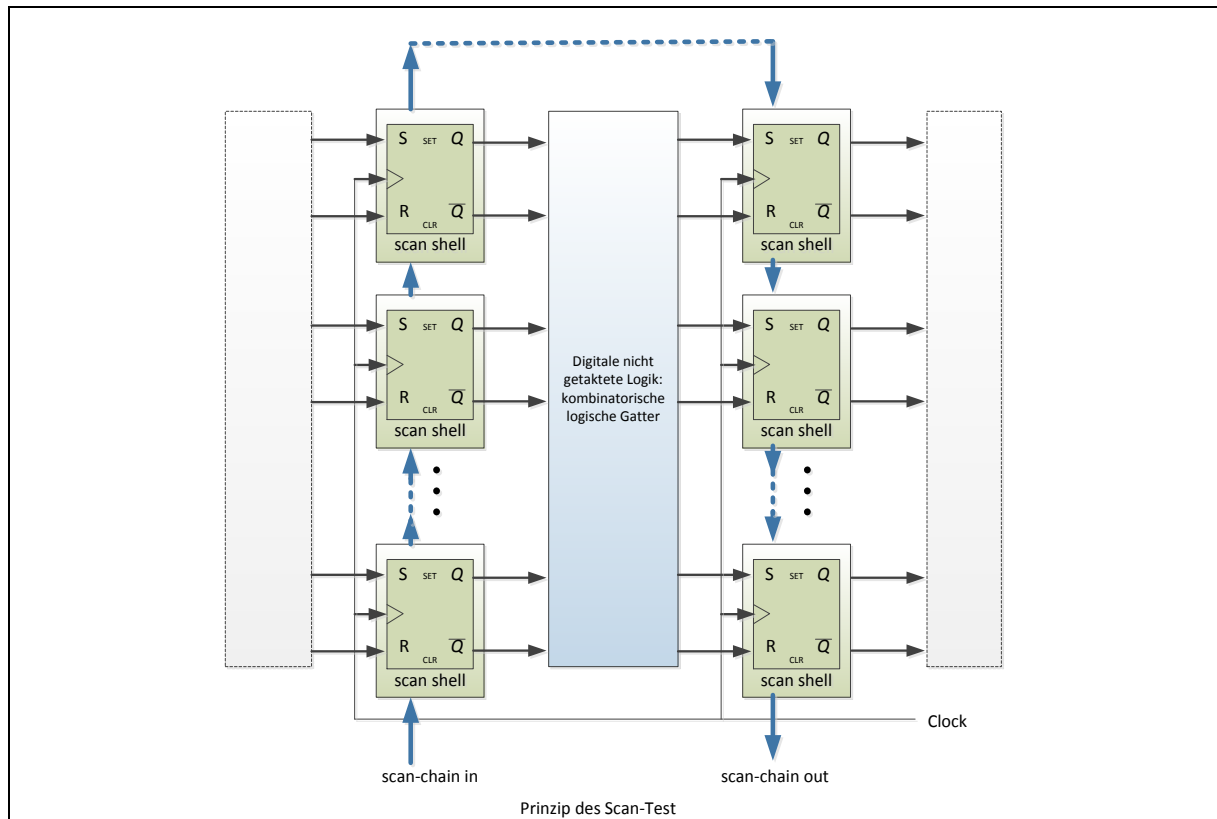
Das gängige Verfahren zum Testen digitaler Bauelemente in der Produktion ist der sogenannte Scan-Test. In diesem Kapitel wird erläutert, wie der Scan-Test arbeitet und gesteuert wird. Anschließend wird betrachtet, ob sich die vorhandenen Teststrukturen auch in einem sicherheitsrelevanten Umfeld zur Diagnose einsetzen lassen.

##### **a. Prinzip des Scan Tests**

Ein Scan-Test wird in der Regel für getaktete digitale Schaltungen eingesetzt. Prozessorkerne gehören zu den beispielhaften Anwendungen dieses Tests. Der Test

lässt sich jedoch auch auf alle digitalen Komponenten eines kompletten Microcontrollers anwenden, für den Fall dass das gesamte System den Regeln eines synchronen getakteten Designs entspricht. Dies ist für alle Microcontroller der Tricore Familie der Fall.

In so einem synchronen getakteten System beginnen und enden alle logischen Operationen in einem Flipflop oder Register. Eingangssignale werden von Flipflops bereitgestellt und die Ergebnisse (Ausgänge) der logischen Verknüpfungen werden am Ende eines Taktzyklus wieder in Flipflops gespeichert.



### Scan Chain und Scan-Flipflops

Für den Scan Test werden für alle Register und Flipflops besondere Typen benutzt. Die normale Flipflop-Funktion ist in einer zusätzlichen Schaltung, Scan-Shell genannt, eingebettet, die es erlaubt auf das Flipflop in einem speziellen Testmodus zuzugreifen. Die Scan-Shell ist so gebaut, dass eine sehr große Zahl von Flipflops mit einer einzigen Schiebekette der Scan-Chain verbunden werden kann. Über diese Schiebekette lassen sich alle Flipflops vorbelegen, sodass für die zu testende Logik Eingangszustände eingestellt werden können. Die resultierenden Testergebnisse lassen sich nach Anlegen eines Taktimpulses an den Ausgangsflipflops der Logik erfassen und über dieselbe Scan-Chain wieder auslesen. Dieser Vorgang wird solange wiederholt und variiert, bis alle für den Test nötigen Eingangszustände durchgeprüft wurden.

Bei den verschiedenen Varianten des Tricore enthält allein die CPU bis zu 100 000 Flipflops. Es werden deshalb entsprechende Entwicklungs-Tools, ATPG (automated test pattern generation) genannt, verwendet. Das ATPG bildet automatisch die Scan-

Chains und übernimmt die Generierung von allen Test-Pattern. Nur so lässt sich mit wirtschaftlich vertretbarem Aufwand die für die Automobilindustrie nötige Testabdeckung von mehr als 99,9999% erreichen.

#### **b. Eignung des Scan-Tests für die Diagnose im Fahrzeug.**

Theoretisch ist es sehr gut denkbar, Scan-Tests auch während des Betriebs eines Fahrzeugs im Prozessor durchzuführen. Hierzu sind allerdings mehrere Voraussetzungen zu beachten.

- Die Laufzeit des Tests muss den Anforderungen der Applikation genügen
- Der Test muss sich neutral verhalten. Der Prozessor muss nach dem Test wieder in denselben Zustand gebracht werden, den er vor dem Test hatte und die Testausführung darf keine Veränderung an den Ausgängen des Microcontrollers hervorrufen.
- Die Ablaufsteuerung für den Scan-Test muss im Baustein integriert sein. Die nötigen Test-Pattern nebst Testergebnissen müssen auf dem Baustein abgelegt sein.

#### **Begrenzung der Testlaufzeit**

Bewegt sich die Testlaufzeit eines Scan-Tests in der Fertigung im Bereich von Sekunden, so muss sie für die Diagnose während des Betriebs in den Bereich von 10-100 $\mu$ s gebracht werden. Das geht nur, indem man die Länge der Scan-Chains radikal reduziert und in selben Masse die Zahl der Scan-Chains erhöht. Die nötigen Test-Pattern müssen eine sehr hohe Zahl paralleler Scan-Chains unterstützen können.

Positiv würde sich eine starke Reduktion des Testumfangs wirken. Nachdem nicht alle Teile eines Prozessors sicherheitsrelevant sind und auch die geforderte Testabdeckung etwas geringer ist, kann die Zahl der im Test beteiligten Flipflops und somit die Zahl der Scan-Chains reduziert werden. Wegen der sehr hohen Zahl an Flipflops einer CPU, wird

#### **Neutrales Testverhalten**

Für einen Einsatz im laufenden Betrieb müssen zu Beginn eines Scan-Tests alle Flipflop-Zustände in besonderen Zellen abgespeichert werden, sodass nach dem Test der Ausgangszustand wiederhergestellt werden kann. Ein Reset der CPU wie im Fertigungstest ist aus Zeitgründen nicht erlaubt.

Zusätzlich müssen getestete Regionen von nicht getesteten isoliert werden. Es liegt in der Natur des Scan-Tests, dass während des Tests alle Ausgänge beliebige Wertekombinationen einnehmen können. Das hat während des Fertigungstests keine Auswirkung, muss aber während des laufenden Betriebs unbedingt verhindert werden.

#### **VIII.5.5.2. Modifikationen des Scan-Test für die Diagnose zu Run-Time**

Um einen Scan-Test im laufenden Betrieb eines Fahrzeugs ausführen zu können, ist ein großer Eingriff in den Fertigungstest nötig.

Dazu müssen erhebliche Aufwände geleistet werden

- Die automatisch generierten Scan-Chains müssen nach Sicherheitsaspekten geordnet und zusammengestellt werden.
- Die Scan-Chains müssen schaltbar gemacht werden, damit während der Fertigungstests sehr lange Schiebeketten, für den Run-time-test in tausende kurze Ketten aufgebrochen werden können
- Die Ablaufsteuerung des Scan-Tests muss integriert, ein extra Speicher für den Test bereitgestellt werden und beides muss unabhängig von der CPU testbar sein.

Ein Aufteilen der Scan-Chains nach Sicherheitsaspekten ist mit heutigen ATPG Tools nicht unterstützt. ATPG betrachtet nicht nur den Prozessor, sondern die gesamte Logik eines Microcontrollers.

Nötig wäre demnach, die Entwicklung geeignete Tools innerhalb von ATPG, die eine gezielte Generierung der vielen hundert kurzen Scan-Chains und den Einbau von Schaltern für den Fertigungstest übernehmen. Dazu müssen die Angaben zu den in Run-Time zu testenden Ketten im Design hinterlegt werden.

Als ausschließlichen Test eines kompletten Prozessorkerns zu Run-Time wird der Scan-Test aus obigen Gründen derzeit abgelehnt. Es könnten zwar die Entwicklungsaufwände mit entsprechenden Tools abgefangen werden, doch blieben die hohen Aufwände an zusätzlicher Hardware.

#### **Lokaler Scan-Test:**

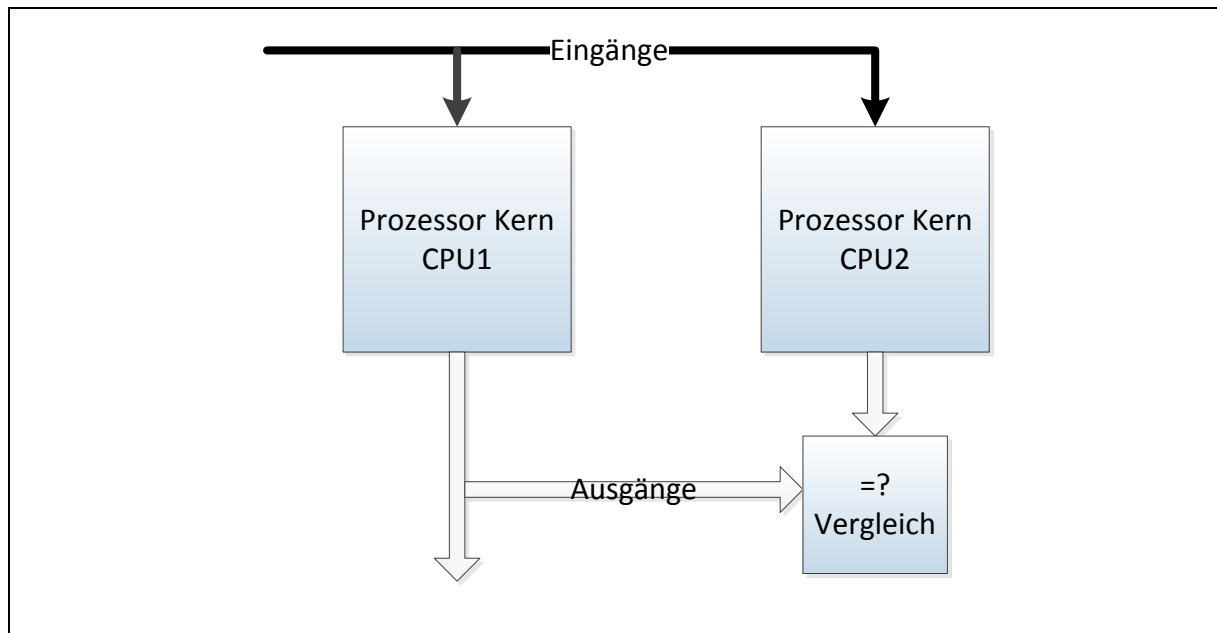
Zu empfehlen ist, den Scan-Test nur in wenigen besonders geeigneten Schaltungsregionen einer CPU als ergänzende Maßnahme zu einem FSBST anzuwenden. Besonders geeignet sind hierfür Funktionsregionen, die sich leicht isolieren lassen und einen erkennbaren Aufwand im FSBST darstellen. Dies kann genutzt werden um die DC des FSBST zu erhöhen oder auch Testlaufzeit zu reduzieren.

Wegen der noch fehlender Toolunterstützung im ATPG und damit sehr hohen Aufwänden, wurde dieser Weg bisher nicht in einer RTL-Netzliste umgesetzt. Testergebnisse können folglich nicht präsentiert werden.

### ***VIII.5.5.3. Doppelprozessorsystem im Lockstep Betrieb***

#### **a. Prinzip des Lockstep**

Doppelprozessorsysteme im Lockstep-Betrieb gehören zu den klassischen Methoden, um Fehler während des Betriebs eines Prozessors zu erkennen.



Prinzipialschaltbild Doppelprozessorsystem

Bei dem Lockstep-Betrieb werden alle Befehle und alle Eingangsdaten an zwei identische aber getrennte CPUs gegeben. Beide CPUs werden synchron betrieben. Die Operationsergebnisse werden anschließend in der sogenannten Lockstep-Control-Unit (LSCU) auf Gleichheit überprüft und im Fehlerfall ein Alarm ausgelöst.

In Prozessoren mit einer Pipeline können auch Zwischenergebnisse verglichen werden. Der Vergleich findet dann auf mehreren Ebenen statt. Am Prinzip des Locksteps stellen diese Erweiterungen keine Änderung dar und werden in der Folge auch nicht gesondert betrachtet.

#### b. Kombination von Lockstep und FSBST

Die ISO 26262 beinhaltet auch Anforderungen an latente Fehler (LF), also Fehler die nur in Kombination mit mindestens einem weiteren Fehler, das Sicherheitsziel verletzen. Die Basisnorm IEC 61508 betrachtet demgegenüber nur Einzelfehler und benutzt als Metrik den %-Anteil der Fehler die das sicherheitsgerichtete System in den sicheren Zustand versetzen, genannt Safe Failure Fraction (SFF).

In der ISO 26262 werden neben den Einzelfehlern engl. Single Point Faults (SPF) die direkt dazu führen, dass das Sicherheitsziel verletzt wird, auch Mehrfachfehler (MPF) betrachtet. Ein latenter Fehler verletzt demnach nicht direkt das Sicherheitsziel sondern ggf. erst wenn ein weiterer Fehler auftritt. Typischerweise sind dedizierte, hardware-implementierte Sicherheitsmechanismen deshalb bzgl. latenter Fehler nach der Norm einmal im Fahrzyklus zu testen. Beispiel: Lockstep Control Unit (LSCU). Ein Ausfall der LSCU führt nicht direkt zu einer Verletzung des Sicherheitszieles. Ohne weiteren Fehler rechnet die CPU einwandfrei. Erst ein weiterer Fehler, z.B. ein transienter Fehler durch alpha Strahlung der einen CPU Registerinhalt verändert führt evtl. zu einer Fehlberechnung die auf Grund des Ausfalls der LSCU nun nicht erkannt wird.

Die Norm erlaubt es nun ausdrücklich, dass ein System nur einmal pro Fahrzyklus auf LFs getestet werden muss, falls die Fitrade aller dieser Fehler zusammen ein gegebenes Maß nicht überschreitet.

Die Diagnose von SPFs während des normalen Betriebs wird durch den Lockstep abgedeckt.

Für die Diagnose von LFs wird auch bei Lockstep-Systemen ein FSBST verwendet.

Für diese Systeme gibt es somit einen großen Vorteil. Die CPU muss nur einmal pro Fahrzyklus, zu Beginn oder am Ende, einen FSBST ausführen.

**Durch den Wegfall von Diagnosezyklen während des normalen Betriebs steht die gesamte Rechenleistung für die Applikation zur Verfügung**

Das ist bei Systemen mit kurzer Fehlerdetektionszeit von sehr großem Vorteil.

**Der FSBST aus Aufgabe 2-4-1 kann ohne Veränderungen auf Lockstep-Systeme angewendet werden.**

Wird die Testsoftware auf einem der Lockstep-System ausgeführt, so werden alle Befehle in beiden Rechenkernen ausgeführt und damit werden automatisch, beide CPUs getestet. Besteht ein FSBST für eine Singlecore-Version einer CPU so kann er auch für die Lockstepversion verwendet werden.

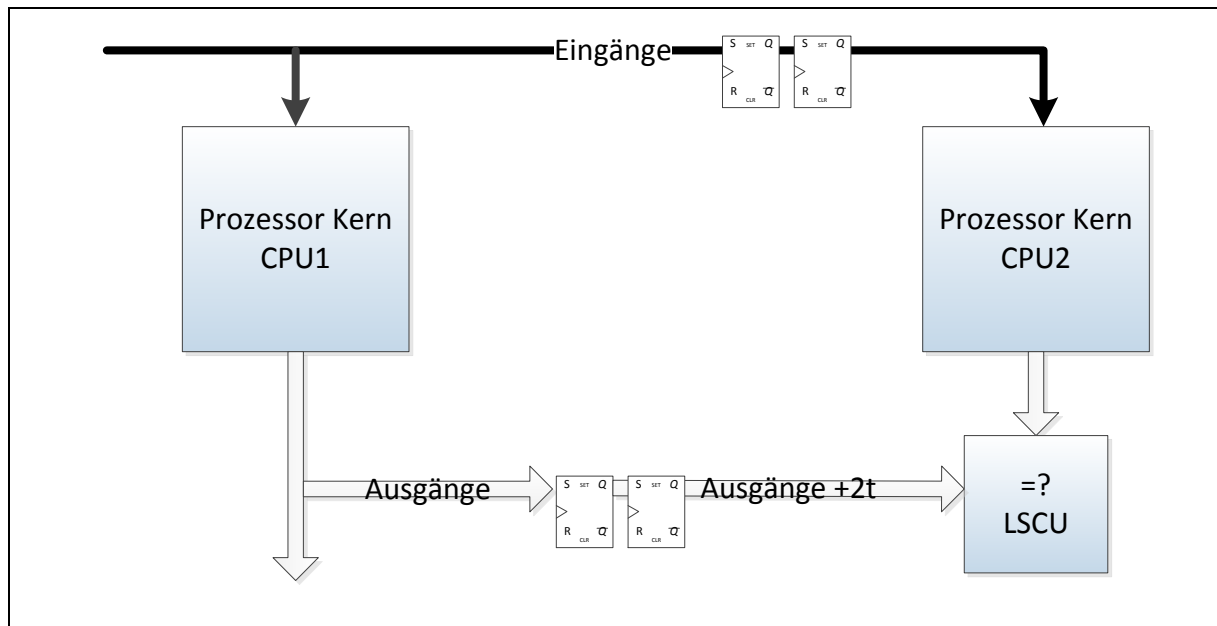
**Die die mit der Methode aus Aufgabe 2-4-1 nachgewiesene Testabdeckung ist weiterhin ohne Abstriche gültig.**

Solange sich CPU2 nicht von CPU1 unterscheidet gilt die Testabdeckung für beide. Die Testabdeckung darf allerdings nicht mit der Fitrade verwechselt werden, welche sich wegen Verdoppelung der Gatter ebenfalls verdoppelt. Der Vergleich der Testergebnisse des FSBST mit einem abgespeicherten Sollwert, kann bei einem Lockstep-System entfallen. Diese Aufgabe wird von der LSCU übernommen.

Die LSCU muss gesondert überprüft werden. Das kann durch einen separaten Test erfolgen. Für Lockstep-Systeme mit Tricore wurde erfolgreich eine LSCU entwickelt, die sich selbst überprüft. Ein eigenes Fehlverhalten der LSCU wird sofort erkannt und führt automatisch zu einem Alarm. Die Laufzeit des SBST wird verkürzt.

### **c. Lockstep mit Zeitversatz**

Viele Common-Cause-Failures haben eine sehr kurze zeitliche Wirkung. Werden die Operationen des Lockstep-Cores entsprechend zeitlich verzögert, so wirken solche Störungen auf beide Prozessoren unterschiedlich. Verursachte Fehler werden beim Vergleich der Ausgänge erkannt.



CPU2 und LSCU operieren zwei Takte später als CPU1

In der Automobilindustrie wird typischerweise ein Zeitversatz von zwei Takten gewählt.

Alle Eingangssignale werden mit Hilfe von Flipflops um zwei Takte verzögert an CPU 2 weitergegeben. Damit führt CPU2 alle Operationen um zwei Takte später aus. Die Ausgänge der CPU1 werden ebenfalls um zwei Takte verzögert und anschließend in der LSCU verglichen.

Dies hat zur Folge, dass auch Alarme um zwei Takte verzögert ausgelöst werden. Für die Ergreifung von Gegenmaßnahmen im Fall von Alarmen muss dieser Zeitversatz berücksichtigt werden.

Der Zeitversatz um zwei Takte ist zurzeit allgemein anerkannt und faktisch „state of the art“.

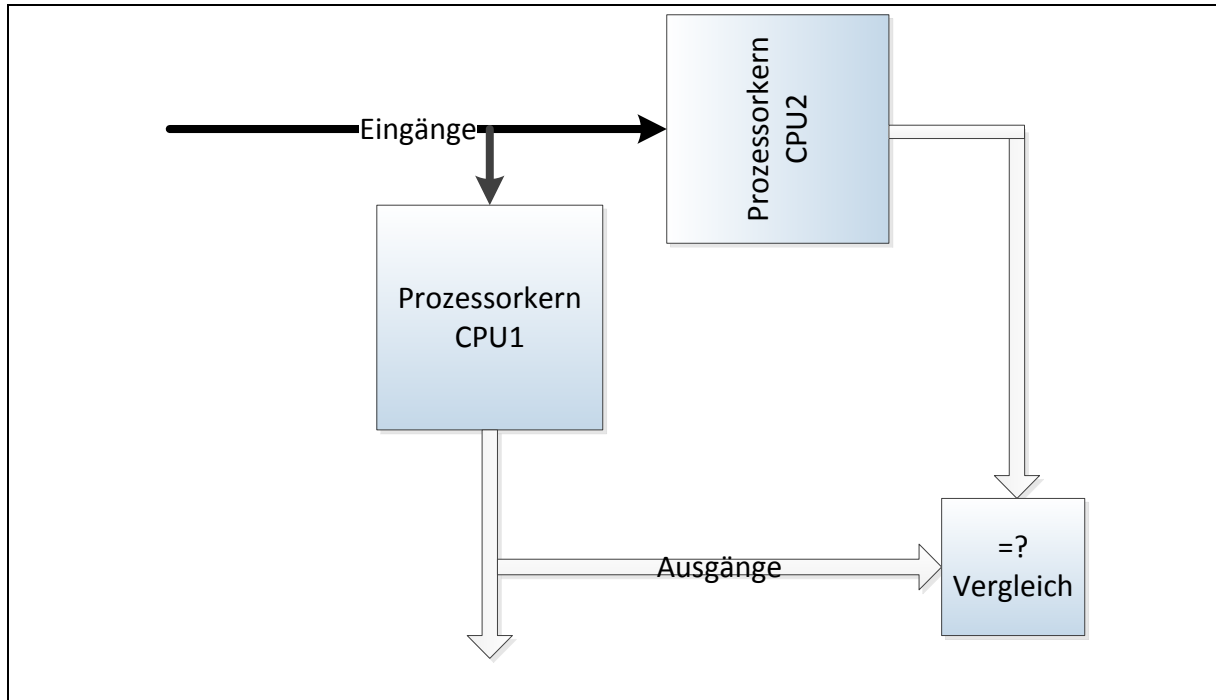
Untersucht wird, ob mit steigender Taktrate die Wirksamkeit des damit verbundenen kürzeren Zeitversatzes nachlässt und ob deshalb ein größerer Zeitversatz nötig ist. Ergebnisse liegen noch nicht vor.

#### d. Diverses Layout des Lockstep-Cores

Nachdem wie gerade beschrieben nicht alle Fehlerquellen eine zeitlich begrenzte Wirkung haben, wird typischerweise gefordert, dass der Lockstepcore als sogenanntes diverses System zum primären Core aufgebaut wird. Das Ziel dieser Maßnahme ist es, zu erreichen, dass Quellen mit lokal begrenzter Wirkung und solche mit geometrisch bedingter Wirkung unterschiedliche Fehler verursachen, die wiederum diagnostiziert werden können.

Im Microcontroller TC1387 aus Aufgabe 2-4-1 wurde diese Forderung damit erfüllt, dass zwei völlig unterschiedliche Prozessoren verwendet wurden, Tricore und PCP.

Der klassische Ansatz für ein diverses Layout mit Lockstep ist, dass im Layout des Siliziums CPU2 zu CPU1 einen genügend großen Abstand hat und um 90° gedreht ist.



Prinzip des diversierten Layouts des Lockstep Cores

Der klassische Ansatz hat jedoch auch einige Nachteile.

Moderne Entwicklungstools unterstützen ein Drehen eines Blocks von Logik nur, wenn dieser Block als sogenanntes Hardmacro vorliegt. In einem Hardmacro sind die Blockgrenzen und das Layout innerhalb des Blocks fest vorgegeben. Alle Hardmacros können daher nicht mehr an die Gegebenheiten in einem bestimmten Design angepasst und optimiert werden. Der Platzbedarf ist deshalb deutlich größer als für digitale Logik üblich.

Als weiterer Nachteil ist die geforderte Distanz zu sehen. Mit neuen Technologien und einhergehendem zunehmendem Shrink, wird es immer kostspieliger dieses Maß einzuhalten.

Infineon hat daher in der neuesten Tricore-Generation eine andere Methode für Diversität angewendet. Dieses Prinzip unterliegt noch der Geheimhaltung. Die im Rahmen von Aufgabe 2.4.1 entwickelte Methode zur Bestimmung der Fehlerabdeckung wurde auch für diese Produkte untersucht. Es hat sich gezeigt, dass sie in vollem Umfang auch für die neue Art von Diversität benutzt werden kann.

#### **VIII.5.5.4. Anwendung und Analyse eines FSBST in einem Tricore mit Lockstep**

Mit der neuesten Generation von Tricore-Produkten, wurden vier Maßnahmen umgesetzt:

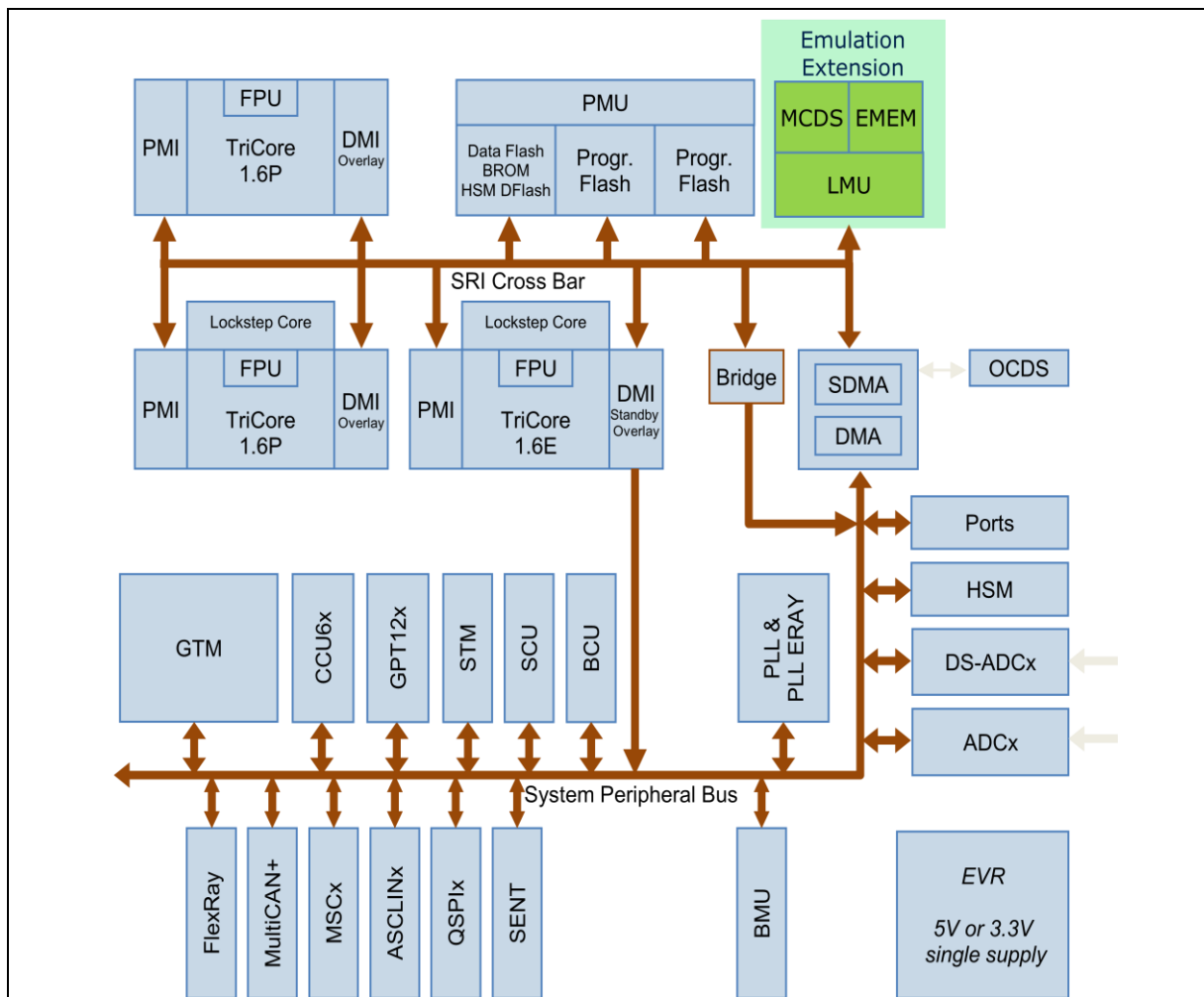
- Design nach ISO 26262



- Integration von Doppelprozessoren für den Lockstep-Betrieb
- Entwicklung eines neuen SBST und Nachweis der Testabdeckung nach Aufgabe 2.4.1 für die neue Generation von Tricore-CPU's
- Entwicklung einer entsprechenden LSCU

### Demonstratorhalbleiterbaustein

Seit Ende 2011 steht für Erprobungen der nicht kommerzielle Demonstratorhalbleiterbaustein TC275D zur Verfügung.



Architektur des Demonstratorhalbleiterbausteins TC275D

Aus Aspekten des Aufgabe 2.4. ist interessant, dass in dem Demonstratorhalbleiterbaustein zwei verschiedene Versionen des Tricore integriert sind.

1. Der TC1.6P mit 6 Pipelinestufen als superscalare Architektur
2. TC1.6E mit 4 Pipelinestufen als einfachscalare Architektur.

### Lockstep mit abschaltbarem Zeitversatz:

Beide CPU Typen liegen auch als Doppelprozessor mit Lockstep vor. Bei beiden ist der Lockstep als inverse CPU ausgeführt und bei beiden lässt sich der Zeitversatz abschalten.

Mit diesem Demonstratorhalbleiterbaustein werden in nächster Zeit die oben erwähnten Untersuchungen durchgeführt.

Diese Arbeiten übersteigen das mit DIANA umgrenzte Aufgabengebiet und den damit einhergehenden Zeitrahmen.

Ergebnisse liegen noch nicht vor.

### VIII.5.6. Zusammenfassung

Die in dieser Aufgabe entwickelte Methode zum Nachweis der Testabdeckung eines FSBST ist erfolgreich. Mit ihr ist es erstmals möglich die Testabdeckung eines FSBST reproduzierbar nachzuweisen.

Die Methode genügt sowohl den Anforderungen der Basisnorm IEC 61508 als auch der Sektornorm ISO 26262 und kann nach heutigem Stand als übergreifend angesehen werden.

Die Methode ist unabhängig von Prozessorarchitekturen und z.B. kann sowohl für klassische Einkernprozessoren als auch für Doppelprozessoren im Lockstep-Betrieb angewendet werden.

Mit den Erkenntnissen aus dieser Methode hat in den letzten Jahren ein Umdenken in der Bewertung von Softwarebasierenden Selbsttests stattgefunden. Ein Nachweis der Fehlerabdeckung nach dieser Methode ist inzwischen „state of the art“ und ist in der Branche somit als Standard anzusehen.

### VIII.5.7. Literatur / Referenz Dokumente

- [1] IEC61508-1 bis IEC61508-7, Ausgabe 2 CDV, 2008-10-31
- [2] Benware et.al. (LSI Logic), “Impact of Multiple-Detect Test Patterns on Product Quality”, ITC International Test Conference 2003, Paper 40.1
- [3] Amyeen et.al. (Intel), “Evaluation of the Quality of N-Detect Scan ATPG Patterns on a Processor”, ITC International Test Conference 2004, Paper 23.3
- [4] Guo et.al. (Intel), “Evaluation of Test Metrics: Stuck-at, Bridge Coverage Estimate and Gate Exhaustive”, 24th IEEE VLSI Test Symposium 2006
- [5] Goel et.al., “Comparing the Effectiveness of Deterministic Bridge Fault and Multiple-Detect Stuck Fault Patterns for Physical Bridge Defects: A Simulation and Silicon StudyLiterature1 for n-detect”, ITC International Test Conference 2009, Paper 1.1
- [6] IEC61508-1 bis IEC61508-7, Ausgabe 2 CDV, 2008-10-31
- [7] Benware et.al. (LSI Logic), “Impact of Multiple-Detect Test Patterns on Product Quality”, ITC International Test Conference 2003, Paper 40.1
- [8] Amyeen et.al. (Intel), “Evaluation of the Quality of N-Detect Scan ATPG Patterns on a Processor”, ITC International Test Conference 2004, Paper 23.3
- [9] Guo et.al. (Intel), “Evaluation of Test Metrics: Stuck-at, Bridge Coverage Estimate and Gate Exhaustive”, 24th IEEE VLSI Test Symposium 2006
- [10] Goel et.al., “Comparing the Effectiveness of Deterministic Bridge Fault and Multiple-Detect Stuck Fault Patterns for Physical Bridge Defects: A Simulation and Silicon StudyLiterature1 for n-detect”, ITC International Test Conference 2009, Paper 1.1

## VIII.6. Ergebnisse zur Aufgabe 2.5: Selbstreparatur-Funktionen für digitale Standard-Logik und Verbindungsstrukturen

### VIII.6.1. Aufgabenstellung

Nano-Technologien für hochintegrierte Schaltungen mit Strukturgrößen unter 50 nm weisen eine zunehmende Anfälligkeit einerseits für transiente Fehler durch Partikel-Strahlung auf, andererseits führt der zunehmende Stress zu vorzeitigen Ermüdungs- und Ausfallerscheinungen, die sich als permanente Fehler manifestieren. Auch „early lifetime failures“, also permanente Ausfälle von Bauelementen nach kurzer Betriebsdauer im Zielsystem, haben sich zu einem Problem entwickelt. Bekannte Techniken der Fehlertoleranz sind vorrangig auf die Erkennung eines oder weniger transienter Fehler optimiert. Sie sind dann, wenn sie nach vorhandenen permanenten Fehlern zusätzliche transiente Fehler erkennen und kompensieren müssen, in der Regel überfordert. Darüber hinaus benötigen Verfahren, die sich auf eine Codierung zur Fehlererkennung oder eine Verdopplung bzw. Verdreifachung stützen, in erheblichem Maße zusätzliche aktive Hardware, also auch erhebliche zusätzliche Verlustleistung. Diese zusätzliche Hardware „altert“ ihrerseits in gleichem Maße, so dass die System-Lebensdauer damit nicht verlängert werden kann. Deshalb werden zusätzliche Methoden benötigt, die eine permanente Reparatur durch Rekonfiguration leisten, wobei hier auf ruhende Reserven zurückgegriffen wird. Diese sind nicht nur zunächst neutral bezüglich der Verlustleistung, sondern sie unterliegen bis zur Verwendung auch viel weniger Stress-bedingten Alterungsprozessen als aktive Redundanz. Eine besondere Herausforderung stellt dabei die Tatsache dar, dass Reparaturfunktionen langsam und aufwändig sind und nicht on-line parallel zum laufenden Betrieb durchgeführt werden können, sondern „off-line“ entweder beim Systemstart oder in Betriebspausen ausgeführt werden müssen. Dann ist jeweils ein diagnostischer Test Voraussetzung für die gezielte Reparatur-Funktion. Hier mussten geeignete Verfahren entwickelt werden. Noch wesentlich kritischer erwies sich allerdings das Problem, bei plötzlichem Auftreten eines Fehlers im laufenden Betrieb die korrekte Systemfunktion abzusichern, bis eine Reparaturfunktion off-line möglich ist. Der Zusatzaufwand an Redundanz einerseits für die schnelle Fehlererkennung und –Kompensation on-line und für Reparaturfunktionen andererseits kann aber prohibitiv hoch werden. Dies erforderte völlig neue Architekturen, welche Funktionen des on-line-Tests einerseits und der Selbstreparatur andererseits in Kombination unterstützen können. Die Basis-Architekturen für die Selbstreparatur waren aus einem DFG-geförderten Vorgängerprojekt bekannt. Die Entwicklung von Testverfahren für die off-line-Fehlerdiagnose einerseits und die Verknüpfung mit on-line-Verfahren bildeten die wesentlichen Aufgaben.

### VIII.6.2. Ergebnisse

#### VIII.6.2.1. Basis-Architektur für Reparaturfunktionen

Die Architektur rekonfigurierbarer Basis-Blöcke mit Redundanz und Schaltfunktionen wurde bereits im DFG-geförderten Vorprojekt SELNA entwickelt [1] (Abbildung VIII-42). Für einen off-line ablaufenden diagnostischen Test wird nach Anlegen eines

Testmusters nacheinander jeder der gleichartig aufgebauten Basis-Blöcke (BB) und der Ersatzblock (Backup1) mit den Test- Ein- und Ausgängen verbunden. Zeigt ein Block dabei einen Fehler, so wird dies in einem „Fehler-Speicher“, bei dem jedem Basis-Block ein Fehler-Bits zugeordnet ist, registriert.

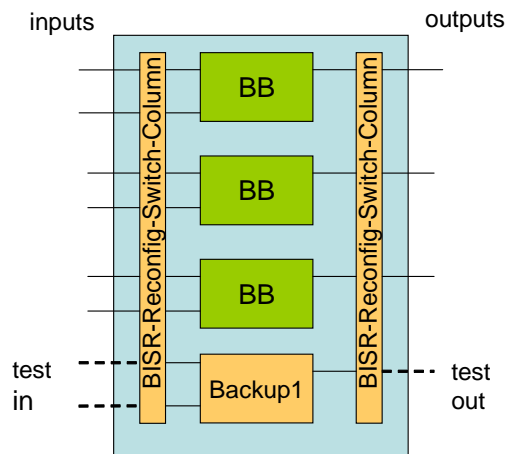


Abbildung VIII-42: Basis-Architektur für rekonfigurierbare Logik-Baugruppe (RLB)

Die Bits in diesem Fehler-Speicher steuern dann nach Ablauf des Tests eine Rekonfigurationslogik, welche als fehlerhaft markierte Basis-Blöcke abschaltet und isoliert. Dazu sind Transistor-Schalter oder Transmission Gates oder Multiplexer / Demultiplexer an den Ein- und Ausgängen notwendig. Es ist außerdem mit einer geringen Erweiterung der Kontroll-Logik möglich, zeitweise 2 Basis-Blöcke mit derselben Funktion parallel zu betreiben. So lange, wie in einem rekonfigurierbaren Logik-Block kein Basis-Block defekt ist, kann man deshalb auch die Zuordnung der Eingänge und Ausgänge zu den Basis-Blöcken sogar im laufenden Betrieb verschieben. Damit ist ohne Betriebsunterbrechung eine „Stilllegung“ hoch belasteter Basis-Blöcke für eine „Entstressung“ (de-stressing) möglich [2].

Das Anlegen der Testvektoren und die Bereitstellung von Referenzwerten für den Vergleich am Test-Ausgang sollen zunächst über einen Scan-Pfad geschehen. Ein solcher Scan-Test ist, wie im selben Projekt in Aufgabe 2.3 entwickelt, auch für einen Start-Up-Test im Zielsystem sinnvoll und möglich. Im Lauf des Projekts wurde allerdings alternativ erstmals eine erweiterte Architektur entwickelt, bei welcher die Test-Eingaben über einen on-chip-Testgenerator (z. B. ein rückgekoppeltes Schieberegister) möglich sind. Damit sind verteilte lokale Testmuster-Generatoren möglich, die insgesamt einen höheren Grad an Parallelität erreichen können als ein zentral gesteuerter Scan-Test. Die Referenz-Ausgabe wird dann für jeden Eingangsvektor durch mehrfachen Vergleich und Mehrheitsentscheidung automatisch generiert (Abbildung VIII-43). Die dazu verwendete Basis-Architektur verwendet allerdings vorteilhaft neben  $n$  Basis-Blöcken 2 Ersatzblöcke, ist also geringfügig aufwändiger als die zunächst betrachtete  $(n+1)$ -Architektur. Sie benötigt außerdem pro Ein- bzw. Ausgang 3 Transistoren als Schalter, um auch bei 2 ausgefallenen Basis-Blöcken die Funktion noch abzusichern.

Die technisch mögliche kleinste Granularität bezüglich der Komplexität der Basis-Blöcke sind logische Gatter. Dann ergibt sich allerdings ein verhältnismäßig hoher Zusatzaufwand für die Schalter und deren Steuerlogik. Noch problematischer ist allerdings, dass dann die irreguläre Verdrahtung zwischen den einzelnen rekonfigurierbaren Basis-Blöcken durch die Reparaturfunktion nicht überdeckt ist. Deshalb ist die minimale Größe für austauschbare Basis-Blöcke auf etwa 100-200 Transistoren anzusetzen. Für Makros dieser Größe ist dann aber oft schon eine reguläre Verdrahtung wie z. B. durch Bus-ähnliche Strukturen einsetzbar. Für solche regulären Bus-Strukturen wurden in einer zu Anfang der Projektlaufzeit fertiggestellten Dissertation [3] effiziente Reparaturfunktionen entwickelt. Ein Problem bei realen Schaltungen besteht darin, zunächst aus einer beliebigen Logik darin vorhandene gleichartige Teilschaltungen zu identifizieren, die in ein Schema nach Abbildung VIII-43 passen. Eine Extraktion regulärer Teilschaltungen wurde im Rahmen von 2 Diplomarbeiten entwickelt und implementiert, wovon die zweite parallel zur Projektarbeit in DIANA entstand [4, 5].

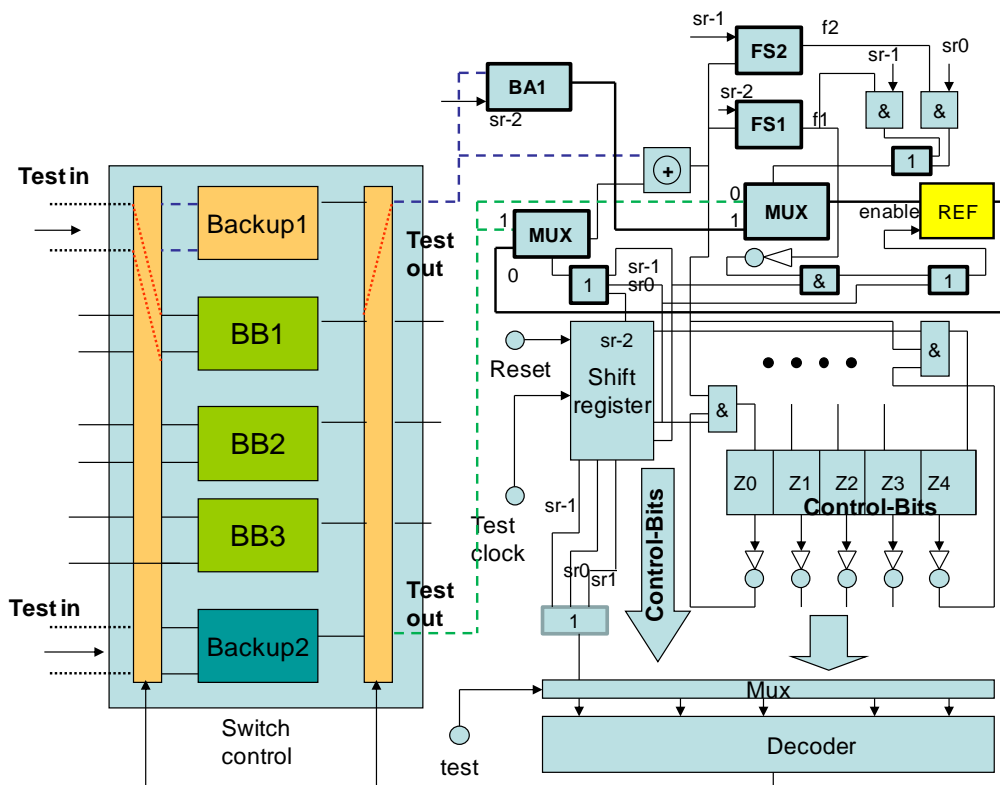


Abbildung VIII-43: Rekonfigurierbarer Logik-Block mit Erweiterung für den diagnostischen Selbsttest

Der mit verbesserten Fähigkeiten für den Selbsttest und die Selbstrekonfiguration verbundene Zusatzaufwand macht allerdings die Selbstreparatur erst ab einer Größe der Basis-Blöcke von ca. 300 Transistoren sinnvoll, da für kleinere Basis-Blöcke dann der Zusatzaufwand bestimmend für formale Abschätzung der Zuverlässigkeit wird. Systematische Berechnungen von Ausfallwahrscheinlichkeiten bei

unterschiedlichen Architekturen wurden parallel zur Projektarbeit im Rahmen eines Doktoranden-Projekts vorgenommen [6].

### VIII.6.2.2. Kombination von On-Line-Fehlertoleranz und Selbstreparatur

Funktionen der Selbstreparatur sind inhärent relativ komplex und zeitaufwändig. Sie sind deshalb weder geeignet, transiente Kurzzeit-Fehler zu kompensieren, noch können sie oder im laufenden Betrieb eines Systems plötzlich auftretende permanente Fehler erkennen und kompensieren. Da die Kombination herkömmlicher Verfahren der Fehlertoleranz mit dem Zusatzaufwand für die Selbstreparatur zu sehr hohen Kosten führen würde, bestand hier die Aufgabe, Architekturen zu entwickeln, welche sowohl Fehler laufenden Betrieb schnell korrigieren können als auch Fähigkeiten zur Reparatur durch Rekonfiguration besitzen.

#### a. Dynamische lokale Dreifach-Auslegung im Fehlerfall

Die Basis-Architektur nach Abbildung VIII-43 würde es alternativ auch zulassen, entweder 2 aus  $n$  Basis-Blöcken selektiv zu verdoppeln, um Fehler on-line zu erkennen, oder 1 aus  $n$  Basis-Blöcken funktional zu triplizieren, um eine on-line-Fehlererkennung und -Korrektur nach dem Prinzip der Dreifachauslegung (triple modular redundancy - TMR) zu erreichen.

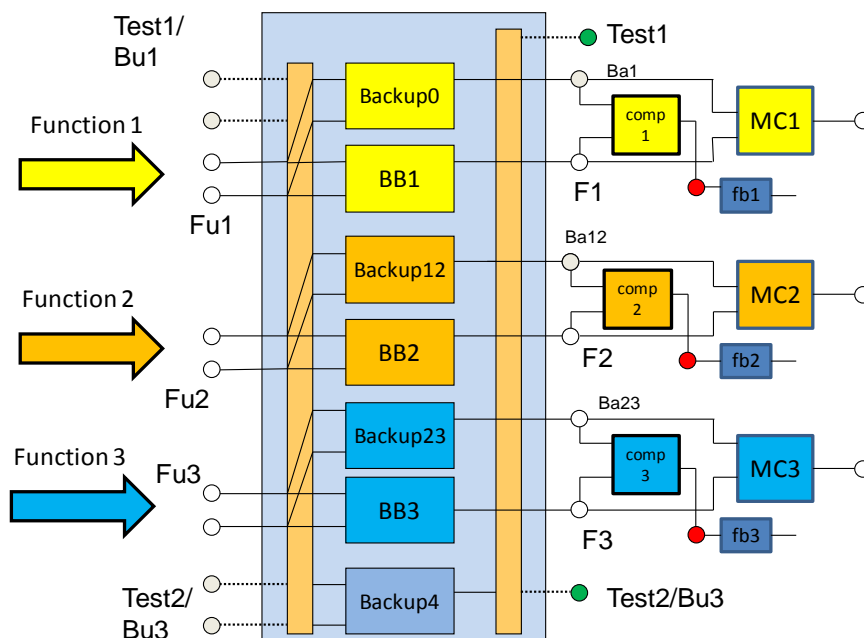


Abbildung VIII-44: Rekonfigurierbarer Block mit laufender Überwachung aller Basis-Blöcke durch Duplizierung und Vergleich

Da aber in der Regel eine Vorab-Festlegung auf selektiv zu überwachende Baugruppen nicht einfach möglich ist, besteht die Notwendigkeit, zumindest optional alle Basis-Blöcke einer laufenden Beobachtung zu unterziehen. Das ermöglicht dann sowohl die Erkennung transienter (kurzzeitiger) Fehler als auch plötzlich erscheinender permanenter Fehler im laufenden Betrieb. Dies ist mittels einer  $(2n + 1)$ -Architektur möglich (Abbildung VIII-44).

Dazu werden zunächst alle Funktionen durch einen Parallelbetrieb von 2 identischen Funktionsblöcken und nachfolgendem bitweisen Vergleich abgesichert. Die Weiterleitung fehlerhafter Ausgabe-Bits kann durch nachgeschaltete Muller-C-Elemente (MC) unterbunden werden. Ein Muller-C-Element lässt eine an den beiden Eingangsbits anliegende Belegung nur (mit Invertierung) zum Ausgang durch, wenn beide Eingänge gleich sind. Im Fehlerfall wird eine Ungleichheit angenommen. Dann bleibt der Ausgang des MV-Elements hochohmig. Ein durch ein nachgeschaltetes „keeper latch“ gespeicherter Wert bleibt erhalten. Diese Architektur erlaubt es alternativ auch, nach Auftreten eines Fehlers z. B. für die Funktion 1, eine Funktionseinheit der benachbarten Funktion „auszuleihen“ und damit temporär und lokal eine Dreifach-Auslegung zu erreichen.

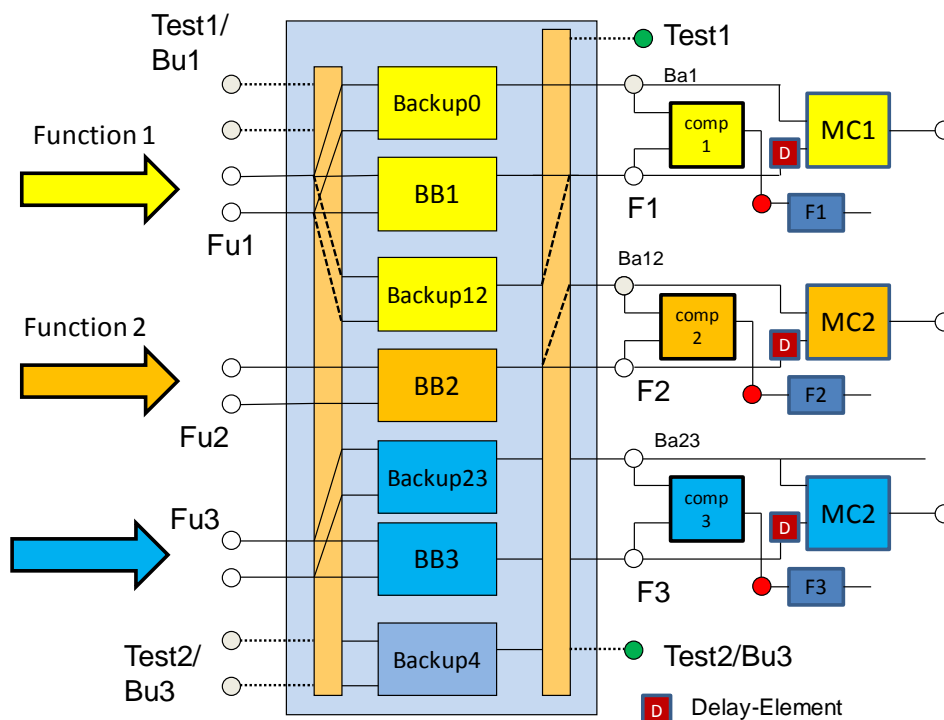


Abbildung VIII-45: Rekonfigurierbarer Logik-Block mit selektiver TMR

Wenn also nach einem ersten Vergleich z. B. das Fehler-Bits „fb1“ durch den Komparator gesetzt wird, dann erfolgt nachfolgend und lokal eine spezielle Routine zur Fehlererkennung und Fehlerkorrektur. Dazu werden die Ergebnisse der 3 Basis-Blöcke (z. B. Backup0, BB1, Backup12) in 2 weiteren Takten jeweils paarweise verglichen (Abbildung VIII-45). Die Voter-Funktion am Ausgang wird dadurch erreicht, dass das Muller-C-Element (MC1) nur bei gleichen Werten an beiden Eingängen die korrekten Ergebnisse zum Ausgang weitergibt. Das heißt, nach den 2



zusätzlichen Takten, in denen einer der beiden zusätzlichen Vergleiche den korrekten Wert geliefert hat, steht dieser am Ausgang von MC1 und wird weitergegeben bzw. in einem Flip-Flop gespeichert.

In dem Zeit-Intervall, in dem eine Funktion, hier Function1, für die Fehlerkompensation einen zusätzlichen Basis-Block ausleiht, wird eine Nachbarfunktion, hier Funktion 2, ohne Absicherung betrieben. Es ist aber ohne wesentlichen Zusatzaufwand möglich, hier durch Einsatz des ohnehin vorhandenen Muller-C-Elements eine Absicherung gegen kurze transiente Fehler zu erreichen. Dazu wird die verbliebene Baugruppe BB2 am Ausgang einmal direkt und einmal über ein Verzögerungsglied mit dem Muller-C-Element verbunden.

Eine lokale TMR-Funktion „auf Bedarf“ ist darstellbar, so lange nicht bereits Baugruppen permanent defekt sind und damit nicht mehr zur Verfügung stehen. Diese Bedingung führt ihrerseits zu einer recht komplexen Kontroll-Logik, die ihrerseits die Transistor-Schalter kontrolliert. Das heißt, in Abhängigkeit von vorhandenen permanenten Fehlern in den für eine Funktion n erreichbaren Basis-Blöcken, auch und gerade den mit einer Nachbarfunktion „geteilten“ Blöcken, wird die Fähigkeit zur Reparatur on-line reduziert.

Die prinzipielle Funktion der Kontroll-Logik zeigt Abbildung VIII-46.

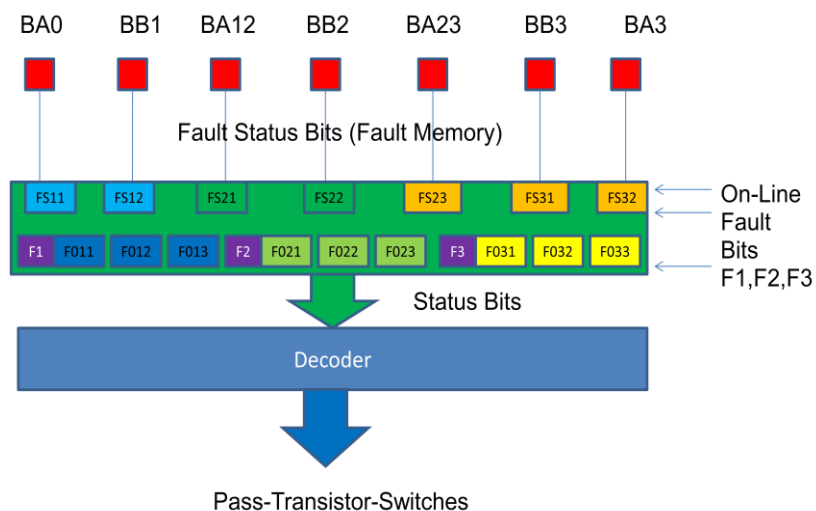


Abbildung VIII-46: Prinzip der Kontroll-Logik

Zunächst sind die den einzelnen Basis-Blöcken zugeordneten „Fehlerbits“ im Start-Up-Test des Systems bestimmt (BA0, BB1,.. BA3). Das heißt, jeder als fehlerhaft erkannte und mit „1“ im Fehler-Bit markierte Funktionsblock steht für Aufgaben der On-Line-Fehlererkennung allenfalls eingeschränkt zur Verfügung. Darüber hinaus sind Blöcke, die wahlweise zur On-Line-Fehlererkennung benötigt werden, auch dann nicht mehr verfügbar, wenn sie bei einer der Funktionen, für die sie verwendet werden können, zur Kompensation eines permanenten Fehlers benötigt werden. Im Zweifelsfall hätte also die Kompensation permanenter Fehler „Vorrang“ vor der Korrektur zusätzlicher (meistens transienter) Fehler. Aus dieser Wechselwirkung ergibt sich die Notwendigkeit, für jede der Eingangsfunktionen ein Fehlermanagement mit Zustandsübergängen zu implementieren (FFO11, FFO 12... bis FFO33), die jeweils ihrerseits von den Fehlerbits abhängig sind. Eine Übersicht

zeigt ein ziemlich komplexes System von Zuständen und Zustandsübergängen in Abhängigkeit von bereits vorhandenen „permanenten“ Fehlern und zusätzlichen möglichen „transienten“ Fehlern, bei dem nur bestimmte Zustandsübergänge zulässig sind, damit z. B. ein plötzlich auftretender Fehler in einer Funktion die parallel ablaufende Nachbarfunktion nicht stören kann.

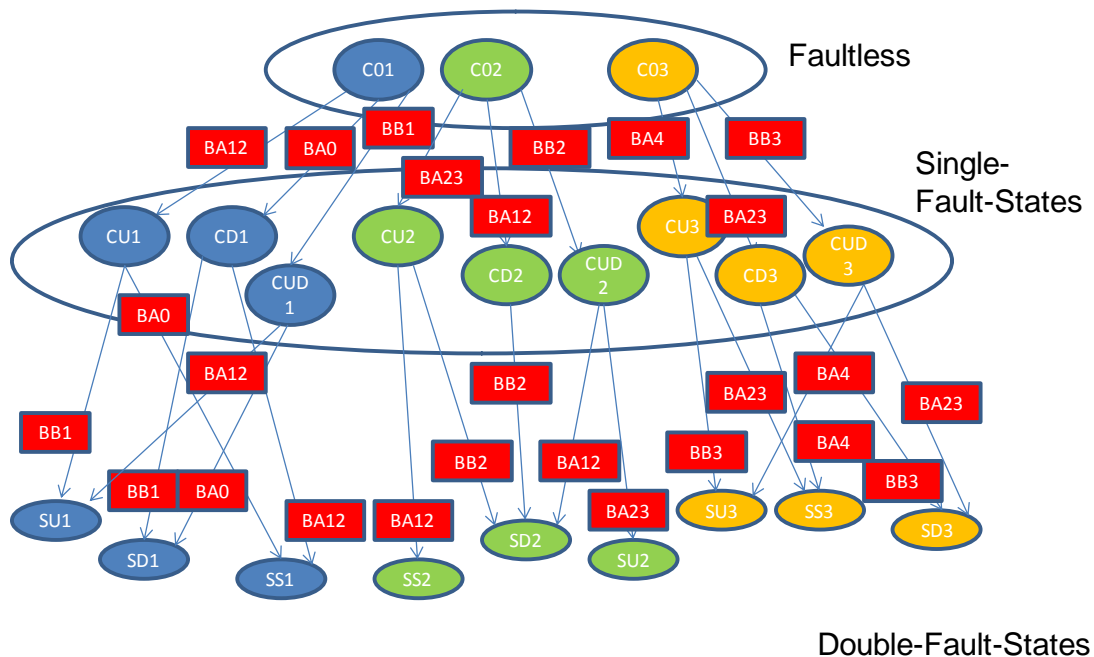


Abbildung VIII-47: Kontroll-Zustände

Wesentlicher Nachteil der  $(2n+1)$ -Architektur nach Abbildung VIII-44 ist das plötzliche Auftreten von Verzögerungen von 2-3 Takten im Fehlerfall. Dagegen treten im Normalbetrieb bis auf die Extra-Zeit für die Vergleichsfunktion keine signifikanten Verzögerungen auf. Gleichzeitig ist der für die Kontroll-Logik benötigte Aufwand relativ hoch. Deshalb wurde ein alternatives Verfahren entwickelt, das einerseits ein „konstantes“ Zeitverhalten bietet und das andererseits mit einer einfacheren Kontroll-Logik auskommt.

#### b. Dreifach-Auslegung im „Time-Sharing“-Betrieb

Eine alternative Architektur, die mit einem laufenden Dreifach-Vergleich durch Vertauschung der Basis-Blöcke für die einzelnen Funktionen arbeitet, zeigt Abbildung VIII-48. Hier wird jede logische Operation nacheinander auf 3 Funktionsblöcken gleichen Typs ausgeführt, wobei die Ergebnisse in Flip-Flops am Ausgang gespeichert werden. Dazu wird eine abgeleitete Clock mit 3 Phasen benötigt, welche die Ergebnisse aus den Funktionsblöcken in drei Takten jeweils unterschiedlichen Flip-Flops am Ausgang zwecks Zwischenspeicherung zuleitet. Dieselbe Taktsteuerung kann auch Schalter am Eingang (3-Way-Switches) oder

Demultiplexer ansteuern, welche jedes anliegende Signal an den Eingängen nacheinander den 3 Funktionseinheiten zuordnen. Abschließend erfolgt ein Dreifach-Vergleich mit Weiterleitung des korrekten Ergebnisses zum Ausgang.

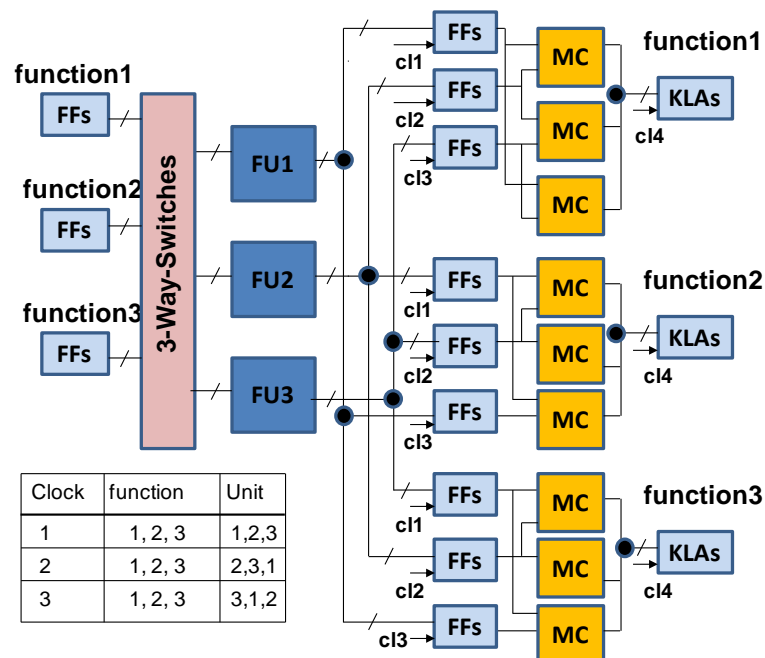


Abbildung VIII-48: Basis-Block mit Time-Shared TMR

Diese Architektur erzeugt eine konstante Verzögerung um 2-3 zusätzliche Takte und weist damit ein „reguläres“ Zeitverhalten auf. Gleichzeitig werden transiente Fehler in den Flip-Flops selbst mit erfasst und korrigiert. Die Basis-Blöcke dürfen wie in der  $(2n+1)$ -Architektur nur kombinatorischer Natur sein. Die Architektur enthält einen neuartigen Voter-Baustein, der auf der Basis von Muller-C-Elementen arbeitet.

Nachdem in 3 Takten die 3 Flip-Flops für jeden Ausgang die mittels der Baugruppen FU1, FU2 und FU3 erhaltenen Ergebnisse beinhalten, erfolgt eine quasi-automatische Mehrheitsentscheidung. Bei 2 korrekten und einem falschen Ergebnis liefert von den 3 Muller-C-Elementen nur das ein korrektes Ergebnis, dessen Eingang mit den korrekt und gleich arbeitenden FUs verbunden ist. Die beiden anderen Muller-C-Elemente sperren bei ungleichen Eingangswerten den Durchgang und schalten auf „high impedance“ am Ausgang. Dem Ausgang nachbeschaltet wird ein sogenanntes „Keeper Latch“. Transiente und permanente Fehler in den Flip-Flops selbst werden hier mit überdeckt. Es ist alternativ auch möglich, vor die MC-Elemente nur Latches (Master) zu legen und das Ergebnis der Muller-C-Elemente in einem gemeinsamen (Slave-) Latch zu speichern, welches dann aber nicht gegen Fehler abgesichert ist.

Die Architektur nach Abbildung VIII-48 ist zunächst mittels der Dreifachauslegung (triple modular redundancy, TMR) in der Lage, sowohl transiente Kurzzeit-Fehler als auch permanente Fehler zu erkennen und zu korrigieren. Selbst bei Mehrfach-Fehlern ist noch eine Korrektur wahrscheinlich, so lange mehrere Fehler in mehreren Bausteinen (FUs) unterschiedliche Teilfunktionen betreffen. Der in dieser Architektur

auftretende Zeitbedarf für die Verarbeitung, der im Vergleich zum Normalbetrieb 3-fach höher ist, legt aber auch eine Erhöhung des lokalen Taktes um den Faktor 3 aus Ausgleich nahe. Dieser führt wiederum zu einer erhöhten Stress-Belastung, welche eine zusätzliche (off-line-)Reparaturfunktion nahelegt. Die erweiterte Architektur ist in Abbildung VIII-49 dargestellt. Sie benötigt nicht nur einen zusätzlichen Funktionsblock, sondern auch erweiterte Schalter an den Eingängen (4-Wege) und den Ausgängen (2-Wege).

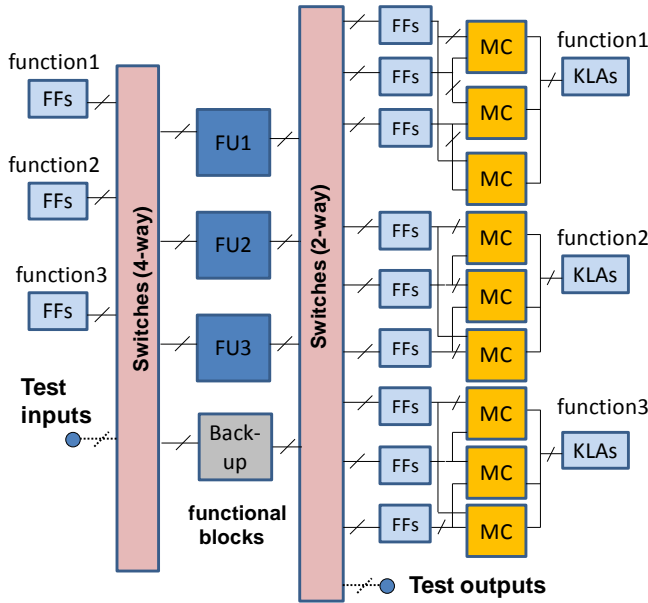


Abbildung VIII-49: Time-Shared TMR-Architektur mit Selbstreparatur-Funktion

Hier wird allerdings zusätzlich auch die Funktion der Off-line-Selbstreparatur benötigt, also ein Controller-Baustein mit Fehlerspeicher nach dem Vorbild von Abbildung VIII-43.

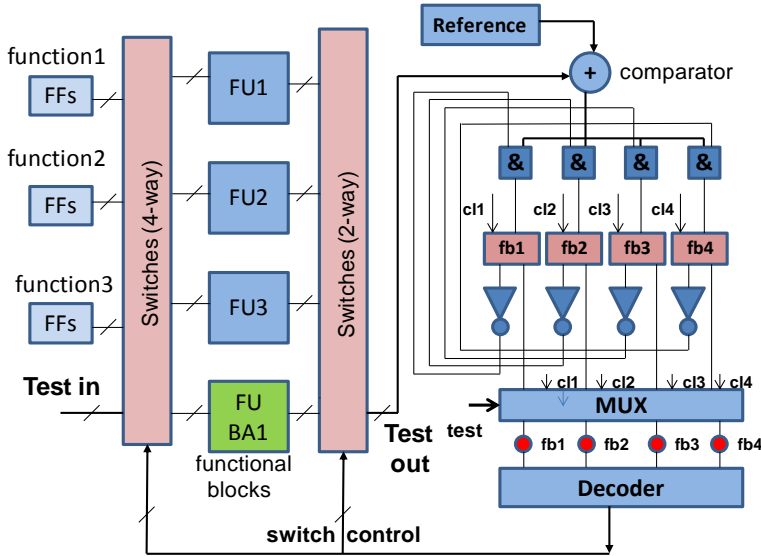


Abbildung VIII-50: für die Selbstreparatur bei Time-Shared TMR und  $n = 3$

Er kann allerdings auch hier die für den aktiven Betrieb benötigten Takte verwenden, solange eine (3+1)-Architektur verwendet wird (Abbildung VIII-50). Mit einem etwas größeren Aufwand für den Selbsttest und die Rekonfiguration ist das Verfahren auch auf beliebige (n+1)-Architekturen skalierbar.

### **VIII.6.2.3. Software und Schaltungsentwürfe**

#### a) Makro-Extraktor zur Ermittlung regulärer Teilstrukturen in Gatter-Netzlisten

Dieses Werkzeug führt auf Logik-Netzlisten eine Analyse bezüglich regulärer Teilstrukturen durch, die als Basis-Elemente in rekonfigurierbaren Logik-Blöcken verwendet werden können. Es ist möglich, diese Suchfunktion durch Vorgabe von Suchmustern, die verfügbaren Makro-Elementen von Zellen-Katalogen entsprechen, zu steuern.

#### b) Konfigurierbares Synthese-Werkzeug für die Controller-Strukturen einfacher Selbstreparatur-Architekturen (n+1- und n+2- Architekturen)

Hier sind erste Werkzeuge entstanden, die es ermöglichen, für einfache rekonfigurierbare Logik-Blöcke die Controller zu synthetisieren, welche eine Selbstreparatur-Funktion steuern können.

#### c) Beispiel-Prozessor mit Selbstreparatur durch Rekonfiguration

Parallel zu DIANA wurden im Rahmen der Graduiertenschule der BTU Cottbus und von Mitarbeitern Verfahren entwickelt, welche Selbstreparatur-Funktionen für statisch geschedulte (RISC und VLIW)-Prozessoren zum Inhalt haben. Entwickelt wurde daraus ein Beispiel-Prozessor (32 Bit, RISC / VLIW-Architektur), der über die Fähigkeit verfügt, ausgefallene arithmetisch / logische Baugruppen, Multiplexer und Register zu „reparieren“. Ausgangspunkt ist eine Architektur, bei welcher die Ablaufplanung vorab durch den Compiler festgelegt wird. Tritt ein permanenter Fehler auf, so ist auf der Basis der Verfügbarkeit multipler Baugruppen dieser Art die vorhandene Ablauf-Organisation (Scheduling und Allocation) so zu modifizieren, dass defekte Baugruppen nicht mehr benutzt sind und durch die Verlagerung der Funktion auf andere Baugruppen ersetzt werden. Über eine Hardware-basierte Selbstreparatur-Funktion verfügt nur die Kontroll-Logik des Prozessors, um eine Software-basierte Selbstrekonfiguration zu ermöglichen. Die nach Diagnose eines Fehlers benötigte zeitliche und räumliche Umverteilung von Instruktionen auf Baugruppen wird mittels einer speziellen Software-Prozedur vom Prozessor selbst ausgeführt, optional von einem externen Hilfsprozessor. Durch die Verlagerung der Reparatur-Funktion in Software wird der Hardware-Overhead signifikant reduziert (ca. unter 10 %).

Neben dem Prozessor-Entwurf (in VHDL) stehen der benötigte Compiler (aus C) und die Software für die Reparaturfunktion zur Verfügung.

### **VIII.6.3. Zusammenfassung und Vergleich**

Entwickelt wurden mehrere Basis-Architekturen für selbstreparierende Logik-Baugruppen, die optional auch die Fähigkeit des on-line-Tests und der schnellen

Fehlerkorrektur haben können. In allen Fällen konnte der benötigte Zusatzaufwand abgeschätzt werden. Entwickelt und ausgeliefert wurden auch CAD-Software-Werkzeuge für:

- a) Die Extraktion regulärer Teilnetze aus irregulären Gatternetzen
- b) Die Synthese der Kontroll-Strukturen für die (n+1) und die (n+2)-Architektur

Darüber hinaus wurde als Beispiel ein Prozessor (in VLIW)-Architektur entwickelt und dokumentiert, der für die Kontroll-Logik die Fähigkeit zur eingebauten Hardware-basierten Selbstreparatur besitzt, während Einheiten des Datenpfades durch Software-Funktionen rekonfiguriert werden können. Als wichtiges Ergebnis kann aber auch gelten, dass Möglichkeiten, Aufwand und Grenzen der einzelnen Architekturen für die Selbstreparatur im Vergleich verfügbar werden (Tabelle I.1). Damit ist es für zukünftige Entwicklungen hoch-zuverlässiger Elektronik-Systeme möglich, die jeweils passende Alternative auszuwählen. Die Tabelle vergleicht die oben vorgestellten Architekturen für die Selbstreparatur in Form der (n+2)-Architektur mit 3 bzw. 5 parallelen Funktionen mit konventioneller Dreifach-Auslegung mit Votern (TMR), mit der Time-Shared TMR ohne Reparaturfunktion, mit einer Time-Shared TMR-Architektur, die für die Selbstreparatur erweitert ist, und mit der Architektur mit Duplizierung und selektiver TMR. Der Overhead bezieht sich auf die benötigten zusätzlichen Transistoren. Ebenso interessant ist aber die Abschätzung der zusätzlich benötigten Verlustleistung. Während reine Reparaturverfahren mit „schlafender“ Redundanz auskommen und damit kaum zusätzliche Verlustleistung benötigen, liegt der Zusatzaufwand für die laufende Überwachung durch Verdopplung bei mehr als 100 %. Die Time-Shared TMR ist bezüglich der zusätzlichen Hardware günstig, benötigt aber bei gleichem Durchsatz eine etwa dreifach höhere Verlustleistung.

Tabelle I.1: Vergleich von Architekturen für Fehlertoleranz und Selbstreparatur

Method	fixed delay	delay-at-fault	self repair	overhead*	extra power
<b>BISR (3+2)</b>	-	<b>(no detect)</b>	<b>yes</b>	<b>130 / 156 %</b>	<b>marginal</b>
<b>BISR (5+2)</b>	-	<b>(no detect)</b>	<b>yes</b>	<b>89 / 70 %</b>	<b>marginal</b>
<b>TMR</b>	-	-	-	<b>220 / 213 %</b>	<b>&gt;200 %</b>
<b>TS-TMR</b>	<b>3-4 cycl.</b>	<b>no extra</b>	-	<b>69 / 61 %</b>	<b>&gt;200 %</b>
<b>TS-TMR-BISR</b>	<b>3-4 cycl.</b>	<b>no extra</b>	<b>yes</b>	<b>120 / 110 %</b>	<b>&gt;200 %</b>
<b>Dupl. / TMR</b>	-	<b>2-3 cycl.</b>	<b>yes</b>	<b>250 / 202 %</b>	<b>~ 100 %</b>

\* with basic block size of 352 / 699 transistors

Die verwendeten Basis-Blöcke, die jeweils überwacht und ersetzt werden, haben hier eine Größe von 350 bzw. 700 Transistoren. Wir glauben, dass es mit den hier gewonnenen Ergebnissen möglich sein sollte, Wege zu finden, um Langzeit-zuverlässige Systeme (wie z. B. Baugruppen der Automobil-Elektronik) auch mit selbst nicht hoch-zuverlässigen Bausteinen der Nano-Elektronik zu realisieren.

Von ganz entscheidender Bedeutung für den Aufwand und insbesondere die Verlustleistung sind aber in jedem Fall die Anforderungen an die zeitlichen Schranken für die Fehlerbehebung. Eine Fehlerkorrektur „im selben Takt“ ist immer mit einem Dreifach-Aufwand an Hardware und Energie verbunden, während sich der Aufwand für ein Reparatur-Zeitfenster von 3-4 Takten schon signifikant reduzieren lässt. Die Hardware-basierte Selbstreparatur auf der Basis „schlafender“ Redundanz benötigt Zeiten in der Größe von einigen 100 Millisekunden. Steht bei regulären Baugruppen wie Prozessoren gar ein Zeitfenster im Bereich von Sekunden zur Verfügung, so kann die Rekonfigurierung auch durch Software erfolgen. Bei solchen Verfahren, die parallel zu DIANA im Rahmen einer Dissertation entwickelt wurden, kann ein geeigneter Prozessor (VLIW) mit einem Overhead von weniger als 20 % auskommen. Nur die Kontroll-Logik des Prozessors benötigt dann die Hardware-basierte Selbstreparatur.

#### **VIII.6.4. Literatur**

- [1] Tobias Koal, Daniel Scheit, Mario Schölzel, Heinrich T. Vierhaus: „On the Feasibility of Built-in Self Repair for Logic Circuits“, IEEE Int. Symposium on Dependability and Fault Tolerance (DFT 2011), Vancouver, Oct. 2011
- [2] Tobias Koal, Heinrich T. Vierhaus: „Combining De-Stressing and Self Repair for Long-Term Dependable Systems“, Proc. IEEE DDECS 2010, Vienna, April 2010
- [3] Daniel Scheit: „Fault-tolerant integrated interconnections based on built-in self repair and codes“, Dissertation, BTU Cottbus, Juli 2011
- [4] Christian Gleichner: „Extraktion gleichartiger Teilschaltungen aus Logik-Netzlisten“, Diplomarbeit (Informatik), BTU Cottbus, April 2009
- [5] Stefan Kulke: „Extraktion regulärer Module aus irregulären Gatter-Netzlisten auf der Basis struktureller Vorgaben“, Diplomarbeit (Informatik), BTU Cottbus, März 2012
- [6] Tobias Koal, Daniel Scheit, Heinrich T. Vierhaus: „Reliability Estimation Process“, Proc. 12th Euromicro Conference on Digital System Design (DSD), Patras, August 2009, pp. 221-224, IEEE CS Press, 2009, ISBN 978-0-7695-3782-5, pp. 221-224
- [7] Tobias Koal, Markus Ulbricht, Piet Engelke, Heinrich T. Vierhaus: „Selbstreparatur für Logik-Baugruppen mit erweiterten Fähigkeiten für die Kompensation von Fertigungsfehlern und Frühausfällen“, Dresdner Arbeitstagung für Schaltungs- und Systementwurf (DASS 2012), Dresden, Mai 2012
- [8] Tobias Koal, Markus Ulbricht, Piet Engelke, Heinrich T. Vierhaus: „Logic Self Repair Architecture with Self Test Capabilities“, ITG/GI/GMM Arbeitstagung „Zuverlässigkeit und Entwurf“ 2012, Bremen, Sept. 2012, Herausgeber: Rolf Drechsler
- [9] Tobias Koal, Markus Ulbricht, Piet Engelke, Heinrich T. Vierhaus: „Kombinierte On-Line-Fehlerkompensation und Selbstreparatur für Logik-Baugruppen“, ITG/GI/GMM-Arbeitstagung „Test und Zuverlässigkeit von Schaltungen und Systemen 2013“, Dresden, Februar 2013

- [10] Tobias Koal, Markus Ulbricht, Heinrich. T. Vierhaus: „On the Feasibility of Combining Fast Fault Detection and Self Repair for Logic Circuits“, Proc. IEEE DDECS 2013, Karlsbad, April 2013
- [11] Tobias Koal, Markus Ulbricht, Piet Engelke, H. T. Vierhaus: „Combining Fault Tolerance and Self Repair in a Virtual TMR Scheme“, Proc. ITG/GI/GMM-Fachtagung “Zuverlässigkeit und Entwurf”, (ZuE 2013), Dresden, September 2013
- [12] Tobias Koal, Markus Ulbricht, H. T. Vierhaus, „Virtual TMR Schemes Combining Fault Tolerance and Self Repair“, Proc. Euromicro-Conference on Digital System Design (DSD 2013), Santander, September 2013, IEEE CS Press



## **IX. Ergebnisse zum Arbeitspaket 3: „Fehleranalyse auf Steuergeräteebene & Demonstrator“**

Vorrangiges Ziel des Projekts DIANA ist es, die Diagnosefähigkeiten von elektronischen Steuergeräten im Automobil so zu erweitern, dass eine schnellere Fehlererkennung und Fehlerbehebung beim Automobilhersteller bzw. in der Werkstatt ermöglicht wird.

Im dritten Arbeitspaket, „AP3: Fehleranalyse auf Steuergeräteebene & Demonstrator“, wurden die Anforderungen aus Steuergerätesicht an die Diagnoseinfrastruktur auf der Bauelemente- und Halbleiterebene definiert. Die in AP2 entstandenen Verfahren wurden auf ihre Eignung für die Diagnose auf Steuergeräteebene bewertet und angepasst. In AP3 entstand ein Demonstrator Continental auf Steuergeräteebene. Ziel war es zu zeigen, dass die Test- und Diagnoseverfahren einen Fehler innerhalb eines Halbleiters, eines diskreten Bauelementes oder in der angeschlossenen Peripherie, wie zum Beispiel bei Sensoren, erkennen und die Fehlerinformation effizient an das Gesamtsystem weitermelden können (siehe auch AP4, Aufgabe 4.4, Uni Stuttgart, SW-basierter Selbsttest- SBST auf CHIPit Synopsis – emuliert Conti Demonstrator und AP3, Aufgabe 3.3: Institut für Technik Intelligenter Systeme, ITIS an der Universität der Bundeswehr München, Diagnose von Komponenten und Modulen im CAN-basierten Systemverbund).

Ergänzend hat die Durchführung einer Produkt-FMEA mit gezielten Detailfragen zur Fehlerauswirkung und deren Wirkungsketten wertvolle Anregungen zu Anforderungen an neue komplexe Bauteile beigesteuert. Als Ergebnis hieraus werden zukünftig komplexe Zukaufteile, wie z.B. integrierte Schaltkreise oder Sensoren hinsichtlich der zu erwartenden internen Fehler und deren Auswirkungen noch genauer beleuchtet. Dies mündet u.a. in einer geänderten Produkthanforderung und Dokumentation für neu zu entwickelnde Bauteile und Komponenten.

Weiter wurde im AP3 mit Aufgabe 3.3 eine Diagnosemethode entwickelt, die es dem Anwender ermöglicht, den spezifikationsgemäßen Zustand von Steuergeräten und deren Komponenten ohne physikalischen Eingriff zu evaluieren. Das Konzept zur Diagnose von Komponenten und Modulen in einem CAN-basierten Systemverbund (Controller Area Network) wurde anhand des DIANA Demonstrators vorgeführt und erklärt.

### **IX.1. Ergebnisse zur Aufgabe 3.1: „Demonstrator zur Absicherung der Praxistauglichkeit der Ergebnisse des Gesamtprojekts“**

#### **IX.1.1. Anforderungen an eine ECU im DIANA Projekt (HM-3.1 Mit AP2 und AP4 abgestimmtes Konzept für einen Demonstrator auf SteuergeräteEbene)**

Zunächst wurden die Anforderungen und Randbedingungen der Diagnoseinfrastruktur betrachtet.

Die Randbedingungen für den Demonstrator sind:

- Vorhandener Sensorcluster soll benutzt werden,
- Verbauort von Demonstrator im Getriebe nur schwer zugänglich,
- Umgebung von Demonstrator im Getriebe stark mit Öl belastet,
- rein digitale Schnittstelle von Demonstrator zum Fahrzeug -> erfüllt damit die Kriterien einer Isolation / eigenständig)

Zentrales Element im Demonstrator ist der Microcontroller - er muss als gesamte Datenverarbeitungsanlage begriffen werden mit den Funktionen:

- hochauflösende ADC-Schnittstelle
- Hochgeschwindigkeitsdatenübertragungskanal
- sicherheitskritische Funktionalität
- eingebettete Test- und Diagnosefeatures
- rekonfigurierbar
- plattformunabhängig
- echtzeitfähig

Als zentrales Element muss für den Microcontroller eine Strategie gefunden werden, Diagnoseinformationen in Echtzeit zu gewinnen. Das Auftreten und die Rückverfolgung von allen Fehlern muss möglich sein (Auftrittsort, Abgrenzungsdefinition).

Weitere Anforderungen und Randbedingungen sind:

- Das Auftreten von Fehlern muss reproduzierbar sein (Erfassung von sporadischen und nicht vorhersehbaren Fehlern -> kontinuierliche Überwachung / polling).
- Diagnosefähigkeit (Welche Parameter sind zu diagnostizieren: Druck, Drehzahl, Wählhebelstellung, ...):
- Zur Verfügungsstellung verschiedenster SW-Schnittstellen, über die man Daten bekommt (offene/chiffrierte Schnittstellen),
- SW-Plattform/Hardware-Anforderungen bei SW-Debugging-Tools, SW-Tool-Flow,
- Datenübertragung Auslesemedium/Schnittstellendefinition (Bus-System)
- Anforderungen an die Performance,
- Datenhaltung/Übertragungsmedium (Speicheranforderungen, Datengrößen, wieviel Diagnosedaten können generiert werden?),
- Testbarkeit bzgl. Zugang –Einbau, Umgebung, Bandbreite, Versorgung,
- Integration von Fremd-/Eigen-IP (erfüllt die OEM-Anforderungen, gewährleistet Geheimhaltung, Produkthaftung wird unterstützt, Standardisierung ist ein MUSS)
- OEM-seitige Anforderungen an die Störfestigkeit/EMV

### IX.1.2. Die Aufgaben einer Transmission Control Unit

Die Continental AG hat mit ihrer Business Unit *Transmission* am DIANA Projekt teilgenommen. Dieser Geschäftsbereich entwickelt am Standort Nürnberg Electronic Control Units (ECU) für Getriebe in Straßenfahrzeugen, wie Doppelkupplungsgetrieben, Automatikgetrieben, Continuous Variable Transmissions usw.. Es war deshalb naheliegend für das Projekt DIANA, den Demonstrator auf einer Transmission Control Unit (TCU) basieren zu lassen, zumal diese eine typische ECU mit folgenden Eigenschaften darstellt:

- $\mu$ C als zentrales Steuerungselement mit möglichst allen Peripheriefunktionen integriert (system-on-a-chip), wie:
  - On-board memory für Cache, Boot ROM, Flash, Daten/Program RAM
  - Separater Peripheral Control Processor mit eigenem Speicher
  - Interrupt Controller
  - Serielle Schnittstellen, wie CAN, FlexRay, LIN...
  - Serial Peripheral Interface (SPI) für die Kommunikation mit Monitoring Unit
  - JTAG Interface für die  $\mu$ C Diagnose
  - A/D Wandler zur Überwachung von TCU Versorgungsspannungen
  - A/D Wandler zur Überwachung von Sensoren (z.B. Drehzahl, Kupplungsdruck, Momentendruck, Gangsteller-Position)
  - A/D Wandler zur Überwachung von Aktuatoren
  - Ausgänge zur Steuerung von Aktuatoren über PWM-, Analog-, Digital-Signale
- Spezieller IC (*System Basis Chip for Transmission Control Units*) mit folgenden Aufgaben:
  - Generierung aller Versorgungsspannungen für die TCU aus der Batteriespannung über Klemme 30/31 (falls nötig, durch Steuerung externer Spannungsregler)
  - Überwachung des Microcontrollers mit Hilfe einer Monitoring Unit, die zyklisch  $\mu$ C Testfunktionen ausführen lässt mit Hilfe eines Question-Answer Protokolls über SPI.
- EMV/Anti-Aliasing Filter für alle AD Wandler Eingänge
- Treiber für Aktuatoren
- Eingangsbeschaltung von CAN Interface
- Externe Beschaltung für  $\mu$ C (Clock Oszillator ...) und *System Basis Chip for Transmission Control Units*
- Sensoren für Kupplungsdruck, Momentendruck (Drehmoment am Getriebeeingang), Wählhebelstellung, Getriebeeingangsdrehzahl, ....

### IX.1.3. Auswahl des Continental Getriebesteuergeräts VL381 als Basis für den Continental Demonstrator

Bei der Auswahl standen die Diagnosefähigkeit, Zugriffsmöglichkeiten von außen, der Einsatz von Halbleiterbauelementen der Projektpartner in dem Steuergerät – insbesondere der Prozessor – sowie Schnittstellen und Bus-System im Fokus der Betrachtungen. Das Konzept des Continental Demonstrators basiert auf einer

Continental Transmission Control Unit VL381, die Zugriffsmöglichkeiten über CAN Bus und über JTAG aufweist. Die JTAG (Joint Test Action Group) Schnittstelle erlaubt den low-level Zugriff auf alle HW Ressourcen des TCU Microcontrollers. Hiermit ist eine wesentlich größere Testtiefe gegenüber der bisherigen Diagnosemöglichkeit in KFZ-Steuergeräten möglich. Der Continental Demonstrator hat die Funktionen einer TCU für ein mechanisch stufenloses Getriebe (CVT-Continuously Variable Transmission).

Die entwickelte Basis-Software deckt alle für den vorgesehenen Verwendungszweck notwendigen Funktionen ab. Diese beinhalten eine geeignete Initialisierung der Elektronik inklusive des Microcontrollers, Funktionsschnittstellen zu allen erforderlichen Ein- und Ausgangssignalen der Schaltung und die Bereitstellung eines konfigurierten OSEK-Betriebssystems mit mehreren zyklischen Tasks als Basis für ein Laufzeitsystem. Zusätzlich ist ein Softwaretreiber für die CAN-Schnittstelle und ein darauf aufsetzender CCP-Treiber integriert. Über diese Kommunikationsschnittstelle kann bei Bedarf mit einem geeigneten Applikationssystem gemessen und appliziert werden. Mit dem ebenfalls mitgelieferten CCP-Flashkernel ist es möglich, vom Applikationssystem aus einen Flashvorgang des Continental Demonstrators durchzuführen.

Der Continental Demonstrator beinhaltet Sensoren (für Drehzahl, Druck, Wählhebelstellung, ...) in einem Sensorcluster.

#### **IX.1.4. Funktionsprinzip des Continental Demonstrators**

##### **IX.1.4.1. Getriebesteuergerät VL381**

Das Getriebesteuergerät VL381 ist ein Vorort-Steuergerät. Die elektronischen Komponenten sind aufgrund der erhöhten Umwelтанforderungen in Hybridtechnologie auf einer Keramikleiterplatte angeordnet und im Klebe- und Bondverfahren elektrisch kontaktiert. Die Hybridelektronik ist im Steuergerätegehäuse integriert und wird über Bondverbindungen und Stanzgitter mit den Steckern und Sensoren verbunden.

Das VL381 wird zur Ansteuerung und Regelung eines Umschlingungs-Getriebes (CVT) betrieben. Es regelt Ventile zur Ansteuerung von Kupplung, Übersetzung und Kühlung bzw. Sicherheitsfunktionen. Im Steuergerät ist die Sensorik zur Erfassung von Drehzahlen, Druck, Wählhebelstellung und Temperatur integriert.

#### **Versorgungsspannung**

Die für die Versorgung des Steuergeräts notwendige Spannung wird über die Klemmen 30/31 des Getriebesteckers eingespeist. Die Versorgungsspannungen der Schaltungsmodule und der Sensoren wird durch diskrete Linearregler wie auch von einem Multifunktions-ASIC stufenweise generiert.

#### **Microcontroller ( $\mu\text{C}$ )**

Der Microcontroller ( $\mu\text{C}$ ) bedient und überwacht alle Funktionen des Steuergeräts. Der  $\mu\text{C}$  verfügt über einen internen Flash Speicher zur Aufnahme der Funktionssoftware und Parametersätze und hat folgende Eigenschaften:

- Typ Infineon TC1766

- 32-Bit Core
- Peripheral Control Processor – PCP – wird als I/O Prozessor verwendet
- 1,5MB Flashspeicher
- 56 KB SRAM
- 32 Channel ADC mit 10 Bit Auflösung
- 64-Bit local memory bus
- General Purpose Timer Array Module (GPTA)
- MultiCAN Module. Erlaubt Kommunikation mit Fahrzeug CAN und Applikations CAN.
- SPI Interface. Erlaubt Kommunikation mit EEPROM, Temperatur Sensor und Multifunktions-ASIC.

## EEPROM

Es wird ein EEPROM mit 32kBit (4KByte) verwendet, das über SPI angesprochen wird. Es dient als Speicher für u. a. Applikationsdaten, Fehlerspeicher und Snapshot-Daten.

## System Basis Chip for Transmission Control Units - Multifunktions-ASIC

Es kommt ein Multifunktions-ASIC (Hersteller Bosch) mit folgenden Funktionen zum Einsatz:

- *Spannungsversorgung*  
Die Eingangsspannung des Steuergeräts Kl.30 wird von diskreten Linearreglern auf eine niedrigere Spannung (UHx) gebracht. Diese dient als Versorgung für den Multifunktions-ASIC, welcher die weiteren internen Versorgungsspannungen des Steuergeräts (5V, 3,3V, 2,6V, 1,5V) erzeugt.
- *Sensorversorgungsspannungen*  
Das *System Basis Chip for Transmission Control Units* - Multifunktions-ASIC kann 3 unabhängige Versorgungsspannungen für Sensoren erzeugen (2x5V, 1x 3,3V oder 5V, ein- und ausschaltbar).  
  
Es wird eine Sensorspannungsversorgung mit 5V für die zwei Antriebssensoren verwendet. Die schaltbare Sensorversorgungsspannung wird genutzt um die Auszeiterfassung zu betreiben.
- *CAN-Transceiver*  
Der Multifunktions-ASIC enthält einen wakeup-fähigen highspeed (500kbaud eingestellt, 1000kbaud möglich) CAN-Transceiver. Dieser wird als Fahrzeug-CAN (mit Drossel, EMV-Filter verwendet und den µP weitergeleitet).
- *Monitoring Unit*  
Die Monitoring Unit dient zur Realisierung der Level 2 Diagnose äquivalent zum EGAS-Sicherheitskonzept. Die Kommunikation mit dem Hauptrechner findet über die SPI-Schnittstelle statt. Das Monitoring Module überwacht die Reset und Disable-Leitungen.
- *Wakeup Fähigkeit*  
Die interne Spannungsversorgung verfügt über eine Inhibit-Funktion, über die das Steuergerät ein- und ausgeschaltet werden kann. Dadurch ist sowohl ein wakeup durch den CAN-Transceiver als auch durch Kl.15 möglich.

- *Nachlauffunktion*

Es ist eine Nachlauffunktion im Multifunktions-ASIC realisiert, die bei Zündung aus (KI.15 aus) die TCU solange mit Spannung versorgt, bis der  $\mu$ C den Nachlauf im Multifunktions-ASIC abschaltet.

### **Treiber Bank für Aktuatoren**

Die Ansteuerung der Stromsteller-FETs für die Aktuatoren (VKA-Kupplungsventil, VKU- Übersetzungsventil, VKUE-Kühlventil) erfolgt durch ein ASIC der Fa. AMIS. Der ASIC bietet neben der Überstromabschaltung mit einstellbarer Schwelle auch die Diagnose der Kurzschlüsse und von Open-Load.

#### **IX.1.4.2. Continental Demonstrator - Conti-Steuergeräteelektronik**

Die auf dem Funktionsprinzip des VL381 basierende Conti-Steuergeräteelektronik wurde zwecks Zugänglichkeit als Leiterplatten-Steuergerät umgesetzt und ermöglicht darüber hinaus einen erweiterten low-level Zugriff über JTAG auf alle HW Ressourcen des TCU Microcontrollers.

Desweiteren wurde bei der Conti-Steuergeräteelektronik der im VL381 verwendete Microcontroller TC1766 ersetzt durch den aktuelleren Infineon Microcontroller TC1797 (CPU mit 180MHz):

- 32-Bit Core
- Peripheral Control Processor – PCP – wird als I/O Prozessor verwendet; 48kByte on-chip SRAM
- 4 MB Flashspeicher aufgeteilt in Program Flash und Data Flash
- 156 KB SRAM
- 2 ADC und Fast ADC mit 48 Kanal Mux
- Two General Purpose Timer Arrays
- MultiCAN Module (mit vier CAN Knoten und 128 Message Objekten)
- SPI Interface

Die Schnittstellen zwischen der Conti-Steuergeräteelektronik und dem Sensorcluster sind:

- Eingangssignale

Getriebeantriebsdrehzahl (2 x Hallgeber-Sensoren), Getriebeabtriebsdrehzahl (1 x Hallgeber-Sensor), Getriebeöltemperatur, Kupplungsdruck (1 x Drucksensor), Momentendruck bzw. Anpressdruck (1 x Drucksensor), Wählhebel (4 x Hallgeber-Sensor)

- Ausgangssignale

Proportionaldruckregler für die Übersetzungsverstellung, Anpressdruck und Kupplungsdruck

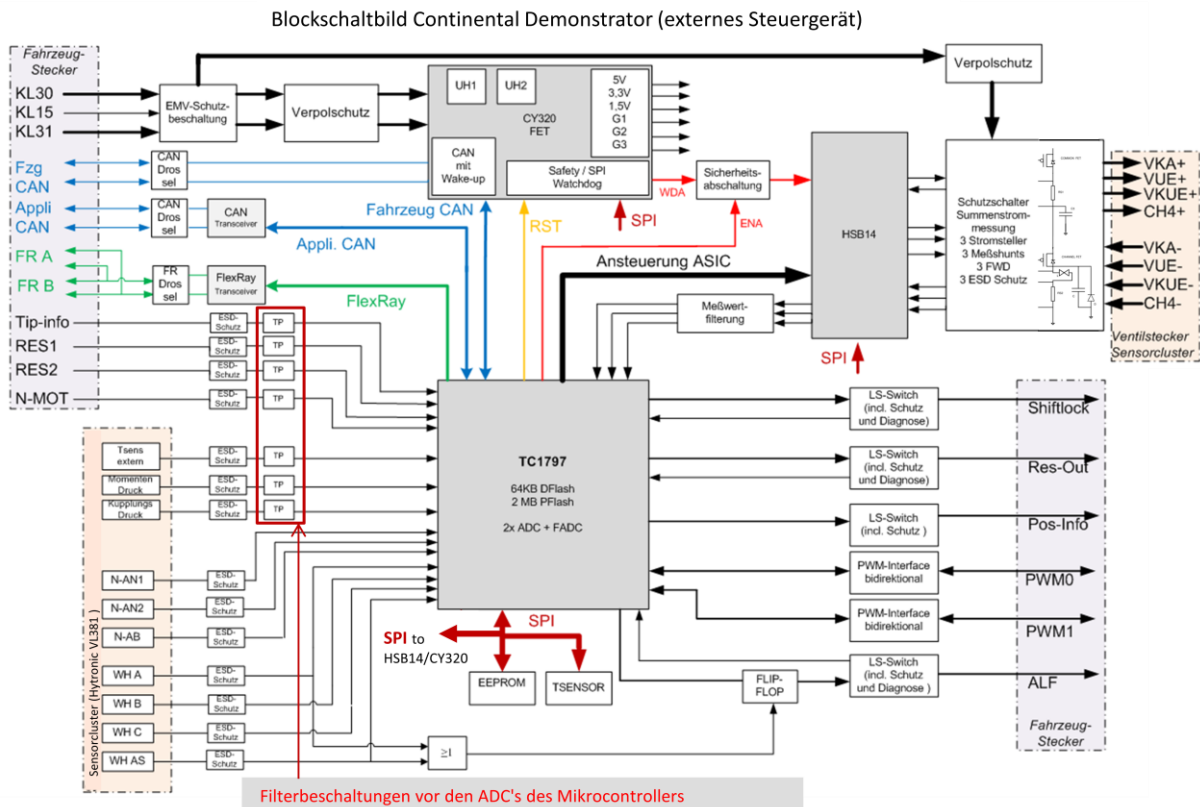


Abbildung IX.1: Blockschaltbild Continental Demonstrator

### IX.1.4.3. Continental Demonstrator - Sensorcluster

Damit an dem bestehenden Getriebe keine Änderungen notwendig wurden, verwendet der Continental Demonstrator in Abbildung IX.1 den Sensorcluster des bereits existierenden Steuergeräts VL381, d.h. die in das Getriebe eintauchenden Sensoren. Die Steuergeräteelektronik von VL381 wird aber nicht verwendet.

Im Sensorclustergehäuse VL381 sind integriert:

- Getriebeantriebsdrehzahl (2 x Hallgeber-Sensoren)

Als Antriebsdrehzahlsensoren (Blockschaltbild, Abbildung IX.1: N-AN1, N-AN2) werden zwei Hallgeber verwendet, die in einem Dom des Sensorclustergehäuses integriert sind und mittels eines Stanzgitters und über einen Kabelbaum an den **Continental Demonstrator** - Conti-Steuergeräteelektronik angebunden werden. Die Hallgeber werden vom Steuergerät mit 5V (5V des Multifunktions-ASIC) versorgt. Das digitale Signal des OC-Ausganges des Sensors wird dem  $\mu\text{C}$  mittels eines Pull-up auf 3,3V (3,3V von Multifunktions-ASIC) über einen Widerstand zur weiteren Verarbeitung zur Verfügung gestellt. Ein Blockkondensator am Hybrid-Eingang ist aus EMV - Gründen eingesetzt.

An diesen Sensoren wird ein Geberrad mit 48 Polpaaren vorbeibewegt. Dadurch wird ein Taktsignal erzeugt, das der  $\mu\text{C}$  zur Bestimmung der

Getriebeantriebsdrehzahl verwendet. Weil der Sensorabstand das 1,25fache des Polabstandes ist, kann auf die Drehrichtung geschlossen werden.

- Getriebeabtriebsdrehzahl (1 x Hallgeber-Sensor)

Als Abtriebsdrehzahlsensor (Blockschaltbild, Abbildung IX.1) wird ein Hallgeber verwendet, der in einem Dom des Sensorclustergehäuses integriert ist und mittels eines Stanzgitters und über einen Kabelbaum an den **Continental Demonstrator** - Conti-Steuergeräteelektronik angebunden wird. Der Hallgeber wird vom Steuergerät mit 5V (Sensorspannung 1 des Multifunktions-ASIC) versorgt. Das digitale Signal des Open Collector Ausgangs des Sensors wird dem  $\mu\text{C}$  mittels eines Pull-up auf 3,3V (3,3V von Multifunktions-ASIC) über einen Widerstand zur weiteren Verarbeitung zur Verfügung gestellt. Ein Blockkondensator ist am Eingang des **Continental Demonstrators** - Conti-Steuergeräteelektronik aus ESD und EMV - Gründen eingesetzt.

An diesem Sensor wird ein Geberrad mit 48 Polpaaren vorbeibewegt. Dadurch wird ein Taktsignal erzeugt, das der  $\mu\text{C}$  zur Bestimmung der Getriebeabtriebsdrehzahl verwendet.

- Wählhebel (4 x Hallgeber-Sensoren)

Zur Erkennung der Wählhebelposition werden vier Hallgeber verwendet, die in einem Dom des Sensorclustergehäuses integriert sind und mittels eines Stanzgitters und über einen Kabelbaum an die Conti-Steuergeräteelektronik angebunden werden. Die Hallgeber werden vom Steuergerät mit 5V (5V des Multifunktions-ASIC) versorgt. Das digitale Signal des Open-Collector-Ausgangs des Sensors wird dem  $\mu\text{C}$  mittels eines Pull-up auf 3,3V (3,3V von Multifunktions-ASIC) über einen Widerstand zur weiteren Verarbeitung zur Verfügung gestellt. Ein Blockkondensator ist am Eingang des **Continental Demonstrators** - Conti-Steuergeräteelektronik aus ESD und EMV - Gründen eingesetzt.

An diesen 4 Sensoren wird eine Magnetkulissee vorbeibewegt, die für jede aufeinander folgende Wählhebelposition einen charakteristischen 4-Bit-Code übermittelt, der sich vom benachbarten Halbbyte um je 1 bit unterscheidet.

- Kupplungsdruck (Drucksensor)

Als Kupplungsdrucksensor wird ein Absolutdrucksensor mit einem Arbeitsdruckbereich von 1 - 20 bar verwendet. Der Sensor wird mit einer Spannung von 5V (5V des Multifunktions-ASIC) versorgt. Die analogen Ausgangssignale der Sensoren werden über einen Kabelbaum dem  $\mu\text{C}$  zur Verfügung gestellt und dabei über einen Tiefpass 1.Ordnung gefiltert und auf 3,3V normiert. Ein Blockkondensator am Eingang des **Continental Demonstrators** - Conti-Steuergeräteelektronik ist aus ESD und EMV - Gründen eingesetzt.

- Momentendruck bzw. Anpressdruck (Drucksensor)

Als Momentendrucksensor wird ein Absolutdrucksensor mit einem Arbeitsdruckbereich von 1 - 70 bar verwendet. Der Sensor wird mit einer Spannung von 5V (5V des Multifunktions-ASIC) versorgt. Die analogen



Ausgangssignale der Sensoren werden über einen Kabelbaum dem  $\mu\text{C}$  zur Verfügung gestellt und dabei über einen Tiefpass 2.Ordnung gefiltert und auf 3,3V normiert. Ein Blockkondensator am Eingang des **Continental Demonstrators** - Conti-Steuergeräteelektronik ist aus ESD und EMV - Gründen eingesetzt.

### IX.1.5. Realisierung der Conti-Steuergeräteelektronik - Hardware

Zur Realisierung der Conti-Steuergeräteelektronik des Continental Demonstrators wurde das bestehende, mittels Hybridtechnik umgesetzte, VL381 (Vorort-Steuergerät) unter Berücksichtigung der geänderten Anforderungen, Randbedingungen und Umgebungsbedingungen als externes Steuergerät konzipiert und umgesetzt. Die Conti-Steuergeräteelektronik wurde als Leiterplatten-Steuergerät mit zum Teil geänderten Bauteilen und neuer Technologie (z.B. ASIC, ICs in „packaged version“) entwickelt, aufgebaut und getestet. Der Microcontroller TC1797 wurde neu eingeführt sowie auch eine JTAG Schnittstelle für die erweiterten low-level Zugriffsmöglichkeit auf alle HW Ressourcen des TCU Microcontrollers.

Für die Conti-Steuergeräteelektronik wurden die bestehenden EMC-Maßnahmen des VL381 Steuergeräts übernommen.

Die Abbildung IX.2 zeigt die Architektur des externen Steuergeräts - Continental Demonstrators mit Conti-Steuergeräteelektronik und Sensorcluster

Continental Demonstrators mit Conti-Steuergeräteelektronik und Sensorcluster

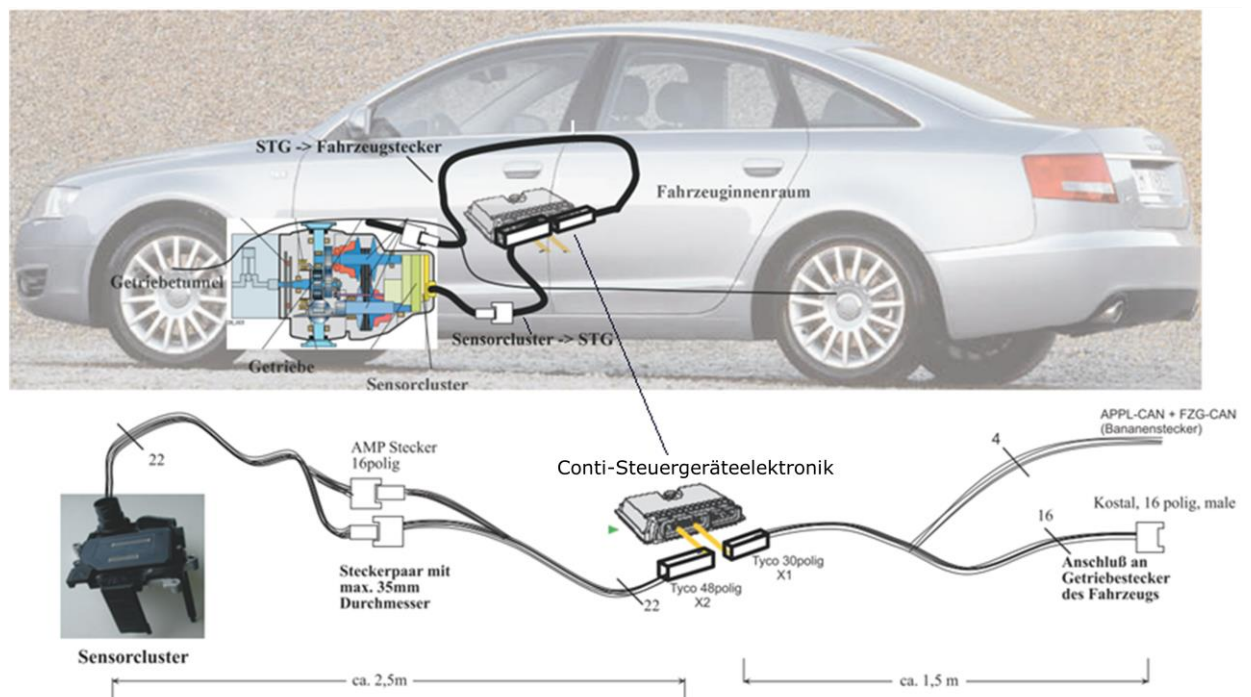


Abbildung IX.2: Continental Demonstrator

Zum Ankoppeln von zusätzlicher Hardware sowie für das messtechnische Überprüfen von Fehlerreaktionen dienen die angeordneten Testpunkte um den Microcontroller TC1797.

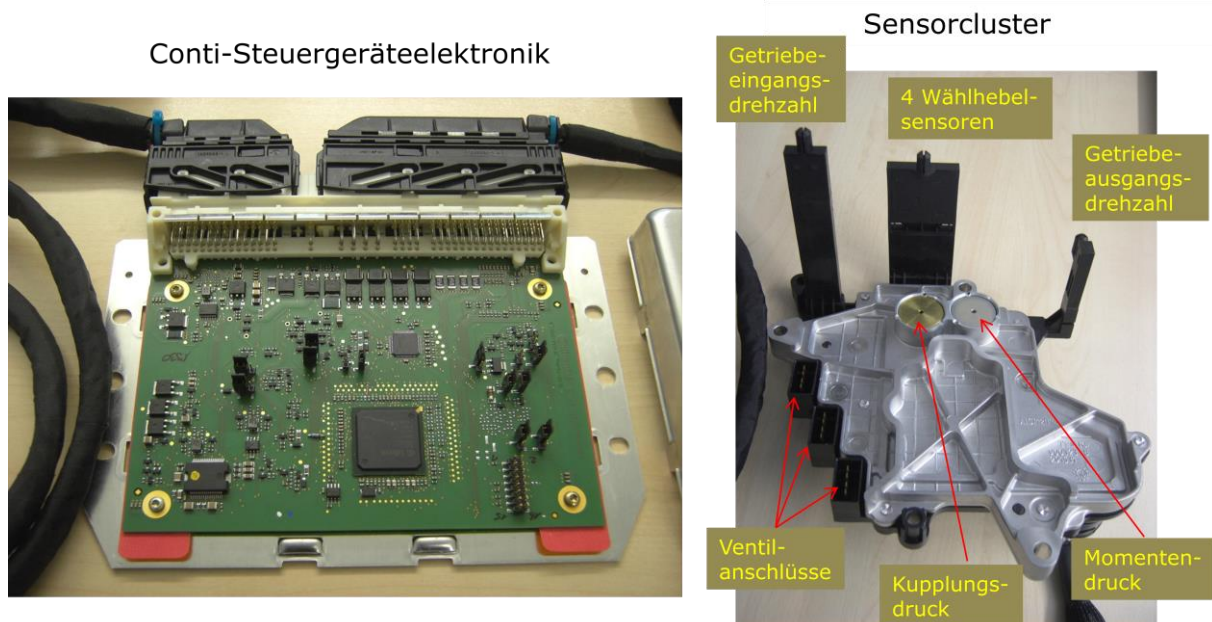


Abbildung IX.3: Conti-Steuergeräteelektronik und Sensorcluster

#### IX.1.6. Prüfung und Bereitstellung der Conti Demonstratoren

Im Arbeitspaket 3 wurden sechs **Continental Demonstratoren** (Conti-Steuergeräteelektronik und Sensorcluster) aufgebaut und mit einer **Continental Basis Software** (DIANA Bootloader und Betriebssystem OSEK mit Treibern für Sensoren, Aktuatoren) versehen und getestet.

Anschließend wurden drei Demonstratoren an die universitären Partner

- Friedrich-Alexander-Universität Erlangen-Nürnberg, Lehrstuhl für Zuverlässige Schaltungen und Systeme (LZS)
- Universität der Bundeswehr, München, Institut für Technik Intelligenter Systeme (ITIS)
- Universität Stuttgart, Institut für Technische Informatik (ITI)

übergeben und diese eingewiesen.

Es wurden nachfolgende Entwicklungsunterlagen übergeben:

- Kabelbaum Continental Demonstrator  
Fahrzeugstecker für Applikation CAN / Flexray mit Pinbelegung  
Kabelbaumbelegung
- Layout der Conti-Steuergeräteelektronik  
Bestückungsseite  
Testpunktbeschreibung  
Dokumentation der Testpunktlage

Mit den Continental Demonstratoren war es möglich, folgende Projekt-Ergebnisse in enger Kooperation mit den Verbundpartnern zu prüfen:

- Friedrich-Alexander-Universität Erlangen-Nürnberg, LZS :  
Test Anti-Aliasing Filter vor AD-Wandler des Microcontrollers
- Universität der Bundeswehr, München, ITIS:  
Diagnose in einem CAN basierten Systemverbund - 5 CAN-Knoten einschließlich Integration des Continental Demonstrators
- Universität Stuttgart, ITI:  
Test für digitale Logik zur Halbleiteranalyse

#### **IX.1.7. Literatur**

[1] Naunheimer, Harald / Bertsche, Bernd / Ryborz, Joachim / Novak, Wolfgang (Hrsg.): Automotive Transmissions. Fundamentals, Selection, Design and Application. Second Edition Berlin 2011.

#### **IX.2. Ergebnisse zur Aufgabe 3.2: „Produkt FMEA auf Steuergeräteebene unter Berücksichtigung erweiterter Fehlermodelle“**

Im Arbeitspaket AP3: „Fehleranalyse auf Steuergeräteebene & Demonstrator“ befasst sich die Aufgabe 3.2 „Produkt FMEA auf Steuergeräteebene unter Berücksichtigung erweiterter Fehlermodelle“ mit der Analyse von Failure Modes in Steuergeräten sowie von Fehlern in den Steuergeräten ICs und deren mögliche Auswirkung auf der Fahrzeug-Ebene unter Nutzung der Analysemethode „FMEA“.

##### **IX.2.1. Entwicklung der FMEA Struktur: Strukturelemente und Strukturbäume**

Zur Erstellung der Produkt-FMEA nach VDA-Band 4 (hier werden alle Demonstrator Funktionen mit ihren Failure Modes, während die Zündung eingeschaltet ist, betrachtet) wurde zunächst einmal eine Strukturanalyse vorgenommen.

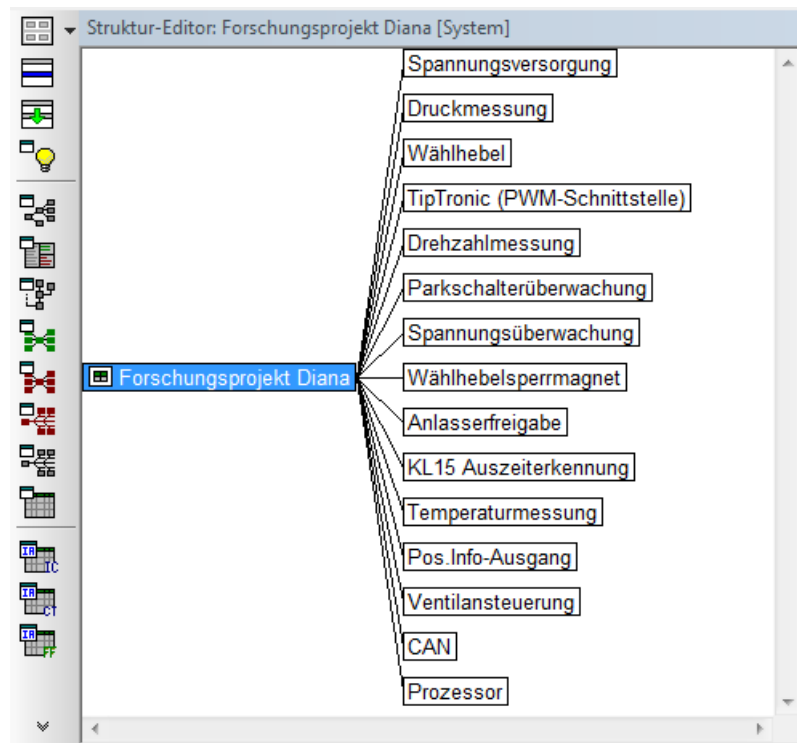


Abbildung IX.4: Strukturbaum der Continental Demonstrators - Conti-Steuergeräteelektronik

Die Produkt-FMEA beschreibt durch einzelne Systemelemente (Prozessor, Spannungsversorgung, Temperaturmessung, ...) die strukturellen Zusammenhänge der Conti-Steuergeräteelektronik.

Der in Abbildung IX.1 dargestellte Betrachtungsumfang wurde durch detailliertere Betrachtungen an den AD Wandlern für die Druckmessung, Überwachung der Versorgungsspannungen, Temperaturmessung, Signalüberwachung der Aktuatorik im Laufe des DIANA Projekts weiter verfeinert.

Hierbei wurde der Strukturbaum erweitert um die Strukturen bis zu den jeweiligen Bauteilen der diskreten Filterbeschaltungen vor dem AD Wandler des Microcontrollers (siehe Abbildung IX.5 bis Abbildung IX.8).

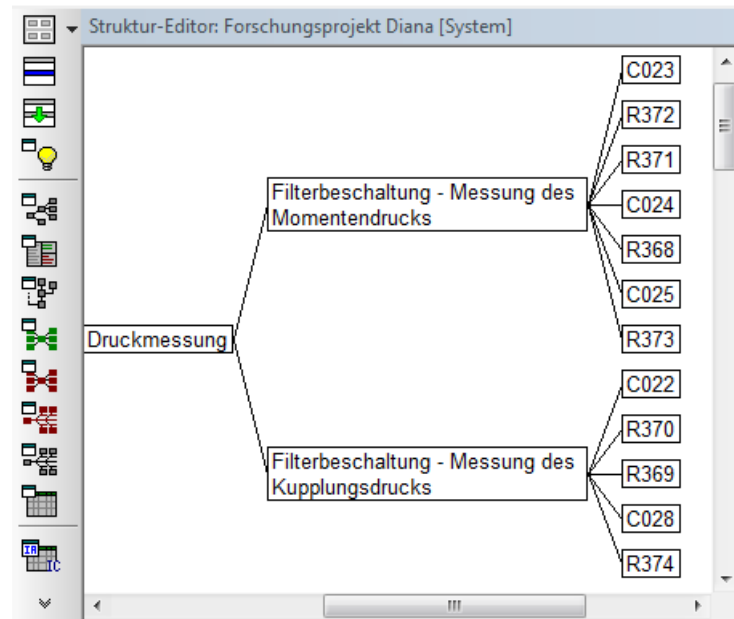


Abbildung IX.5: Filterbeschtaltung der Druckmessung

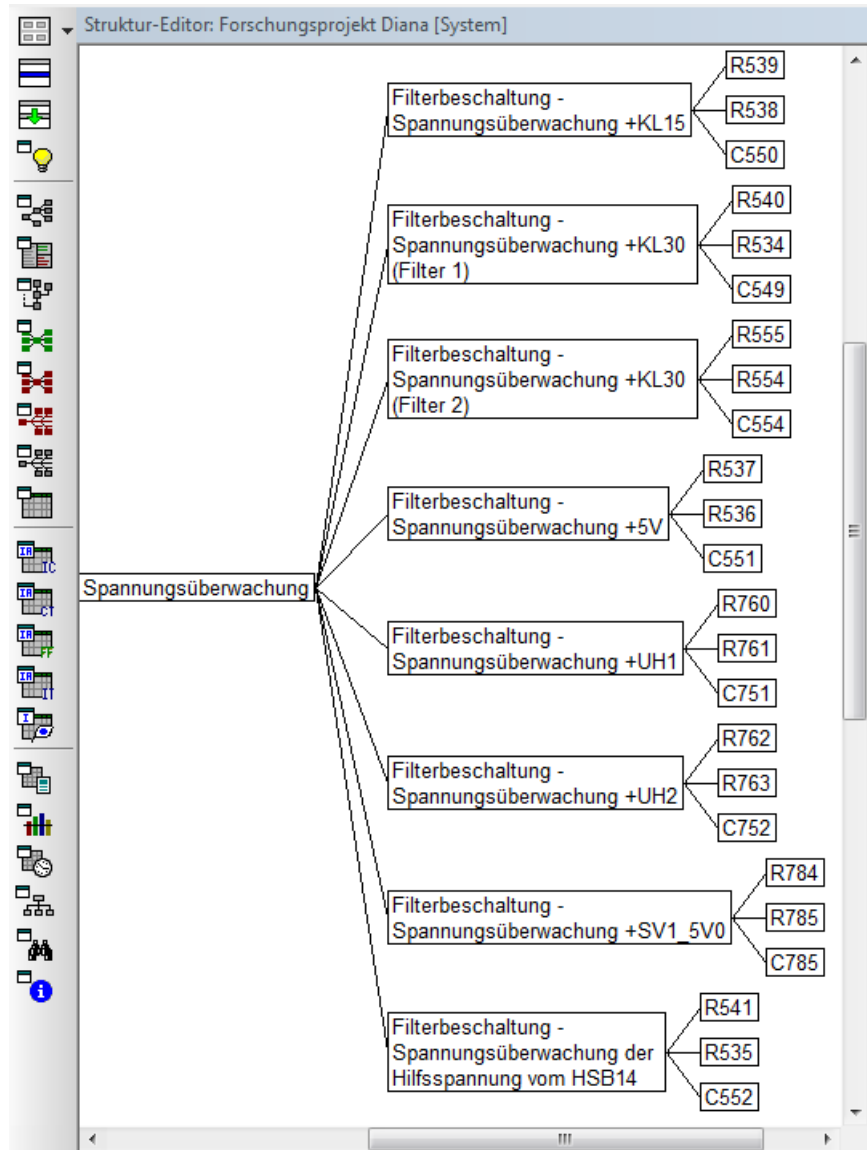


Abbildung IX.6: Filterbeschtaltung der Spannungsüberwachung

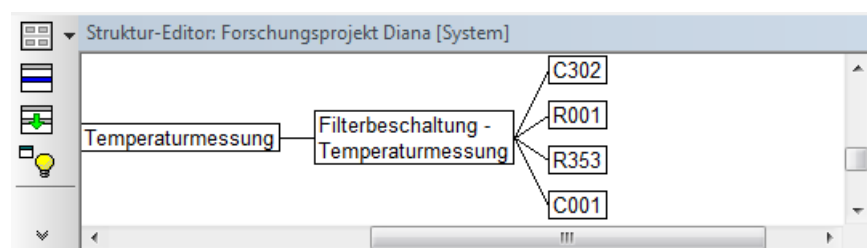


Abbildung IX.7: Filterbeschtaltung der Temperaturmessung

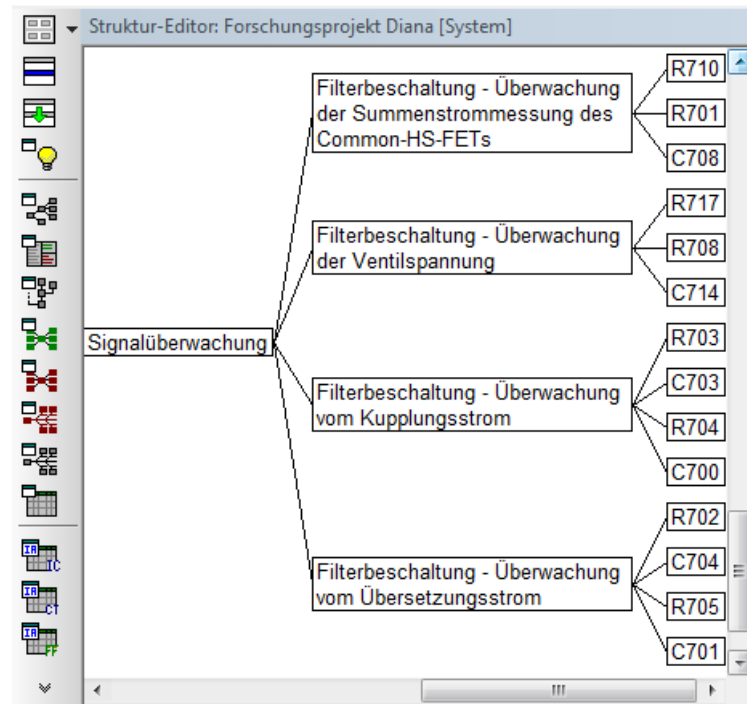


Abbildung IX.8: Filterbeschaltung der Signalüberwachung der Aktuatorik

Die Systemstruktur mit den einzelnen Systemelementen wurde durch eine Funktionsanalyse erweitert. Durch Beschreibung der Funktionalität der einzelnen Systemelemente wurden die Ursachen-Wirkungs-Beziehungen dargestellt.

Jedem Systemelement wurden in den relevanten Zweigen entsprechende Funktionen bzw. Aufgaben zugeordnet und untereinander logisch verknüpft mittels eines Funktionsnetzes.

### IX.2.2. Entwicklung der FMEA Struktur: Failure Modes und Fehlernetze

Bei der Produkt FMEA wurden dann die Strukturelemente/Funktionen in allen relevanten Strukturbäumen und Funktionsbäumen um passive elektrische Bauelemente erweitert. Alle Fehlerbäume wurden um die Failure Modes nach SN29500 in diesen Bauelementen sowie die Maßnahmen, um diese Failure Modes zu entdecken und zu vermeiden, erweitert. Jeder Failure Mode wurde durch seine Auftretenswahrscheinlichkeit  $A = 1-10$  nach SN29500 charakterisiert. Jede Entdeckungs-/Vermeidungsmaßnahme eines Failure Modes wurde durch eine entsprechende Entdeckungswahrscheinlichkeit  $E = 1-10$  charakterisiert. Failure Modes, die nicht entdeckt/vermieden werden können, haben  $E=10$ .

In der DIANA Vorhabensbeschreibung wird gefordert:

„In Phase M 3.2.3 werden die Erkenntnisse aus den vorangegangenen Phasen anhand des realen **Demonstrators** praktisch überprüft. Dazu werden Fehler auf Chipebene injiziert und Fehlerauswirkung auf Fahrzeugebene analysiert.“

Die Produkt FMEA wurde um Failure Modes für alle ICs - insbesondere des Infineon Mikroprozessors - verfeinert. Weil Tests im Fahrzeug und im Fahrzeuggetriebe aus

Sicherheitsgründen nicht möglich sind, wurde eine Testumgebung „*Continental Mobile Steuergeräteperipherie VL381*“ aufgebaut, die es gestattet, wichtige Failure Modes zu überprüfen. Weil HW Fehler auf Chip Ebene kaum zu injizieren sind, wurden Tests mit Bond/Pin- Abrissen durchgeführt (>>>>>>>> Pin Abriss an ADC/MUX Eingang von  $\mu\text{C}$ ).

„Es wird die Wirkungskette von Fehlern von der Chip- zur Systemebene studiert. Erkenntnisse aus diesen Analysen fließen, soweit notwendig, in die Optimierung des FMEA-Prozesses ein. Darüber hinaus fließen Erkenntnisse in die Optimierung und Verwertung der in AP2 entwickelten Selbsttest und Selbstdiagnoseverfahren ein sowie in zukünftige Überlegungen zur Gestaltung des elektronischen Gesamtsystems im Automobil (AP4).“

Die Wirkungsketten von Fehlern auf Chip-Ebene zu Fehlern auf System-Ebene (Fahrzeugebene) wurde über Fehlerbäume in der Produkt FMEA behandelt und an ausgewählten Beispielen in der Testumgebung „*Continental Mobile Steuergeräteperipherie VL381*“ nachvollzogen. In der Produkt FMEA werden außerdem Entdeckungs- und Vermeidungsmaßnahmen angegeben, die beim Auftreten von Chip-Fehlern durch die in AP2 entwickelten *Selbsttest und Selbstdiagnoseverfahren* durchgeführt werden müssen, um das Fahrzeug vor Fahrtantritt oder während der Fahrt in einen sicheren Zustand zu bringen.

## Resümee

Die Durchführung einer Produkt-FMEA mit gezielten Detailfragen zur Fehlerauswirkung und deren Wirkungskette hat wertvolle Anregungen zu Anforderungen an neue komplexe Bauteile beigeleitet. Als Ergebnis hieraus werden zukünftig komplexe Zukaufteile, wie z.B. integrierte Schaltkreise oder Sensoren, hinsichtlich der zu erwartenden internen Fehler und deren Auswirkungen noch genauer beleuchtet. Dies mündet u.a. in einer geänderten Produkthanforderung und Dokumentation für neu zu entwickelnde Bauteile und Komponenten.

### IX.2.3. Failure Modes und deren Wirkungsketten

#### IX.2.3.1. Erweiterung um Failure Modes für diskrete Bauelemente

Die Produkt FMEA wurde um weitere relevante Failure Modes für diskrete Bauelemente verfeinert:

Über die Produkt FMEA wurden alle Failure Modes, die nicht entdeckt/vermieden werden können (Entdeckungswahrscheinlichkeit  $E=10$ ) bzw. nicht hoch-zuverlässig (Entdeckungswahrscheinlichkeit  $E=3$ ) sind, nochmals untersucht und Failure Modes in den Anti-Aliasing Filtern für Sensor Signale vor den ADCs im Infineon Microcontroller für die weitere Untersuchung durch die FAU Erlangen-Nürnberg ausgesucht.

Mit den neuen Diagnosemaßnahmen zur Entdeckung fehlender/defekter Kondensatoren in Anti-Aliasing Filtern durch die FAU Erlangen-Nürnberg konnte die Entdeckungswahrscheinlichkeit bei derartigen Fehlern von  $E=10$  auf  $E=1$ , bzw.  $E=3$



auf  $E=1$  gesenkt werden und damit die Reziprozitätszahl  $RPZ (= A \times B \times E)$  von typ.  $RPZ=120$  auf  $RPZ=12$ .

Für die Diagnoseerweiterung wurde eine Testhardware und Software vom Lehrstuhl für Zuverlässige Schaltungen und Systeme der FAU Erlangen-Nürnberg entwickelt. Für die quantitative Bewertung der Diagnoseerweiterung wurde für den Fehlerfall „unterbrochene, fehlende Kondensatoren/Widerstände“ die Diagnostic Coverage nach ISO26262 ermittelt:

Ohne „safety mechanism“, d.h. keine Diagnoseerweiterung

$DC_{C24, C25 \text{ unterbrochen}} = 0\%$  (Entdeckungswahrscheinlichkeit  $E=10$ )

Mit „safety mechanism“ - Diagnoseerweiterung bei Zündung „EIN“ durch die Ergebnisse des BMBF Förderprojekts DIANA:

$DC_{C24, C25 \text{ unterbrochen}} = 99\%$  (Entdeckungswahrscheinlichkeit  $E=1$ )

Die Filterbeschaltungen vor ADC's des Microcontrollers sind im Blockschaltbild des Continental Demonstrators, Abbildung IX.1 dargestellt:

Filtereingangsbeschaltung vom Drucksensor vor dem ADC des TC1797:

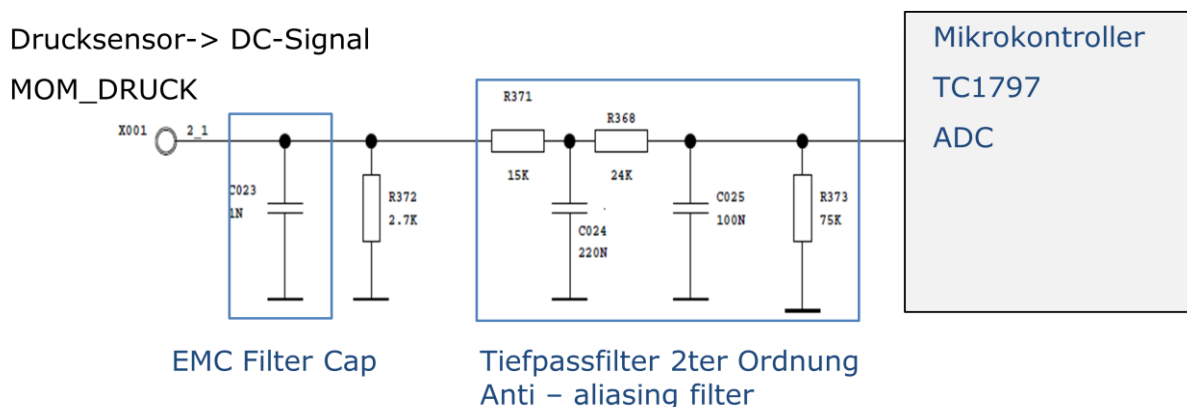


Abbildung IX.9: Filtereingangsbeschaltung vom Drucksensor vor dem ADC des TC1797

### IX.2.3.2. Erweiterung um Failure Modes für ICs

Der im AP 4 Aufgabe 4.3 vorgestellte Ringmodulator fand nicht Eingang in die Produkt-FMEA, weil Continental im Wirkungsprinzip des AD Wandlers als Ringoszillator keine Möglichkeit der Fehlerraten senkung im Vergleich zu dem bisher verwendeten Delta Sigma AD-Wandler sieht. Hingegen ist nach unserem jetzigen Kenntnisstand eine Kostenoptimierung mit dem neuen AD-Wandler möglich.

Die Produkt FMEA wurde um weitere relevante von Infineon festgelegte Failure Modes für ICs verfeinert:

Die Strukturelemente/Funktionen in allen relevanten Strukturbäumen und Funktionsbäumen wurden um IC Funktionen, insbesondere des Infineon Mikroprozessors (Infineon Tricore Core) erweitert.

Alle Fehlerbäume wurden um die Failure Modes mit Auftretenswahrscheinlichkeit **A** in diesen ICs, wie sie aus Unterlagen (Hersteller FMEA) der IC Hersteller sowie der DIANA Projektpartner bekannt waren, erweitert. Außerdem wurden Entdeckungs- und Vermeidungsmaßnahmen zu diesen Failure Modes, wie sie in anderen Conti Getriebesteuergeräten zu finden sind, implementiert.

Die Produkt-FMEA auf Steuergeräteebene beinhaltet neben den Demonstrator Funktionen mit ihren Failure Modes *für diskrete Bauelemente*, die Failure Modes für ICs, wie dem  $\mu$ C Infineon TC1797 (Infineon Tricore Core), die während Zündung „EIN“ und Nachlauf auftreten können.

Die Universität Stuttgart, ITI hat einen strukturellen softwarebasierten Selbsttest SBST entwickelt (siehe Fachvortrag „Ansätze struktureller Testmethoden bei digitaler Automobilelektronik“, Prof. Wunderlich, Universität Stuttgart (ITI) zum 2. Meilensteinreview in Dresden, 120329\_DIANA\_MilestoneReview\_2\_Fachvortrag\_AP4\_Wunderlich.pdf), welcher den digitalen Teil des Infineon Microcontrollers (Tricore Rechnerkern) bis auf Gatterebene in weniger als 1sec testen kann.– Mit diesem Test lassen sich erstmals auch bisher nicht abgedeckte Fehlerfälle mit hoher Diagnostic Coverage DC von typ. 80%-90% entdecken. Die typ. Fehlerrate des Tricore-Rechenkerns kann durch diese Testmethode um 80%-90% gesenkt werden. Die Entdeckungswahrscheinlichkeit verbessert sich damit von E=10 auf E=2-3.

In der Produkt FMEA sind sogenannte Top-Level Failure Modes eingeführt worden, die in einer Produkt FMEA beschreiben, welche System-Fehler z.B. in einem Getriebe, bzw. im Fahrzeug festgestellt werden, wenn ein Failure Mode in einem IC oder einem diskreten elektronischen Bauteil auftritt.

### **IX.2.3.3. Wirkungsketten für Failure Modes in ICs**

Der Infineon  $\mu$ C hat alleine mehrere Hundert Failure Modes. Es werden aus Gründen der Übersichtlichkeit in der Produkt FMEA nur Failure Modes im Infineon Tricore Core betrachtet und dabei der  $\mu$ C Failure Mode - *Ausführung von ALU and MAC Befehlen - ALU data path - D.C. fault model - Corrupted data* mit seinem Fehlerbaum zu den Top-level Fehlern dargestellt:

B=8 (B-Max=8) Fahrzeug startet nicht, oder bleibt liegen

B=8 (B-Max=8) Ruhestromvorgabe wird nicht eingehalten

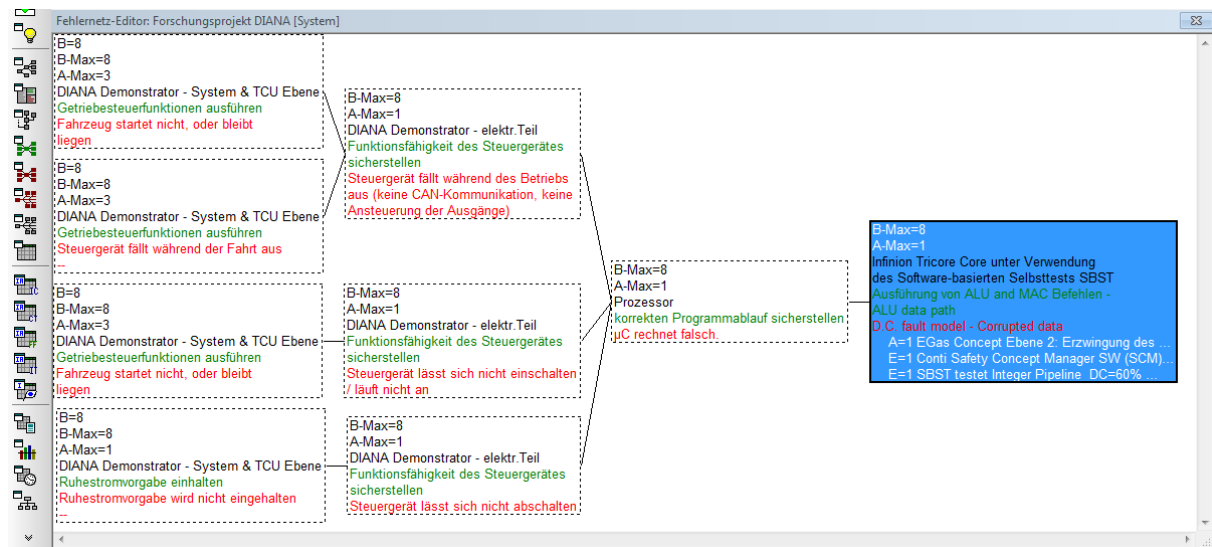


Abbildung IX.10: Wirkungsketten von Failure Modes in ICs

An obigem Fehlerbaum kann man sehen:

Der Top-Level Failure Modes „Komforteinschränkung (verschlechtertes Regelverhalten)“ hat eine Bedeutung (Fehlerschwere) von **B** = 8 (Funktionsfähigkeit des Fahrzeugs **stark** eingeschränkt). Damit hat auch der IC Fehler im Fehlerbaum eine Bedeutung  $B_{\max} = 8$ .

Der IC Fehler (Ausführung von ALU and MAC Befehlen - ALU data path - D.C. fault model - Corrupted data) im Fehlerbaum hat eine Auftretenswahrscheinlichkeit von  $A = 1$ . Diese Auftretenswahrscheinlichkeit ist für die Infineon TC17xx  $\mu$ C Familie im Rahmen einer FMEDA für Conti ermittelt worden. Aus dieser FMEDA stammen auch die Failure Modes des Infineon Tricore Cores, die in der Product FMEA für den Continental-Demonstrator benutzt werden.

Der IC Fehler (Ausführung von ALU and MAC Befehlen - ALU data path - D.C. fault model - Corrupted data) konnte bereits bisher in Conti TCUs entdeckt werden über die SCM SW (Conti Safety Concept Manager) mit einer Diagnostic Coverage von  $DC=90\%$  (-> Entdeckungswahrscheinlichkeit  $E=2$ ).

Mit dem Software-basierten Selbsttest SBST der Universität Stuttgart - ITI während des Hochlaufs des Demonstrators bei Zündung „EIN“, kann zusätzlich eine Diagnostic Coverage von  $DC=60\%$  gewonnen werden. Damit ist  $DC_{\text{gesamt}} = (1 - [(1 - 0,9) * (1 - 0,6)]) * 100\% = 96\%$ . Die Entdeckungswahrscheinlichkeit kann von  $E=2$  auf  $E = 1$  verbessert werden.

Der Fehlerbaum zeigt, dass die Reziprozitätszahl somit von  $RPZ (B=8 * A=1 * E=2) = 16$  auf  $RPZ (B=8 * A=1 * E=1) = 8$  gesenkt werden kann.

Der obige Fehlerbaum ist ein Beispiel, wie für viele andere Failure Modes im Infineon Tricore Core die Entdeckungswahrscheinlichkeit **E** mit Hilfe des SBSTs verbessert werden kann.

#### IX.2.3.4. Wirkungsketten für Failure Modes in diskreten Bauelementen

Die Wirkungskette von Bauelemente-Funktion zu System-Funktion sowie von Bauelemente-Fehler zu System-Fehler wird über den nachfolgenden Fehlerbaum für den Bauelemente-Fehler „Kondensator C023, C024, C025 kurzgeschlossen“ zum Top-Level Failure Mode „Komforteinschränkung (verschlechtertes Regelverhalten)“ aufgezeigt.

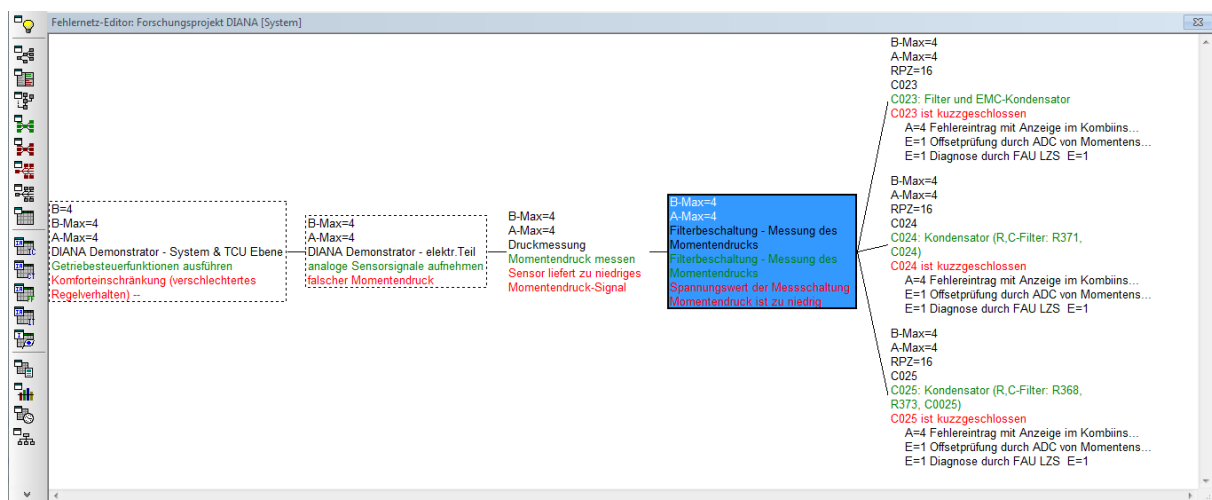


Abbildung IX.11: Wirkungsketten für Failure Modes in diskreten Bauelementen

An obigem Fehlerbaum (Abbildung IX.11) kann man sehen:

Der Top-Level Failure Mode „Komforteinschränkung (verschlechtertes Regelverhalten)“ hat eine Bedeutung  $B = 4$  (Funktionsfähigkeit des Fahrzeugs stark eingeschränkt). Damit haben auch alle Bauelemente-Fehler im Fehlerbaum eine Bedeutung  $B_{\max} = 4$ .

Die drei Bauelemente-Fehler (C023, C024, C025 ist kurzgeschlossen) im Fehlerbaum haben alle eine Auftretenswahrscheinlichkeit von  $A = 4$ . Diese Auftretenswahrscheinlichkeit für Kurzschlüsse in Keramik Kondensatoren ist durch Beroullini ermittelt worden.

Kurzschlüsse in den Filterkondensatoren C023, C024, C025 konnten bereits bisher in Conti TCUs entdeckt werden durch Messung des DC Signals über den ADC im Infineon Tricore  $\mu C$ , allerdings nur mit einer Entdeckungswahrscheinlichkeit von  $E=3$ .

Mit dem Filter-Test der FAU Erlangen-Nürnberg während des Hochlaufs des Demonstrators bei Zündung „EIN“ kann die Entdeckungswahrscheinlichkeit auf  $E = 1$  verbessert werden.

Der Fehlerbaum zeigt, dass die Reziprozitätszahl somit von  $RPZ (B=4 * A=4 * E=3) = 54$  auf  $RPZ (B=4 * A=4 * E=1) = 16$  gesenkt werden kann.

#### IX.2.4. Überprüfen der Produkt FMEA am Gesamtsystem Automobil

Der Continental-Demonstrator wurde in einer Testumgebung „Continental Mobile Steuergeräteperipherie VL381“ eingebracht, die es gestattet, die Umgebung einer TCU im Fahrzeug zu simulieren, also z.B. den Öldruck im Getriebe an der Kupplung auf den Momenten-Drucksensor auszuüben.



Abbildung IX.12: Testumgebung „Continental Mobile Steuergeräteperipherie VL381“

Im blauen Testkoffer wird der SensorCluster des Continental-Demonstrators eingesetzt. Im Testkoffer erzeugt eine Pumpe einen Testdruck auf den Momenten Sensor. Das SensorSignal läuft über ein Verbindungskabel vom SensorCluster zur Conti-Steuergeräteelektronik (liegt rechts auf dem Tisch). Über das rechte Kabel hängt der Continental-Demonstrator über ein CAN Netzwerk an einem speziellen OBD Tester. Dieser kann Fehlerzustände sowie empfangene Sensordaten abfragen.

Weil HW Fehler auf dem Infineon  $\mu C$  nicht zu realisieren sind – hier hätte uns Infineon präparierte ICs geben müssen – wurden Tests mit Pin- Abrissen am ADC/MUX Eingang *Pin AE1 (AN0) - CINA\_MESS\_MOM\_DRUCK* des Infineon  $\mu C$ s durchgeführt und die Reaktion des Continental-Demonstrators über den OBD Tester geprüft und mit den Entdeckungs- und Vermeidungsmaßnahmen in der Produkt FMEA verglichen. Wie der folgende Fehlerbaum aus der Produkt FMEA zeigt, kann



[4] Birolini, Alessandro: Reliability Engineering. Theory and Practice. Sixth Edition Berlin 2010.

[5] Börcsök, Josef: Funktionale Sicherheit. Grundzüge sicherheitstechnischer Systeme. 2. überarbeitete Auflage Heidelberg 2008.

[6] Bertsche, Bernd: Reliability in Automotive and Mechanical Engineering. Determination of Component and System Reliability. Berlin 2008.

### **IX.3. Ergebnisse zur Aufgabe 3.3: „Diagnose von Komponenten und Modulen im CAN-basierten Systemverbund“**

#### **IX.3.1. Motivation und Ziele**

Mit der stetig zunehmenden Verwendung verschiedener Halbleiterelemente und deren dichter werdender Integration in Steuergeräten im Automobilbereich stellt die Diagnose von Fehlern in den elektronischen Komponenten ein immer größeres Problem dar. Derzeit ist es beispielsweise nicht möglich, Elektronikfehler auf der Halbleiterebene über handelsübliche Diagnosegeräte, wie sie in KFZ-Werkstätten verwendet werden, ausfindig zu machen. In der Regel können nur Fehlfunktionen einer gesamten Komponente wie einem Steuergerät oder einem Sensor ausgemacht werden, jedoch ohne dass dies einen Rückschluss auf die eigentliche Fehlerquelle zulässt. Insbesondere Halbleiterelemente sind anfällig für Defekte auf der Schaltungsebene, beispielsweise durch Störungen bei der Vernetzung innerhalb einzelner Halbleiterelemente und bei deren Verbindungen. Ebenso wenig kann ein Ausfall dieser Elemente oder ein permanenter beziehungsweise sporadisch auftretender Funktionsfehler detektiert werden. Oft führt dieses Problem zu langen Werkstattlaufzeiten und erhöhten Reparaturkosten, wenn eine Komponente aufgrund ihres Defektes komplett ausgetauscht werden muss. Die Möglichkeit, Fehler auch feingranularer auf Ebene der Leiterkarte zu bestimmen bietet die Chance, Reparaturen sowohl kostengünstiger als auch ressourcenschonender durchzuführen. Gerade der letzte Punkt hat weiter an Bedeutung gewonnen. Darüber kann mit entsprechendem Zugang auf die Halbleiterebene auch tiefer in eine Komponente eingegriffen werden und im Falle von reprogrammierbaren Elementen wie FPGAs (Field Programmable Gate Arrays) sogar eine physikalische Reparatur oder das Austauschen von Hardware verhindert werden. Dies ist möglich, indem man bei begrenzten Fehlern bestimmte Funktionen über andere Schaltpfade realisiert und somit den defekten Teil der Hardware einfach abschaltet. Solch eine Reparaturmethode kann letztlich die Werkstattkosten drastisch senken, wenn ein Reparaturvorgang ohne einen tatsächlichen Eingriff in die elektronischen Komponenten eines Automobils stattfindet.

Mit der Realisierung dieser Ziele kann ein Großteil der schon derzeit in Fahrzeugen installierten Infrastruktur weiter benutzt werden, was eine standardmäßige Ausstattung zukünftiger Modelle weiter vereinfacht. So sind die elektronischen Fahrzeugkomponenten wie Steuergeräte oder Sensoren bereits miteinander vernetzt. Bei der Fehlersuche kann ein standardisiertes Diagnosegerät angeschlossen werden, um den Status dieser Komponenten auszulesen und so eine

defekte Komponente zu identifizieren. Mit dem Diagnosegerät kann nun über einen Gateway auf den CAN-Bus zugegriffen und über diesen der Fehlerspeicher einzelner Komponenten ausgelesen werden. Als Ergebnis der Arbeiten in diesem Projekt ist nun auch die Prüfung von einzelnen Halbleiterelementen im Systemverbund möglich.

Im folgenden Abschnitt wird zunächst das Konzept des entwickelten Verfahrens zur Fehlererkennung dargestellt. Anschließend wird der Aufbau des Demonstrators im Allgemeinen und die Integration der Conti-Steuergeräteelektronik im Speziellen erläutert. Darauf aufbauend wird die praktische Umsetzung der Fehlererkennung im Demonstrator mittels eines Fehleremulators erläutert. Abschließend wird der entwickelte Bootloader zur Steuerung von Testdurchläufen vorgestellt und der während des Projektes erreichte Fortschritt zusammengefasst.

### **IX.3.2. Konzept**

Für die Fehlerdiagnose auf Halbleiterebene mit Hilfe von standardisierten Diagnosegeräten mussten einige Voraussetzungen geschaffen werden. Die Anforderung war es, mit einem Diagnosegerät, das an einer OBD-II-Schnittstelle angeschlossen ist, über einen Gateway Nachrichten an eine Teststeuerung in den CAN-Knoten schicken zu können. Diese Teststeuerung sollte wiederum mit den einzelnen Halbleiterkomponenten verbunden sein, um entsprechende Prüfprozeduren durchführen zu können. Außerdem ist es erforderlich, im CAN-Knoten Testnachrichten von anderen eintreffenden Nachrichten unterscheiden zu können. Für die Teststeuerung war daher ein Interface erforderlich, welches die Ausführung verschiedener Testbefehle ermöglicht. Für die Steuerung von Tests der individuellen Bausteine auf Halbleiterebene konnten dabei schon bestehende Standards wie JTAG verwendet werden. Durch die Nutzung einer dedizierten Teststeuerung mit der eine bestehende CAN-Komponente erweitert wird, war es möglich, Testprozeduren beliebig an die gewünschte Methodik der Fehlersuche anzupassen. Insbesondere der Einsatz eines Field Programmable Gate Arrays (FPGA) stellte darüber hinaus die Option einer Rekonfiguration der Testlogik für unterschiedliche Prüfungsszenarien bereit.

In diesem Abschnitt wird die Verbindung der Komponenten sowie die Integration mit einem Testmodul erläutert. Anschließend wird beschrieben, wie die erforderlichen Nachrichten auf dem CAN-Bus ausgetauscht werden.

#### **IX.3.2.1. Koppelung der Komponenten**

Für die Prüfung von einzelnen Bauteilen wie FPGAs oder CPUs müssen diese jeweils mit einem JTAG-Interface ausgestattet sein, welches Zugriff auf den Scan-Path der Bauteile gewährt. Heutzutage verfügen viele handelsübliche Chips über ein solches Interface (1). Zur Steuerung der Tests werden CAN-Knoten nach dem bisherigen Design lediglich mit einer weiteren Steuerungskomponente versehen, die die Testabläufe mit Hilfe der entsprechenden JTAG-Interfaces steuert. Diese ist, wie in Abbildung IX.14 dargestellt, mit den JTAG-Interfaces aller im CAN-Knoten enthaltenen elektronischen Bauteile verbunden. Wie im JTAG-Standard vorgegeben, sind alle Bauteile parallel an die TCK- und TMS-Leitungen angeschlossen während die TDI- und TDO-Ein- beziehungsweise Ausgänge in Reihe geschaltet sind.



### IX.3.2.2. Kommunikation über den CAN-Bus zur Teststeuerung

Die im vorhergehenden Abschnitt beschriebene Koppelung der einzelnen Komponenten wurde dazu genutzt, mit Hilfe des Teststeuerungsmoduls Prüfungen der verschiedenen Halbleiterelemente durchzuführen. Dafür war es erforderlich, dass entsprechende Nachrichten von einem Diagnosegerät über den OBD-II-Gateway an einen CAN-Knoten geschickt werden konnten. Des Weiteren musste innerhalb des CAN-Knotens feststellbar sein, ob es sich bei einer eintreffenden Nachricht um einen Testbefehl oder eine Nachricht für den normalen Laufzeitbetrieb handelt. Dafür wurde der CAN-Controller, der die Kommunikation des CAN-Knotens mit dem CAN-Bus regelt, zusätzlich mit dem Teststeuerungsmodul verbunden. Der Ablauf erfolgt dann nach folgendem Schema: Bei Ankunft einer Nachricht wird deren Identifizier geprüft. Ein CAN-Controller kann so programmiert werden, dass er, falls die Nachricht den für diesen Knoten zutreffenden Identifizier trägt, außerdem prüft, ob in den Nutzdaten des Data Frames ein bestimmtes Bitmuster enthalten ist (2). Abhängig von diesem Bitmuster entscheidet sich, ob es sich um eine Nachricht zur Steuerung der eigentlichen Komponentenfunktionalität oder zum Zugriff auf die Teststeuerung handelt. Im zweiten Fall werden die Nachrichten anstatt an die übrigen Komponenten an das Teststeuerungsmodul weitergeleitet. Somit können auch ohne physikalischen Eingriff Halbleiterbausteine geprüft werden. Entsprechende Status-Codes oder Prüfergebnisse werden genauso wieder vom

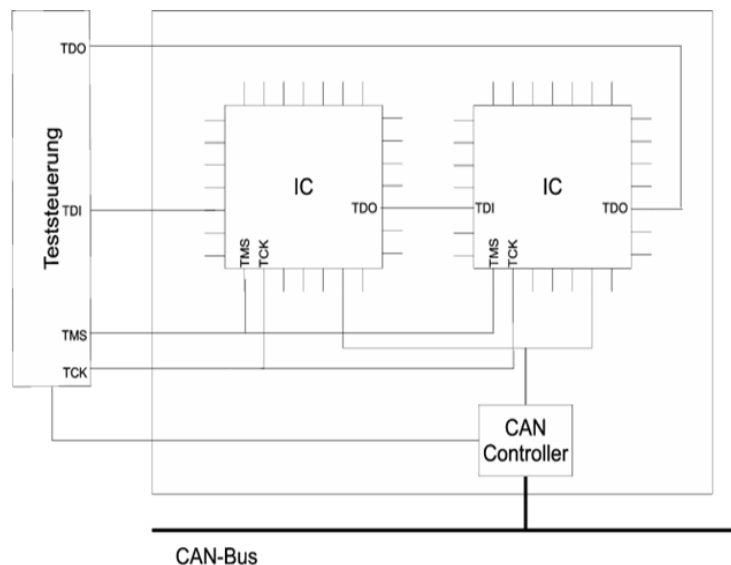


Abbildung IX.14: Verbindung der TCK-, TMS-, TDI- und TDO-Leitungen der JTAG Interfaces mit der Teststeuerung.

Testmodul an den CAN-Controller weitergeleitet und von diesem an den CAN-Bus geschickt. Abbildung IX.15 zeigt die Position der Daten in einem Data Frame, wie er über den CAN-Bus von einem Diagnosegerät oder anderen Komponenten verschickt wird. Die Verarbeitung im CAN-Controller ist in Abbildung IX.16 dargestellt.

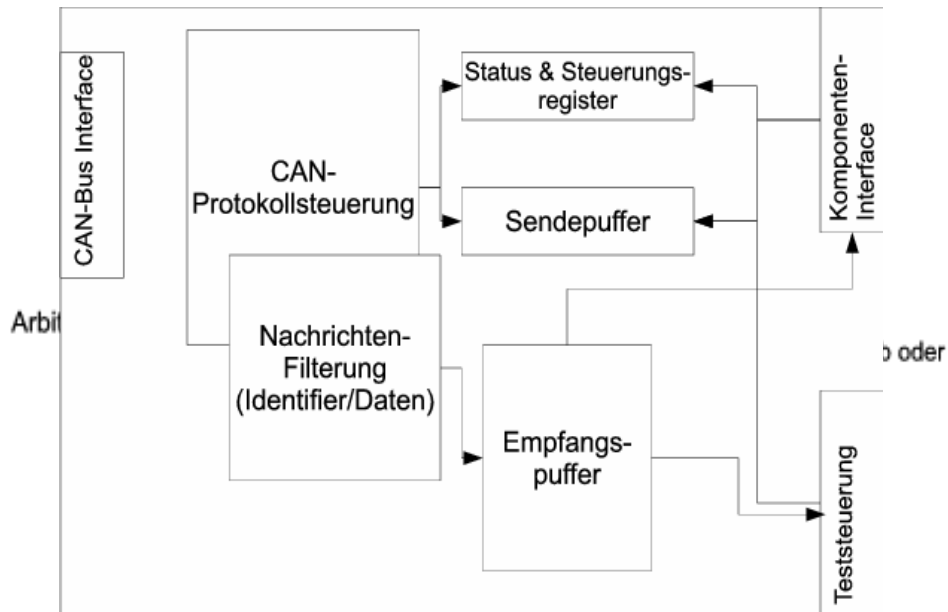


Abbildung IX.15: Position des Identifiers und Bitmusters für die Teststeuerung.

Abbildung IX.16: Verarbeitung von Nachrichten im CAN-Controller.

### IX.3.2.3. Interface Definition

Das im vorhergehenden Abschnitt beschriebene Testmodul kann über den CAN-Bus durch ein Diagnosegerät oder eine Diagnose-Software mit entsprechenden Befehlen angesprochen werden. Verschiedene Funktionalitäten sind im Rahmen der Fehlerdiagnose erforderlich:

- **Start Test:** Mit diesem Befehl können diverse Testprozeduren über die Teststeuerung gestartet werden. Nach Eintreffen des entsprechenden CAN-Bus Data Frames mit der Testanforderung wird unverzüglich der Test der einzelnen Elemente im CAN-Knoten nach den in der Nachricht angegebenen Vorgaben durchgeführt. Sobald der Test abgeschlossen ist, wird eine Statusnachricht generiert, die einen entsprechenden Status-Code, der über das Testergebnis

informiert, enthält. Diese Nachricht wird durch den CAN-Controller mit Hilfe des entsprechenden Identifiers über den CAN-Bus an das Diagnosegerät verschickt.

- Statusanforderung: Die Statusanforderung gibt Auskunft über den gegenwärtigen Zustand der Komponenten. Als Antwort werden entsprechende Status-Codes geschickt. Falls ein fehlerhaftes Bauelement vorhanden ist, werden zusätzliche Fehlerinformationen geliefert.
- Fehlerereignis: Wenn eine Komponente einen Fehler auf Halbleiterebene feststellt – beispielsweise durch regelmäßige Selbsttests im laufenden Betrieb – so soll diese auch aktiv Fehlermeldungen beispielsweise an eine spezielle andere, am CAN-Bus angeschlossene, Komponente schicken können. So besteht die Möglichkeit, auch sporadische oder nur unter bestimmten Bedingungen auftretende Elektronikfehler zu entdecken.

### IX.3.3. Aufbau des Demonstrators

Der prototypische Aufbau zur Demonstration der Fehlerdiagnose mit einer Kombination aus CAN-Bus und Scan-Path enthält mehrere verschiedene CAN-Knoten. Deren Eigenschaften können dabei über verschiedene Konfigurationsmöglichkeiten verändert werden. Sie unterscheiden sich in den Einsatzbereichen, denen sie im Demonstrator je nach ihrer Funktionalität zugeordnet sind. Die verschiedenen Knoten aus Abbildung IX.17 können dabei in drei Kategorien eingeteilt werden:

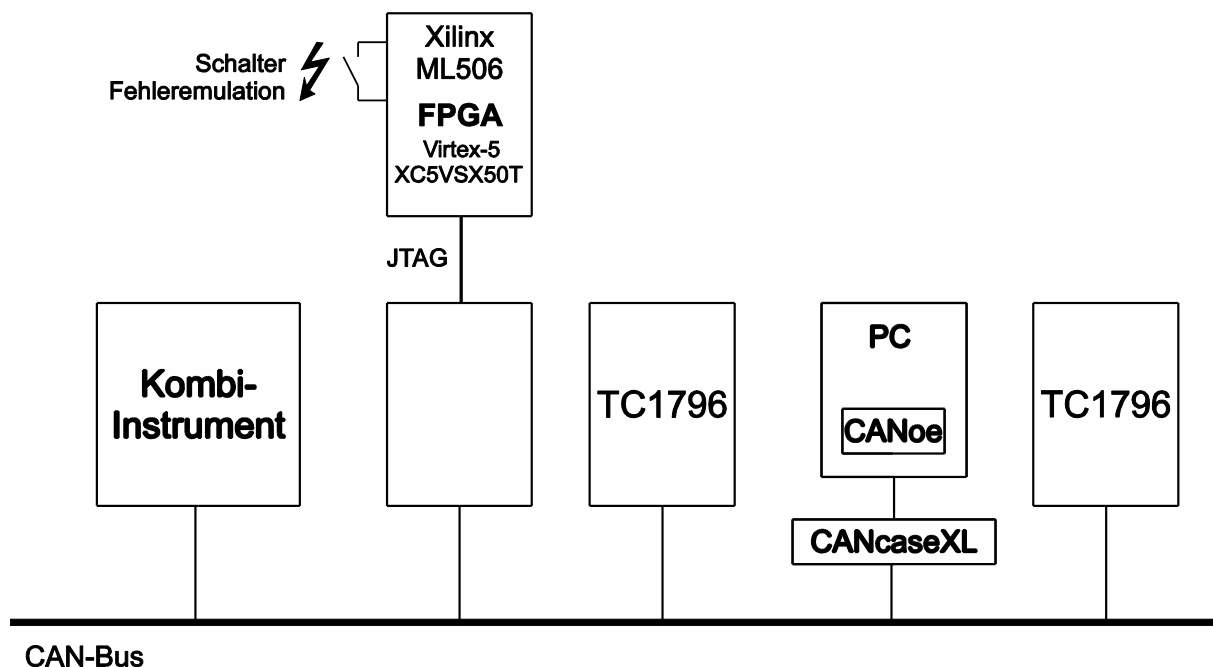


Abbildung IX.17: Erweiterter Aufbau des Demonstrators gegen Projektende.

- Konfigurierbare Komponenten: Die am CAN-Bus angeschlossenen Komponenten können flexibel konfiguriert werden. Die Möglichkeit, einerseits den fehlerfreien Betrieb wie auch andererseits das Auftreten von Störungen in einer Komponente zu emulieren, ist dabei gegeben. Diesen Teil des Demonstrators bilden zwei TriBoards TC1796 von Infineon Technologies AG sowie die Conti-Steuergeräteelektronik von Continental AG, die im nächsten Abschnitt beschrieben wird. Im integrierten Speicher können verschiedene Programme, beispielsweise zur Betriebssimulation und Fehlerprüfung, abgelegt werden.
- Analysekomponente: Von dieser Komponente ausgehend werden sowohl verschiedene Diagnosemethoden auf den konfigurierbaren Knoten ausgeführt, wie auch Fehlermeldungen von diesen Knoten empfangen. Die Analysekomponente stellt im Falle des hier spezifizierten Demonstrators ein Computer bereit, auf dem die Software CANoe der Firma Vector Informatik GmbH installiert ist. Mit Hilfe der Software können verschiedene Diagnoseverfahren ausgeführt werden.
- Demonstrationskomponente: Der Demonstrationsteil dient dazu, Teile der Funktionalität auch durch die Ausgabe von Daten über Komponenten, wie sie in handelsüblichen Fahrzeugen verwendet werden, zu veranschaulichen. Die meisten modernen Fahrzeuge enthalten mittlerweile entsprechende Anzeigeeinstrumente, welche eine Vielzahl von Informationen über den aktuellen Betriebszustand sowie Einstellungsmöglichkeiten des Fahrzeugs anzeigen können. Für den Demonstrator wird ein Kombinationsinstrument (Kombi-Instrument), wie es von unserem Projektpartner Audi eingesetzt wird, verwendet. Es verbindet verschiedene Messwert- und Informationsanzeigen miteinander.

#### **IX.3.4. Integration der Conti-Steuergeräteelektronik**

Zusätzlich zu den zuvor beschriebenen beiden TriBoards, dem Bus-Analyzer sowie des Kombi-Instruments wurde die Conti-Steuergeräteelektronik des Projektpartners Continental AG in den Demonstrationsaufbau eingebunden. Bei dieser handelt es sich um eine Testplattform auf Basis einer Getriebe-Steuereinheit, wie sie auch in Automobilen zum Einsatz kommt. Abbildung IX.18 visualisiert den erweiterten Aufbau des Demonstrators

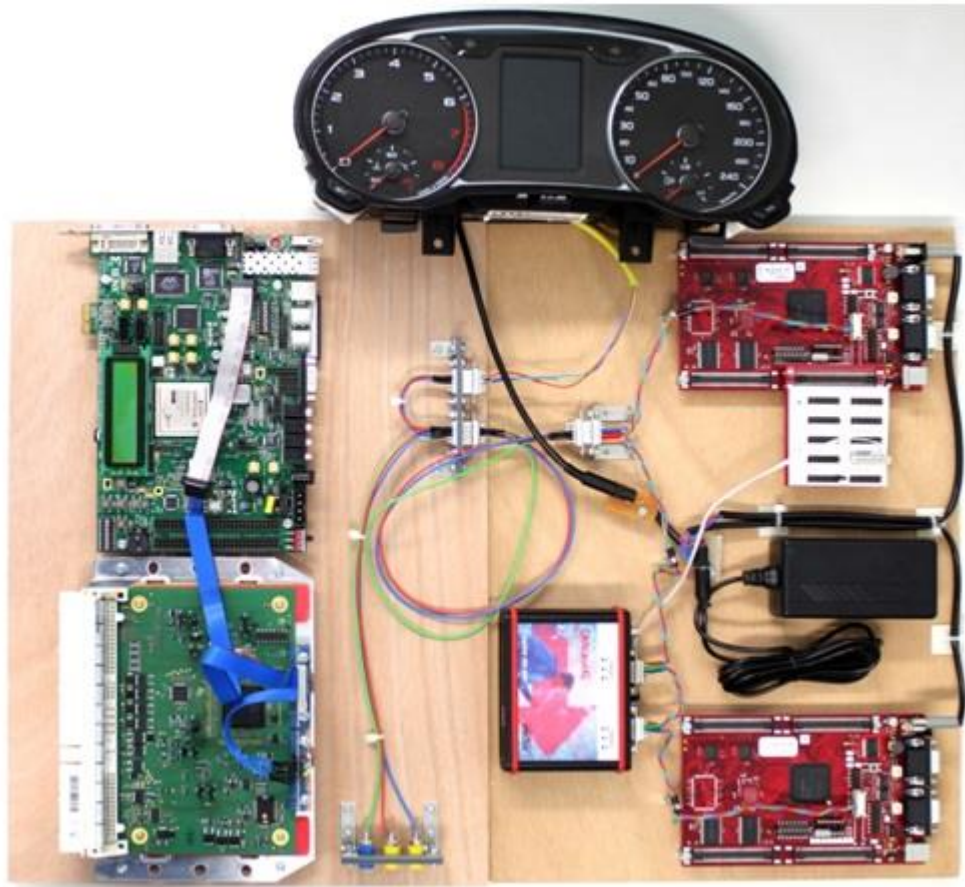


Abbildung IX.18: Prototypischer Aufbau des Demonstrators: Conti-Steuergeräteelektronik (links unten) und FPGA (darüber) wurden in den bestehenden Aufbau integriert.

Ebenso wie bei den TriBoard-Entwicklerkits wird ein Microcontroller der Infineon Technologies AG als zentrale Komponente verwendet. Zum Einsatz kommt dabei ein TC1797. Während die für die TriBoards TC1796 verwendeten Entwicklertools auch für die Conti-Steuergeräteelektronik verwendet werden können, muss bei der Programmierung auf einige Besonderheiten der einzelnen Prozessoren geachtet werden (3). Da jedoch bereits mit einem TriBoard TC1797 eine prototypische Implementierung des Fehleremulators entwickelt wurde (siehe Abschnitt IX.3.5.2), mussten nur kleine Veränderungen vorgenommen werden, um die Conti-Steuergeräteelektronik in den Demonstrator zu integrieren. Dabei handelte es sich vornehmlich um Anpassungen bei der Ansteuerung externer Komponenten, da die Pinbelegung der Conti-Steuergeräteelektronik sich von der Pinbelegung der TriBoards unterscheidet.

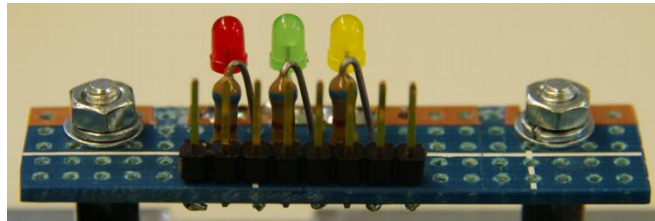


Abbildung IX.19: LED-Leiste als Erweiterung der Conti-Steuergeräteelektronik zur Visualisierung interner Zustände und Abläufe (4).

Da typische Aufgaben eines solchen Steuergeräts wie Motor- oder Getriebesteuerung sich im Demonstrator schlecht darstellen lassen, wurde die Conti-Steuergeräteelektronik um eine LED-Leiste erweitert, sodass interne Abläufe besser visualisiert werden können (siehe Abbildung IX.20). Des Weiteren mussten einige Kontakte nach außen gelegt werden, die für die Anbindung des FPGA benötigt wurden. Abbildung IX.20 zeigt die dafür benutzten Testkontakte. Konkret wurden dabei die Testpins P060 bis P067 – entsprechend den logischen Ports 2.8 bis 2.15 – benutzt, da diese noch nicht funktional gebunden waren.

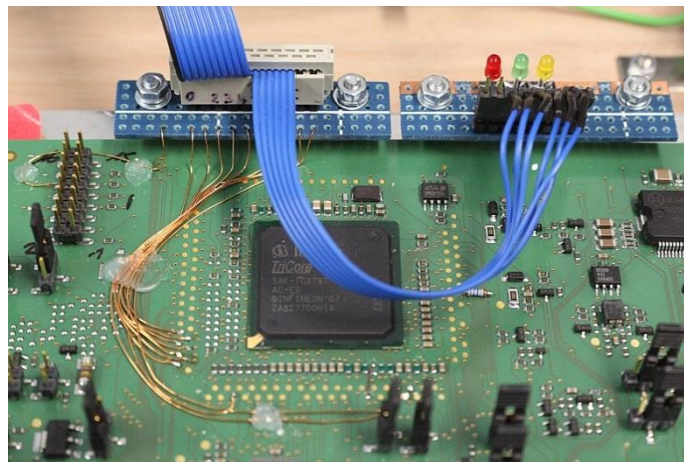


Abbildung IX.20: Erweiterung der Conti-Steuergeräteelektronik durch eine Steckverbindung nach außen und die in Abbildung IX.19 dargestellte LED-Leiste. Um den TriCore herum können deutlich die hierfür genutzten Testkontakte gesehen werden.

### IX.3.5. Fehleremulator

Zur Demonstration der Fehlerdiagnosemethode besteht auch die Möglichkeit, einen Fehler in einem Halbleiterelement wie beispielsweise einem ASIC zu emulieren. Dabei wird die Fehleremulation auf einer Hardwarekomponente realisiert. Dies gewährleistet einen realistischen Test wie er auch mit anderen, in Fahrzeugen verwendeten, Komponenten möglich wäre. Aufgrund dieser Anforderungen ist es notwendig, die fehlerbehaftete Komponente als Knoten in den CAN-Bus zu integrieren. Im Falle des hier spezifizierten Demonstrators kann dabei die schon vorhandene Hardware genutzt werden. Die verwendeten TriCores von Infineon besitzen beispielsweise eine Vielzahl von Interfaces zum Anschluss weiterer Komponenten wie zum Beispiel einem FPGA. Auf diesen Chip kann dann über die JTAG-Schnittstelle zugegriffen werden. Dadurch ist es möglich, das FPGA so zu programmieren, dass mit Hilfe eines Schaltmechanismus zwischen einem fehlerfreien und einem fehlerbehafteten Betriebsmodus jederzeit gewechselt werden kann. Über den TriCore ist die Kommunikation mit dem CAN-Bus möglich, so dass eine Fehlerdiagnose und -analyse durch Nachrichtenaustausch möglich ist.

Abbildung IX.18 veranschaulicht den Anschluss der Hardwarekomponente zur Fehlersimulation an ein Gerät mit einem TriCore. Durch diese generische Herangehensweise können beliebige Fehler in einer Hardwarekomponente emuliert und diagnostiziert werden. Die Notwendigkeit, feststehende Fehlerkategorien und Fehlercodes im Voraus zu definieren entfällt dadurch. Der Testaufbau kann je nach Prüfanforderung zeitnah und flexibel an die speziell zu untersuchenden Fehlfunktionen angepasst werden.

#### IX.3.5.1. Fehlererkennung

Es wurden verschiedene Tests zur Erweiterung des Demonstrators in Hinsicht auf die Fehlererkennung in externen Komponenten durchgeführt. Dabei wurden zum einen das Fehlverhalten von analogen Komponenten und zum anderen auch das Auftreten von Fehlern bei digitalen Schaltungen betrachtet.

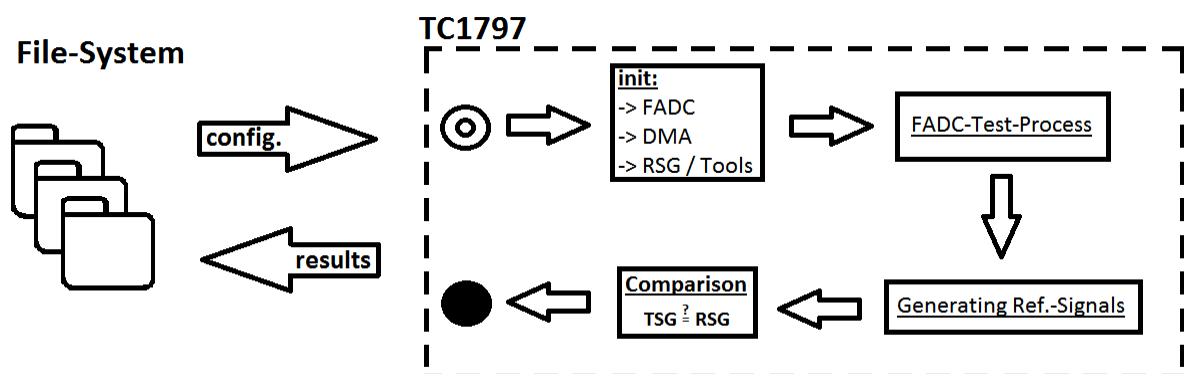


Abbildung IX.21: Diagnose des im TriCore integrierten FADCs über den CAN-Bus am Beispiel des TC1797 (5).

Für die erste Testkategorie wurde die Prüfung von Analog-Digital-Umsetzern (ADU, engl. ADC) ausgewählt, da der verwendete TriCore über mehrere dieser Umsetzer

verfügt (6). Abbildung IX.21 zeigt das Konzept für die Prüfung des Flash-ADCs (FADC) des TriCores (5) (7). Um über ein entsprechendes Prüfsignal zu verfügen, muss ein Test-Signal-Generator (TSG) am FADC des TriCores angeschlossen werden. Die ausgegebenen Werte werden anschließend mit denen eines Reference-Signal-Generators (RSG) verglichen. Dadurch können verschiedene Fehler wie beispielsweise Missing Code, bei dem Lücken im Wertebereich auftreten, oder Offsets der Werte erkannt werden. Die Steuerung des Diagnoseverfahrens und Rückmeldung der Ergebnisse erfolgt über den CAN-Bus. Nach demselben Prinzip ist es ebenfalls möglich, angeschlossene analoge Komponenten, zu prüfen. Zur Diagnose von Fehlern digitaler Schaltungselemente wurde das bereits erläuterte Konzept des Fehleremulators verwendet.

### IX.3.5.2. Umsetzung

Zur Fehleremulation wurde ein Virtex-5-ML506-Board der Firma Xilinx an das zuvor beschriebene TriBoard über eine JTAG-Schnittstelle angebunden. Das FPGA übernimmt in diesem Fall die Rolle einer angebotenen externen Komponente, wie beispielsweise eines ASICs. Durch die freie Rekonfigurierbarkeit lässt sich das Verhalten im Gegensatz zu beliebigen (komplexen) Komponenten exakt spezifizieren und dementsprechend Fehler genau diagnostizieren. Das FPGA-Board besitzt eine JTAG-Schnittstelle, die zu Konfigurations- und Diagnosezwecken verwendet wird. Zwar besitzt auch der TriCore Microcontroller eine JTAG-Schnittstelle, jedoch wird diese nur für den On-Chip-Debugging-Support verwendet (8) und kann daher nicht für beliebige Kommunikation mit anderen Elementen wie dem FPGA genutzt werden. Aus diesem Grund wurde eine Software-Implementierung der JTAG-Schnittstelle erstellt, die alle notwendigen Teilaspekte umfasst, um erfolgreich mit dem FPGA kommunizieren zu können.

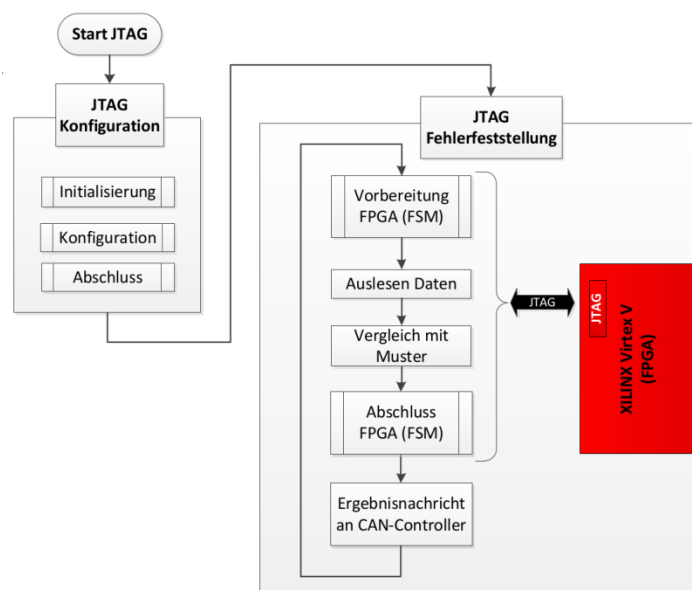


Abbildung IX.22: Ablauf des Testverfahrens am Beispiel des FPGA (9).



### IX.3.5.3. Realisierung der Testinfrastruktur

Mit Hilfe der JTAG Implementierung wurde anschließend die im letzten Abschnitt beschriebene Testmethode umgesetzt. Abbildung IX.22 veranschaulicht den Ablauf eines Tests: Während des Systemstarts initialisiert der TriCore auf dem Steuergerät die JTAG Schnittstelle zum FPGA. Diese wird zur Diagnose oder zur Konfiguration des FPGA genutzt, nachdem eine entsprechende Aufforderung mit Nutzdaten von der Diagnosesoftware erhalten wurde. Initialisiert durch den Endanwender wird von CANoe aus die Testkonfiguration des FPGA über den TriCore angestoßen: Sobald das FPGA zur Konfiguration bereit ist, wird eine entsprechende Antwort an die Diagnosesoftware gesendet und daraufhin die Konfigurationsdatei übertragen. Sobald diese erfolgreich auf das FPGA übertragen wurde, führen FPGA und Steuergerät die entsprechenden Testroutinen aus, wobei das Steuergerät die Ausgaben des FPGA mit eigenständig berechneten oder vorgegebenen Daten vergleicht. Als Beispielfunktion wurde die Berechnung von Pseudo-Zufallszahlen mittels Linear Feedback Shift Register (LFSR) implementiert (9). Diese können leicht vom Steuergerät berechnet werden, sodass sich die korrekte Funktion des FPGA überprüfen lässt. Durch einen Taster auf dem FPGA-Board kann die Berechnung gestört werden, indem der Eingang des LFSR – während der Taster gedrückt wird – auf high gelegt wird. In Abbildung IX.23 ist die vom FPGA berechnete Funktion auf Gatter-Ebene dargestellt. Das per Boundary Scan ausgelesene Ergebnis wird kontinuierlich mit den vorgegebenen beziehungsweise unabhängig berechneten Daten verglichen. Das Ergebnis des Vergleichs wird anschließend der Diagnosesoftware über eine CAN-Nachricht mitgeteilt.

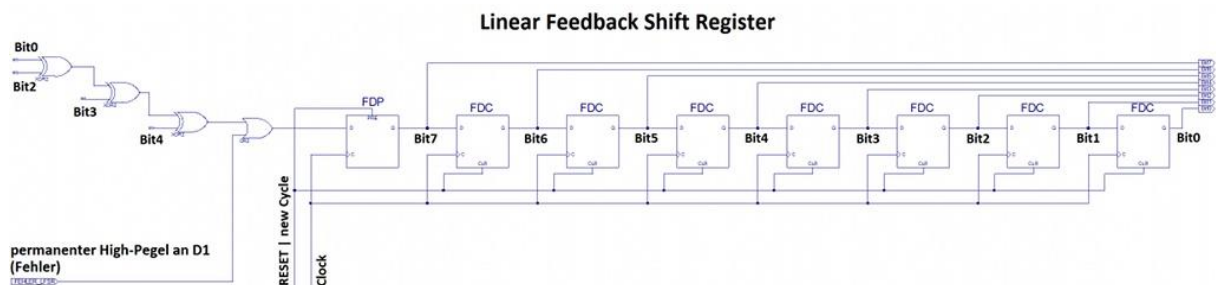


Abbildung IX.23: Linear Feedback Shift Register (LFSR) zur Erzeugung von Pseudo-Zufallszahlen entsprechend der Funktion  $\text{Bit7} = (\text{Bit0} \oplus \text{Bit2}) \oplus \text{Bit3} \oplus \text{Bit4}$  (9).

### IX.3.5.4. Diagnose mit CANoe

Der Demonstrator ist über das kommerzielle Multibus-Diagnosegerät CANcaseXL mit einem PC verbunden. Mit Hilfe der Software CANoe können damit CAN-Frames an einzelne Komponenten des Demonstrators gesendet und Antworten empfangen werden. Dabei können auch ganze CAN-Knoten durch CANoe simuliert werden (10). Zur Bedienung des Demonstrators und zur Visualisierung von Testergebnissen wurde in CANoe eine Oberfläche mit entsprechenden Anzeigemöglichkeiten für die Diagnose des FPGA erstellt (siehe Abbildung IX.24). Über die Oberfläche können Tests für die unterschiedlichen Knoten des Demonstrators gestartet werden – dabei sowohl einzeln als auch parallel:

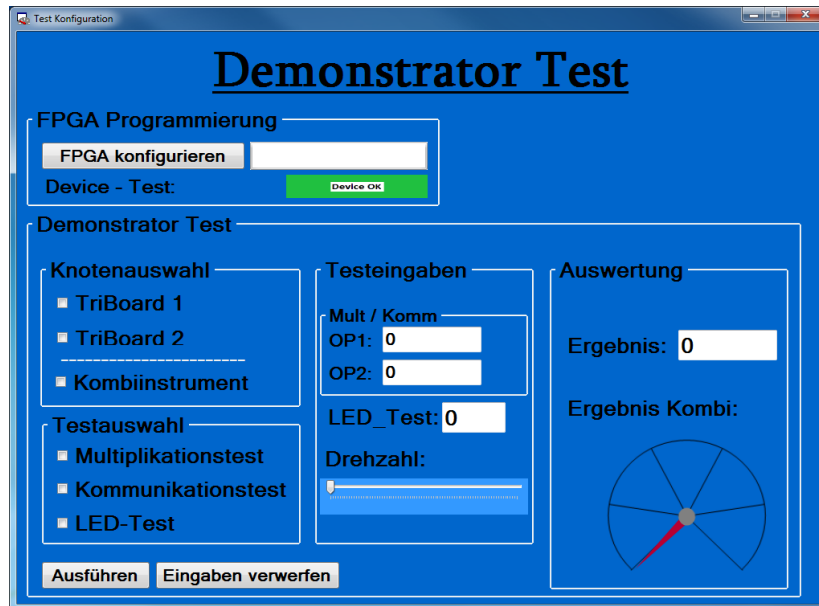


Abbildung IX.24: Grafische Oberfläche in CANoe zur Diagnose von Fehlern und Funktionsstörungen des Demonstrators (4).

- Device-Test (FPGA): Nach erfolgreicher Konfiguration des FPGA (siehe oben), wird dessen Ausgabe kontinuierlich überprüft und bei Abweichungen vom erwarteten Muster eine entsprechende Nachricht an die CANoe Oberfläche gesendet.
- Multiplikationstest: An das ausgewählte TriBoard werden die beiden vom Benutzer spezifizierten Faktoren gesendet. Der TriCore berechnet das Produkt und sendet das Ergebnis der Operation an CANoe zurück.
- Kommunikationstest: Hierbei wird analog zum Multiplikationstest vorgegangen, jedoch sendet der ausgewählte TriCore das Ergebnis nicht zurück an CANoe, sondern zum jeweils anderen TriBoard zur Überprüfung. Dieses wiederum vergleicht das derart erhaltene Ergebnis mit einer eigenen Berechnung und sendet das Ergebnis des Vergleichs an CANoe.
- LED-Test: Die vom Benutzer vorgegebene Dezimalzahl wird an die ausgewählten TriBoards gesendet und die Binärdarstellung der Zahl über die LEDs der TriBoards ausgegeben.
- Drehzahl: Vom Benutzer kann des Weiteren eine Drehzahl vorgegeben werden, die dann sowohl auf der entsprechenden Anzeige des Kombi-Instruments als auch grafisch in der CANoe Oberfläche dargestellt wird.

Nach Abschluss der jeweiligen Testroutinen auf der Hardware wird dem Benutzer eine entsprechende Rückmeldung über die Oberfläche gegeben. Das Ergebnis des Kommunikations- beziehungsweise Multiplikationstests kann auch wahlweise über die Drehzahl-Anzeige des Kombi-Instruments ausgegeben werden. Dabei erfolgt die Ausgabe jedoch nicht zyklisch, sodass auf der Anzeige des Kombi-Instruments nur ein kurzer Ausschlag zu beobachten ist.

Die gewonnenen Diagnosedaten werden im aktuellen Aufbau des Demonstrators grafisch angezeigt. Eine Umsetzung in ein Textformat für Handheld-Diagnosegeräte zur Verwendung in KFZ-Werkstätten ist leicht realisierbar.

### **IX.3.6. TC1796-Bootloader zur Teststeuerung**

Der Bootloader, der prototypisch für den TC1796 entwickelt wurde, dient der Steuerung von Tests auf dem TriCore ohne dessen eigentliche Funktion zu beeinflussen. Gesendet von der Diagnosekomponente, empfängt der Bootloader Testroutinen, initiiert deren Ausführung und sendet anschließend die entsprechenden Ergebnisse zurück. Zunächst werden wir auf die Anforderungen an einen solchen Bootloader eingehen, anschließend das entsprechend entwickelte Protokoll erläutern und schließlich auf die konkrete Umsetzung für den TriCore eingehen.

#### **IX.3.6.1. Anforderungen**

Um eine flexible Testumgebung zur Prüfung von Halbleiterelementen bereitzustellen, muss der für den TriCore TC1796 konzipierte Bootloader verschiedenen Anforderungen genügen (11):

- Nutzung des CAN-Bus: Der Bootloader soll es ermöglichen, sowohl den Code für Prüfroutinen wie auch Testvektoren je nach Diagnoseanforderungen über den CAN-Bus auf den TriCore zu übertragen. Aufgrund dieser Rahmenanforderung und um Kompatibilität mit den in der Industrie eingesetzten Systemen zu gewährleisten, müssen alle für die Testausführung nötigen Daten in CAN-Frames über das Bussystem übertragen werden.
- Verwendung eines Protokolls zur Teststeuerung: Um eine beliebige Anpassung von Testroutinen und -daten zu gewährleisten, ist ein Protokoll erforderlich, das den Ablauf der Datenübertragung regelt. Dabei müssen die verschiedenen möglichen Phasen wie die Bereitstellung der Prüfroutine sowie von entsprechenden Diagnosedaten bis hin zur Testausführung und Übermittlung der Resultate berücksichtigt werden.
- Anpassbare Testroutine: Mit dem Bootloader soll es möglich sein, neben der Testroutine selbst in einem separaten Schritt auch passende Testvektoren für die Prüfung von Bauelementen über den CAN-Bus auf den TriCore zu senden. Extern generierte Prüfmethode sollen dabei lediglich den Vorgaben des Kommunikationsprotokolls genügen müssen. Die Ergebnisse der Diagnoseroutinen sollen über den CAN-Bus an einen Diagnoseknoten übertragen und gespeichert werden können. Darüberhinaus muss die Testroutine unabhängig von den im Normalbetrieb erfüllten Aufgaben des TC1796 ausgeführt werden können, beispielsweise nur im Rahmen der Wartung in einer Kfz-Werkstatt. Ansonsten darf sie keinen Einfluss auf das Verhalten des Chips haben.
- Fehlererkennung: Neben einer Protokollierung des Kommunikationsverkehrs erfolgt auch eine Signalisierung von Übertragungsfehlern mit Hilfe von Error-Frames, wie sie in der Spezifikation des CAN-Bus-Standards festgelegt sind (12). Falls während der Ausführung von Tests oder dem Senden von Testdaten Fehler auftreten, werden diese detektiert und protokolliert. Der Verlust von CAN-Messages bei mehreren am Bus angeschlossenen Knoten muss erkannt werden.

Diese Anforderungen stellen die Grundlage für die Entwicklung eines entsprechenden Kommunikationsprotokolls, welches sowohl den Datenverkehr auf dem Bus, wie auch die Abläufe der Prüfroutinen regelt, dar. Die Protokolldetails werden im nächsten Abschnitt vorgestellt.

### IX.3.6.2. Bootloader-Protokoll

Das Bootloader-Protokoll regelt den Ablauf der Kommunikation zwischen einem Diagnosegerät und einem am CAN-Bus angeschlossenen, zu diagnostizierenden Knoten. Dies ist erforderlich, um eine gewünschte Testroutine auf den CAN-Knoten zu übertragen sowie Prüfdaten zu übermitteln. Anschließend müssen die angefallenen Testresultate ebenfalls wieder über den CAN-Bus an den Diagnoseknoten zurückgesendet werden. Daneben sind weitere Maßnahmen zur Gewährleistung eines fehlerfreien Ablaufs nötig (11).

- Protokollablauf: Um einerseits die Trennung von Testcode und Testdaten sowie andererseits auch weitergehende Funktionalität wie die Fehlerdetektion zu gewährleisten, müssen die entsprechenden Funktionen durch Abläufe im Bootloader-Prozess realisiert werden. In Abbildung IX.25 sind die verschiedenen notwendigen Phasen und ihre Einbettung in die Bootloader-Ausführung dargestellt. Nach einer Initialisierungsphase für den Verbindungsaufbau (INIT\_PHASE) folgt eine Datenübertragungsphase (DATA\_PHASE). Die Ausführung (APP\_PHASE) sowie der Ergebnistransfer (RESULT\_PHASE) werden mit einem Verbindungsabbau in der CLOSE\_PHASE abgeschlossen. Die Fehlererkennung läuft parallel zur Erkennung von Übertragungsproblemen während des Gesamtablaufes.

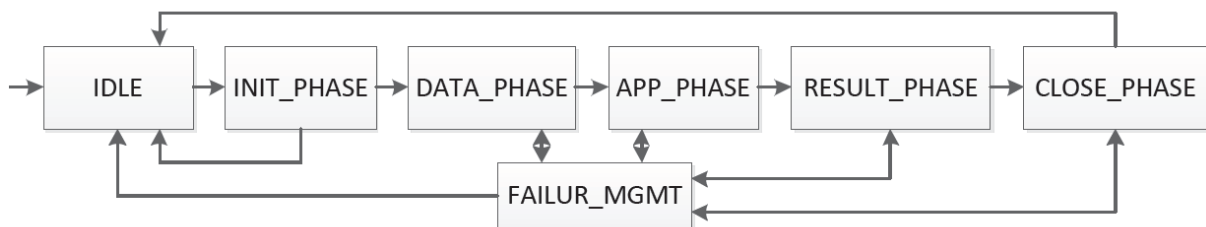


Abbildung IX.25: : Überblick über die Abfolge der einzelnen Phasen des Bootloader-Protokolls (11).

- Fehlerverhalten: Zur Fehlerbehandlung werden neben den schon durch das CAN-Protokoll festgelegten Verfahren noch weitere Methoden eingesetzt, um zusätzlich zur Datenintegrität auch eine korrekte Reihenfolge der übertragenen Nachrichten zu gewährleisten. Darüberhinaus muss der Verlust von Nachrichten erkannt werden können.
- Fehlerzähler: Für jede Verbindung wird ein Fehlerzähler eingesetzt, der bei Auftreten eines Fehlers und mit Empfang eines Fehler-Frames erhöht wird, bis er durch eine erfolgreich abgeschlossene Verbindung wieder zurückgesetzt wird. Für den Fehlerzähler wird ein Schwellwert festgelegt (Standardeinstellung 64), welcher bei Überlauf dazu führt, dass die Verbindung beendet werden muss. Treten nicht behebbare Fehler auf, wird dadurch ebenfalls die Verbindung geschlossen.

- Absicherung gegen Nachrichtenverlust: Neben der Sicherung der Datenintegrität mittels CRC-Checks und Empfangsbestätigungen bietet das CAN-Protokoll aufgrund der seriellen Übertragungsweise auch eine Reihenfolgesicherung. Allerdings kann der Verlust einer Nachricht bei mehreren am Bus angeschlossenen Knoten nicht mehr genau zugeordnet werden. Dies ist beispielsweise der Fall, wenn der Empfang einer Nachricht nur von anderen CAN-Knoten und nicht vom eigentlich adressierten Knoten bestätigt wird. Deshalb wird in diesem Fall ein ERR\_FRM gesendet, um den Verlust der Nachricht anzuzeigen. In diesem Frame sind darüberhinaus weitere Informationen wie die gegenwärtige Bootloader-Phase sowie die Adresse des Empfangsknotens enthalten.
- Frames: Zur Umsetzung der verschiedenen Phasen von Verbindungsaufbau über Datenübertragung, Fehlermanagement bis hin zum Verbindungsabbau werden verschiedene Frame-Typen genutzt, welche über den CAN-Bus versendet werden. Insgesamt sind 18 verschiedene Frames definiert, die die obigen Anforderungen umsetzen (11).

### IX.3.6.3. Umsetzung des Bootloader-Konzeptes

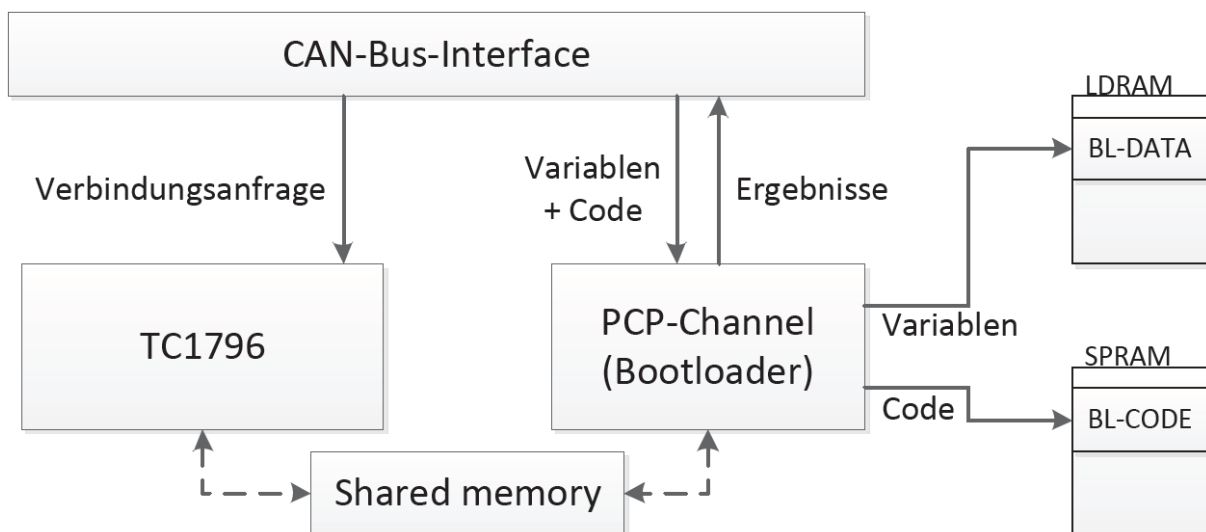


Abbildung IX.26: : Konzept für die Umsetzung des Bootladers auf dem TC1796 (11).

Eine der Vorgaben für die Funktionalität des Bootladers ist die Unabhängigkeit der Ausführung der Testroutine vom Laufzeitbetrieb und der dabei vom TC1796 erfüllten Aufgaben (11). Da die Nutzung des TriCore Bootstrap-Loaders einen Reset erfordern würde, wird stattdessen der Peripheral Control Processor (PCP) des TC1796 benutzt, um ankommende Bootloader-Befehle am CAN-Bus zu verarbeiten (6). Dies ist möglich, da der PCP über eine direkte Anbindung an das MultiCAN-Modul des TriCores verfügt. Bei eintreffenden CAN-Nachrichten kann ein Interrupt ausgelöst und somit eine entsprechende Verarbeitung von Befehlen ausgeführt werden. Mit Hilfe von EXIT-Instruktionen ist es dabei möglich, die Abarbeitung von Routinen durch den PCP jederzeit zu stoppen und anschließend wiederaufzunehmen. In Abbildung IX.26 sind die Datenströme für die Steuerung des Bootladers auf dem TriCore über den CAN-Bus dargestellt.

Nach dem Verbindungsaufbau über den TC1796 wird die Kontrolle des Bootloaders durch den PCP übernommen. Der Programmcode wird dabei im SPRAM abgelegt, während die Variablen in den LDRAM kopiert werden. Dabei darf die Größe der reservierten Speicherbereiche den normalen Laufzeitbetrieb nicht beeinträchtigen. Nach Abarbeitung der Testbefehle werden entsprechende Ergebnisse wieder über den CAN-Bus an den Diagnoseknoten gesendet. Da diverse TC1796-Module nur im privilegierten Modus konfiguriert werden können, muss bei der Programmierung des Knotens die Veränderung von Einstellungen beispielsweise für das Clocking von Modulen schon im Hinblick auf spätere Testroutinen berücksichtigt werden. Die Testroutine selbst wird in einem Code-Template untergebracht, welches einen reservierten Bereich für die Definition von nötigen Variablen vorsieht.

### **IX.3.7. Zusammenfassung und Ausblick**

Das in diesem Projektbeitrag konzipierte und beispielhaft implementierte Prüfkonzept erlaubt eine genauere Diagnose von Elektronikfehlern in Kraftfahrzeugen als dies bisher der Fall ist. Dabei können Fehler auf Ebene der Komponenten ohne physikalischen Eingriff diagnostiziert werden. Fehlerquellen können daher exakter identifiziert werden, um zielgerichtet Steuergeräte reparieren zu können und zukünftige Produkte zu verbessern. Durch den Einsatz von standardisierten Protokollen ist der vorgestellte Ansatz plattformunabhängig und erlaubt die Wiederverwendung von Hard- und Software, die bereits bei der Entwicklung von Komponenten zur Fehlersuche genutzt werden. Somit werden Mehrkosten minimiert und eine Kompatibilität mit gängigen Komponenten sichergestellt. Der prototypisch entwickelte Demonstrator veranschaulicht den Einsatz der Methode auf unterschiedlichen Komponenten im CAN-Bus Systemverbund: Einzelne oder mehrere Komponenten können über ein in der Praxis etabliertes Diagnosegerät hinsichtlich Fehlfunktionen evaluiert werden. Dabei wird nicht nur die Funktion der zentralen Recheneinheit der Steuergeräte überprüft, sondern auch daran angeschlossener Peripherie, um eine möglichst genaue Fehlerlokalisierung zu ermöglichen.

Somit wurden die Anforderungen der Continental AG und Infineon Technologies AG in Zusammenarbeit mit beiden Projektpartnern vollständig erfüllt: Der spezifikationsgemäße Zustand von Steuergeräten kann vom Anwender ohne physikalische Eingriffe evaluiert werden. Dies wird durch den prototypisch entwickelten Demonstrator veranschaulicht, der mit der von Continental bereitgestellten Conti-Steuergeräteelektronik ein Gerät auf Basis einer kommerziell erhältlichen Steuereinheit beinhaltet. Des Weiteren wird durch die Nutzung etablierter Standards auf Hardware-, Software- und Protokollebene sichergestellt, dass das beschriebene Verfahren auch für komponentenbasierte Systeme mit CAN-Vernetzung in anderen Anwendungsdomänen adaptiert werden kann.

Auch eine Nutzung in Kombination mit anderen Bus-Systemen aus dem Automobilumfeld, wie LIN (Local Interconnect Network) oder FlexRay, ist mit dem vorgestellten Konzept möglich. Ebenso könnten für andere in Steuergeräten verbaute diskrete und analoge Komponenten Testroutinen entwickelt werden, die auf den vorgestellten Konzepten des Fehleremulators (FGPA) und der Überprüfung von Analog-/Digital-Konvertern aufbauen. Weiterhin wünschenswert wäre eine

Möglichkeit zur Reparatur von Steuergeräten ohne diese ersetzen oder ausbauen zu müssen. Dazu könnte das in den letzten Meilensteinberichten erläuterte Konzept der Rekonfiguration eines FPGAs genutzt werden, sodass Signale auf andere oder Ersatzkomponenten umgeleitet werden und somit nicht ein Defekt in einer einzelnen Halbleiterkomponente zum Ausfall des gesamten Steuergeräts führt.

### **IX.3.8. Literaturverzeichnis**

- [1] Eijnden, Peter van den. JTAG ist mehr als nur ein Port. [http://www.epp-online.de/archiv/-/article/16537481/15904604/JTAG-ist-mehr-als-nur-ein-Port/art\\_co\\_INSTANCE\\_0000/maximized/](http://www.epp-online.de/archiv/-/article/16537481/15904604/JTAG-ist-mehr-als-nur-ein-Port/art_co_INSTANCE_0000/maximized/).
- [2] Pfeiffer, Olaf. Selecting a CAN Controller. <http://www.esacademy.com/en/library/technical-articles-and-documents/can-and-canopen/selecting-a-can-controller.html>.
- [3] Infineon Technologies AG. AP32187 Migration guide for TriCore™ based Microcontrollers. Oktober 2011.
- [4] Dentl, Dominik. CAN-Bus gestützte Fehlerdiagnose von integrierten Schaltkreisen in kommerziellen Getriebesteuerungsgeräten. Neubiberg : Universität der Bundeswehr München, Februar 2013.
- [5] Krüger, Andreas. Ausarbeitung zum Praxisprojekt: „Verification of ADC-BIST Test Concept“ bei Infineon Technologies AG. November 2011.
- [6] Infineon Technologies AG. TC1796 - User's Manual V2.0. Juli 2007.
- [7] Weber, Martin. Ausarbeitung zum Praxisprojekt: "Verification of ADC-BIST Test Concept" bei Infineon Technologies AG. November 2011.
- [8] Infineon Technologies AG. TC1797 - Datenblatt V1.1. April 2009.
- [9] Krüger, Andreas. JTAG-basierte Konfiguration und Fehlerdiagnose eines FPGAs für Automotive-Anwendungen. Neubiberg : Universität der Bundeswehr München, Juli 2012.
- [10] Vector Informatik GmbH. CANoe 8.0. November 2012.
- [11] Naprienko, Alexander. Entwicklung und Implementierung eines Bootloader-Programms für eine CAN-Schnittstelle im Rahmen einer erweiterten Diagnosefähigkeit. Neubiberg : Universität der Bundeswehr München, Juli 2011.
- [12] Robert Bosch GmbH. CAN Specification 2.0. 1991.

## X. Ergebnisse in Arbeitspaket 4

### X.1. Aufgabe 4.1 Datenerfassung und Diagnose in Automotive Systemen

Das folgende Kapitel gliedert sich in zwei Teile. Zuerst werden die Eigenschaften der bestehenden Testinfrastruktur im TriCore detailliert analysiert und eine entsprechende Methode für den zerstörungsfreien Test und die Diagnose des Chips vorgestellt. Der zweite Teil befasst sich mit der Modellierung nötiger Anpassungen der Testinfrastruktur, die die autonome Durchführung diagnostischer Tests ermöglichen. Das entwickelte Modell bildet die wichtigsten Eigenschaften des Testablaufs fürs Systemintegration ab.

Die Testinfrastruktur des Ziel-Microcontroller TriCore realisiert die bekannte EDT (Embedded Deterministic Test) Architektur [1], die ursprünglich für die Reduktion der Testdatenmenge bei der Herstellung konzipiert wurde, und ähnelt der STUMPS [2] Architektur für den eingebauten Selbsttest (BIST). In Arbeitspaket 4.1 wurde analysiert, dass nur geringe Änderungen in der Steuerung des Tests und einige zusätzliche Speicher erforderlich sind, damit diese Testinfrastruktur auch autonom Test und Diagnose im Feld unterstützen kann.

#### X.1.1. Wiederverwendung der bestehenden Testinfrastruktur

Der TC1797, der einen TriCore-Prozessor beinhaltet, verfügt über einen sogenannten Single Scan Chain Modus (SSCM). Hierbei werden alle Prüfpfade des Cores seriell kaskadiert, so dass sie wie ein einziger, langer Prüfpfad angesteuert werden können. Wie in Abbildung X.1 dargestellt kann auf diese Weise der aus der Chip-Herstellung bekannte Scantest im Feld wieder verwendet werden. Diese Vorgehensweise ermöglicht die Durchführung des Tests im verbauten Zustand. Voraussetzung hierfür ist der Zugang zur JTAG-Testschnittstelle der Zielchips.

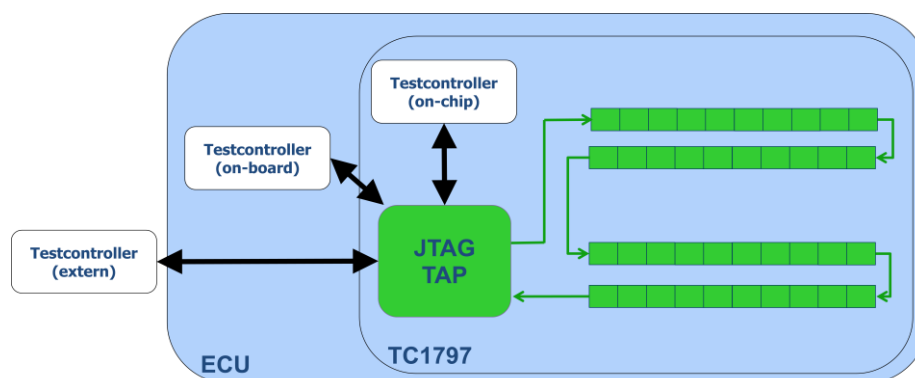


Abbildung X.1: Wiederverwendung der Testinfrastruktur des TC1797

Die Wiederverwendung des Herstellungstests für die Fehleranalyse im Automobilbereich wurde zum ersten Mal in [3] konzipiert und exemplarisch für das



Zielsystem VL381 von Continental am 16.4.2013 bei der Abschlussbegutachtung vorgeführt.

Eine Applikation der vorgeschlagenen Testmethode ist die Halbleiter-Analyse von Feldrückläufern, die beim IC-Hersteller stattfindet, sofern ein systematischer Defekt vermutet wird. Dieser Anwendungsfall stellt die höchsten diagnostischen Anforderungen, da er vorzugsweise im verbauten Zustand des Chips geschehen sollte, um die Fehlermechanismen nicht zu verfälschen und erfolglose Diagnoseversuche zu vermeiden. Das entwickelte Testverfahren ermöglicht die Wiederverwendung der für den Herstellungstest generierten Testmustermenge.

### **X.1.2. Modellierung der Testinfrastruktur für diagnostischen BIST**

Um eine optimale Nutzung der verfügbaren Ressourcen im System zu ermöglichen und eine ausreichende Fehlerabdeckung und Diagnoseauflösung zu erreichen, muss der Testinfrastruktur des Chips so modifiziert werden, dass damit die notwendige Flexibilität für die Testplanung erlaubt wird. Die notwendigen Ergänzungen der STUMPS-Architektur für die autonome Erfassung der diagnostischen Daten wurden in [4] [5] und [6] veröffentlicht, und ein entsprechender diagnostischer Algorithmus in [7] vorgestellt. Dazu müsste die Testinfrastruktur mit ausreichendem Detail modelliert werden, so dass zwei verschiedene Zwecke verfolgt werden konnten: auf der einen Seite sollte die getestete Komponente so beschrieben werden, dass eine realistische Abschätzung der erreichbaren Diagnoseabdeckung und Hardwarekosten auf Chip-Ebene ermittelt wird, auf der anderen Seite sollten die relevanten Systemanforderungen wie z.B. Testdauer und Ressourcenauslastung dargestellt werden.

Für die Modellierung der Teststrategie und ihre Verknüpfung zur Applikationsschicht des gesamten Systems wurde ein Transaktionsebenenmodell eingesetzt (Transaction Level Models, TLMs). Die kommunikationsbezogene Sichtweise von TLMs ist auch zur Modellierung eines Testverfahrens geeignet, da beim Test große Datenmengen ausgetauscht werden müssen (siehe Abbildung X.2). Die Modellierung der Testapplikation umfasst die Übertragung der Testdaten zum Chip, die Durchführung der Tests sowie die Sammlung der Ergebnis- und Diagnosedaten.

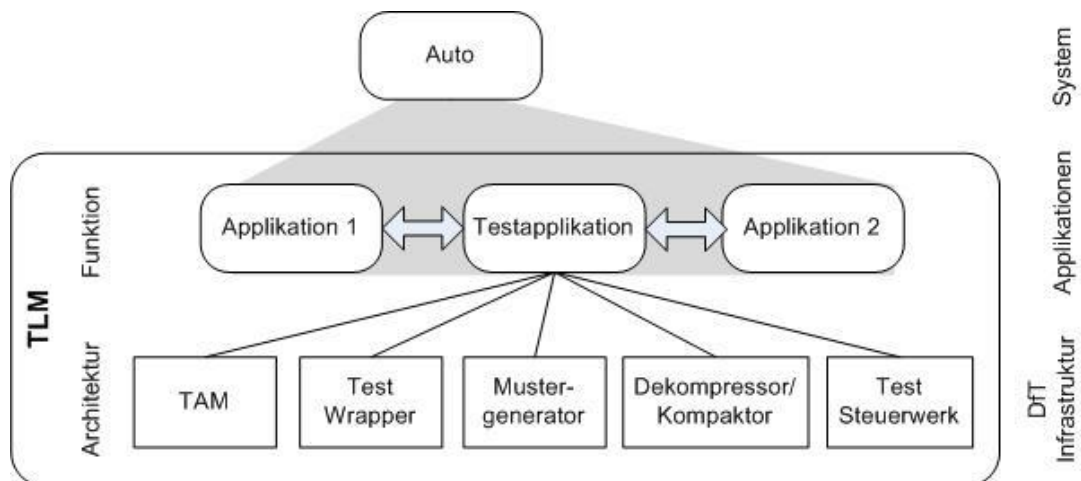


Abbildung X.2: Modellierung einer Testapplikation

Die vorgeschlagenen Test- und Diagnoseapplikationen auf Systemebene verfolgen drei verschiedene Optimierungsziele: Fehlerabdeckung, Kosten und Testdauer. Das wichtigste Ziel ist zuerst eine hohe Fehlerabdeckung, jedoch können so die Kosten für die Testmustererzeugung drastisch wachsen. Mixed-Mode BIST [5] kann auch dazu verwendet werden, dass die Speicherbedarf reduziert wird. Die Testdauer kann hierbei zunehmen, da eine größere Menge an Zufallsmuster angelegt werden muss.

Um das Zusammenspiel dieser Aspekte genauer zu bestimmen, werden unterschiedliche Profile mit diversen Eigenschaften erzeugt. Ein Profil beinhaltet die Kosten und Testdauer von einem BIST-Verfahren mit fester Fehlerabdeckung und einer vorgegebenen Anzahl von Zufallsmuster. Dazu wurde der TriCore mit vollständigem Prüfpfad synthetisiert und mit den notwendigen Teststrukturen ausgestattet.

### X.1.3. Integration von Diagnosefunktionalität im Gesamtsystem

Die Modellierung des Test- und Diagnoseverfahrens ergänzt das bestehende funktionale Modell der Funktionalität eines Fahrzeugs und unterstützt Entwurfs- und Analysemethoden, um strukturelle Tests durchzuführen und Diagnosedaten im Gesamtsystem zu sammeln.

Die entwickelten Modelle und Methoden ermöglichen die Integration erweiterter Diagnosefunktionen in das Gesamtsystem Automobil unter Ausnutzung von Phasen der Inaktivität von Steuergeräten im Betrieb und die Analyse bestehender Trade-Offs bei der Übertragung von Diagnosedaten (Detaillierungsgrad, Komprimierung). Dieser Ansatz wurde in [8] im Detail vorgestellt. Er nutzt gemeinsame Feldbusse für die Übertragung der Test- und Diagnosedaten aus und ermöglicht die Verwendung bereits reservierter Kommunikationsstrukturen für den Zweck der Diagnose.

Die Untersuchung der BIST-Profile bei der entwickelten Methode für Entwurfsraumexploration (engl. Design Space Exploration) legt in diesem Beispiel 176 optimale Implementierung fest, die verschiedene Kompromisse bezüglich Testqualität, Kosten und Nachlaufzeit darstellen (siehe Abbildung X.3). In Hinsicht auf Kosten ist die Methode in der Lage, Implementierungen mit weniger als 1%

Mehrkostenbedarf zu ermitteln. Solche Implementierungen erreichen eine Testabdeckung von circa 80 % der Haftfehler.

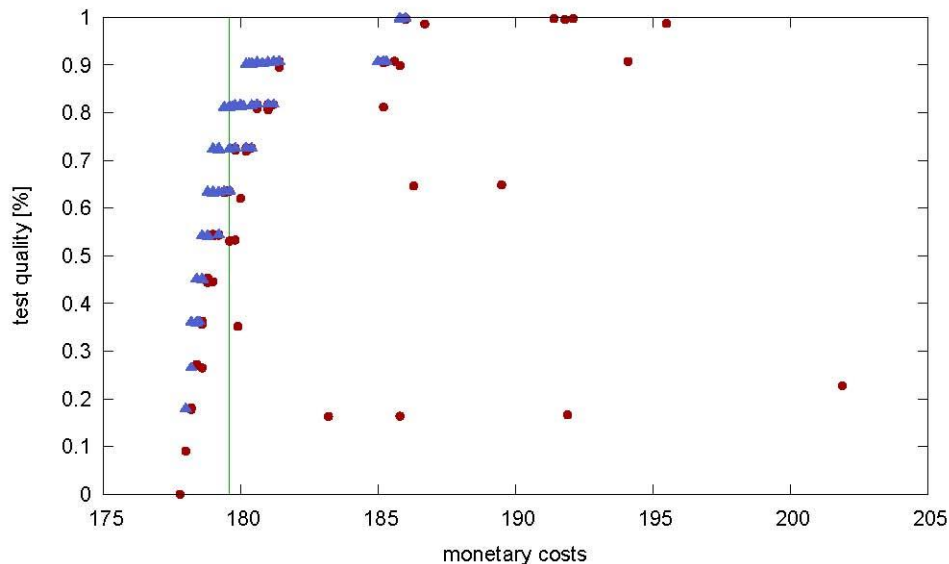


Abbildung X.3: 176 Implementierungen mit verschiedenen Eigenschaften bezüglich Kosten und Testqualität. Implementierungen mit Nachlaufzeit bis zu 5 Sekunden sind mit rot markiert. Implementierungen mit größeren Nachlaufzeit sind mit blau markiert

#### X.1.4. Literatur/Referenzen

- [1] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee "Embedded deterministic test," In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. Vol. 23, no.5, pp. 776- 792, May 2004.
- [2] P. H. Bardell, W. H. McAnney. "Self-Testing of Multichip Logic Modules," in Proc. IEEE International Test Conference (ITC'82), 1982, pp. 200-2004
- [3] A. Cook, D. Ull, M. Elm, H.-J. Wunderlich, H. Randoll, S. Doehren "Reuse of Structural Volume Test Methods for In-System Testing of Automotive ASICs," in Proc. IEEE Asian Test Symposium (ATS'12) , 2012, pp. 214-219.
- [4] A. Cook, S. Hellebrand, T. Indlekofer, H.-J. Wunderlich "Diagnostic Test of Robust Circuits, " in Proc. IEEE Asian Test Symposium (ATS'11), 2011, pp. 285-290.
- [5] A. Cook, S. Hellebrand, and H.-J. Wunderlich, "Built-in Self-Diagnosis Exploiting Strong Windows in Mixed-Mode Test, " in Proc. IEEE European Test Symposium (ETS'12), Annecy, France, may 2012.
- [6] A. Cook, S. Hellebrand, M. E. Imhof, A. Mumtaz, and H.-J. Wunderlich, "Built-in Self-Diagnosis Targeting Arbitrary Defects with Partial Pseudo-Exhaustive Test," in Proc. Latin American Test Workshop (LATW'12), Quito, Ecuador, april 2012.
- [7] A. Cook, M. Elm, H. Wunderlich, and U. Abelein, "Structural In- Field Diagnosis for Random Logic Circuits," in Proc. IEEE European Test Symposium (ETS'11), Trondheim, Norway, may 2011, pp. 111 –116.

- [8] F. Reimann, M. Glaß, J. Teich, and U. Abelein, "Scenarienbasierte Integration von Diagnosefunktionalität in E/E Architekturen, " *GMM-Fachbericht-AmE 2013*.

## **X.2. Aufgabe 4.2 Methoden zur optimalen Diagnoseauflösung auf Systemebene**

Viele der in DIANA entwickelten Diagnosetechniken erfordern große Mengen an Testsignaturen. In dieser Aufgabe wurden Verfahren entwickelt, um diese Datenmengen im System vorhalten zu können und die Diagnosefunktionen mit den funktionalen Applikationen abzustimmen. Hierzu wurde zunächst ein ganzheitliches Systemmodell entwickelt, in der die Komponenten der E/E Architektur sowie die Applikationen des Normalbetriebs als auch der Diagnose und deren zeitliches Verhalten im Abstraktionslevel von Betriebsphasen modelliert werden können, siehe X.2.1.

Da insbesondere in der zur Diagnose wichtigen Startphase die verfügbare Zeit zur Durchführung von Diagnose stark eingeschränkt ist, wird diese Betriebsphase in X.2.2 genauer untersucht. Insbesondere die Frage nach der möglichen diagnostischen Auflösung und die daraus folgenden zeitlichen Auswirkungen der zusätzlichen Diagnose auf das Aufstartverhalten werden detailliert simuliert.

Die umfangreichen Datenmengen, die sowohl in der Offline-Diagnose zur Erkennung permanenter Fehler als auch während des Betriebs für transiente und intermittierende Fehler anfallen, erfordern den Einsatz von Datenkompressionstechniken, wie sie auch im Produktionstest verwendet werden. In X.2.3 werden entsprechend geeignete Kompaktierungsverfahren vorgestellt.

Die nun zur Verfügung stehenden Werkzeuge zur Modellierung, Evaluation und Kompaktierung von Diagnoseapplikationen werden in X.2.4 in eine ganzheitliche Optimierung integriert, um systemweit optimale Entwurfsentscheidungen zur Einbringung erweiterter Diagnose in ein Gesamtsystem zu ermöglichen.

### **X.2.1. Aufgabe 4.2.1 Modellierung des Gesamtsystems zur Integration erweiterter Diagnose unter Berücksichtigung von Betriebsmodi**

Um erweiterte Diagnosefunktionen in ein Gesamtsystem optimal integrieren zu können, ist Wissen über die E/E Architektur nötig. Der zur Modellierung entwickelte, graphenbasierte Ansatz wurde im Meilensteinbericht 4.2.1 eingeführt. Das Modell berücksichtigt entsprechend funktionale Applikationen, die E/E-Architektur, die Betriebsmodi und die unterschiedlichen Diagnosefunktionen selbst.

Die funktionalen Applikationen, siehe Abbildung , bestehen aus Einzelfunktionen (gelb), die über Nachrichten (blau) kommunizieren. In dem Modell sind Eigenschaften wie Nachrichtengröße, Ausführungszeit, Speicherbedarf, usw. hinterlegt.

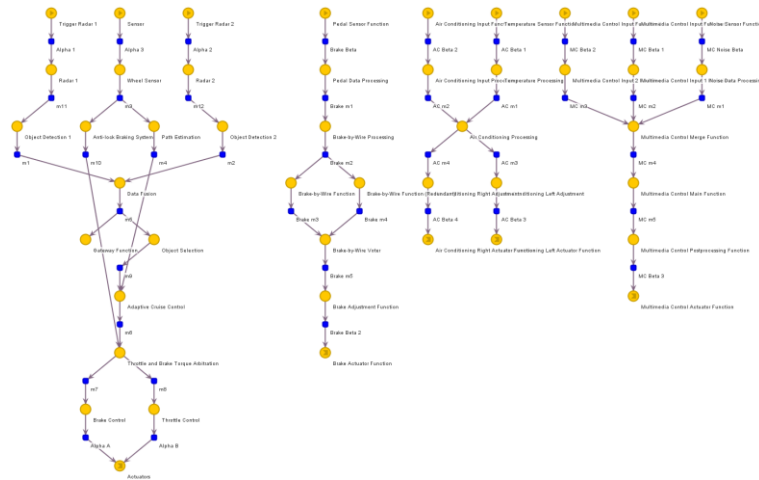


Abbildung X.4: Die vier Applikationen des exemplarischen Teilsystems. Die Funktionen (gelb) tauschen entsprechend ihrer Datenabhängigkeiten Nachrichten (blau) aus.

Für die Umsetzung steht ein Architekturtemplate aus Steuergeräten (rot), Sensoren und Aktuatoren (beide Typen braun) zur Verfügung, die mit Feldbussen (gelb) verbunden werden können, siehe Abbildung X.5. Stehen in Zukunft speziell bestückte Steuergeräte zur Verfügung, deren Prozessoren die in DIANA entwickelten Methoden zur Halbleiterdiagnose integrieren, können diese als zusätzliche Alternativen ergänzt werden und würden über ihre Parametrisierung – verbesserte Diagnosefähigkeit bei ggf. erhöhtem Energieverbrauch oder Kosten – zusätzliche Trade-Offs generieren.

Wie im Meilensteinbericht zu Aufgabe 4.1.4 erläutert, wurden auch die einzelnen Betriebsmodi, in denen sich die Applikationen befinden können, modelliert, vergleiche Abbildung X.6. Das Wissen über die Betriebsmodi wird im Weiteren genutzt, um Diagnose in das System einzuplanen, ohne dass der Normalbetrieb gestört wird.



In dieses Gesamtsystem sind nun die in DIANA entwickelten Funktionen für die erweiterte Diagnose einzuplanen. Näher betrachtet werden im Folgenden SBST oder BIST, die Methodik ist aber auch für andere entwickelte Diagnoseverfahren einsetzbar.

Im Folgenden kann für jedes Steuergerät gewählt werden, ob und welcher der Typ von erweiterter Diagnose (z.B. BIST oder SBST) auf dem Steuergerät implementiert werden soll. Hinzu kommt der Speicherort der Testpattern für die BISTs, vergleiche Abbildung X.7: Diese können entweder lokal am Steuergerät oder zentral am Gateway hinterlegt werden. In ersterem Fall kann der BIST direkt aktiviert werden, wenn das System in einen Betriebsmodus wechselt, in dem das Steuergerät nicht verwendet wird (Teilnetzbetrieb). Jedoch erzeugt der zusätzliche Speicher monetäre Kosten. Im zweiten Fall, der Speicherung der Testmuster auf dem Gateway, müssen die Muster zunächst übertragen werden, bevor der eigentliche BIST gestartet werden kann. Dies hat zur Folge, dass sich die Nachlaufphase sowie die Zeit, die das Steuergerät nicht in den Teilnetzbetrieb gelegt werden kann, verlängert. Außerdem erhöht eine längere Zeit zur Durchführung des Tests die Möglichkeit, dass das Steuergerät vorher wieder benötigt wird und in den normalen Betriebsmodus wechseln muss. Der Test müsste in diesem Fall abgebrochen werden und liefert keine Ergebnisse. Werden SBST eingesetzt, stellt sich zusätzlich die Frage mit welcher Periode sie ausgeführt werden. Je häufiger der SBST eingeplant werden kann, desto früher können Fehler bereits während des Betriebs festgestellt werden. Jedoch erzeugt dies zusätzlichen Energieverbrauch.

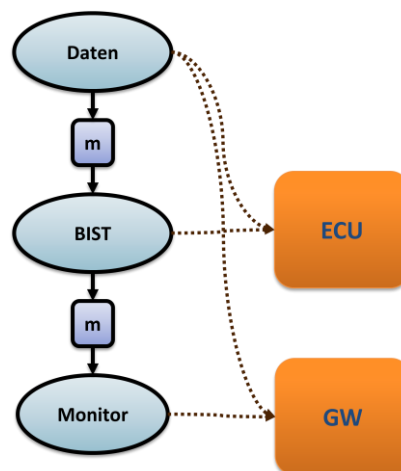


Abbildung X.7: Testmuster zentral oder verteilt speichern? Daten können z.B. auf ECU oder Gateway gespeichert werden. Bei ersterem müssen die Testmuster vor dem Start des Test zunächst übertragen werden.

Nach ihrer Fertigstellung verschicken sämtliche Diagnosefunktionen Nachrichten an das zentrale Gateway, die detaillierte Informationen über die Testergebnisse enthalten. Diese Nachrichten müssen, wie die Nachrichten zur Übertragung der Testmuster im Falle des Speicherns auf dem Gateway, in das System eingeplant werden. Abbildung zeigt die daraus resultierende Diagnoseapplikation des Gesamtsystems.



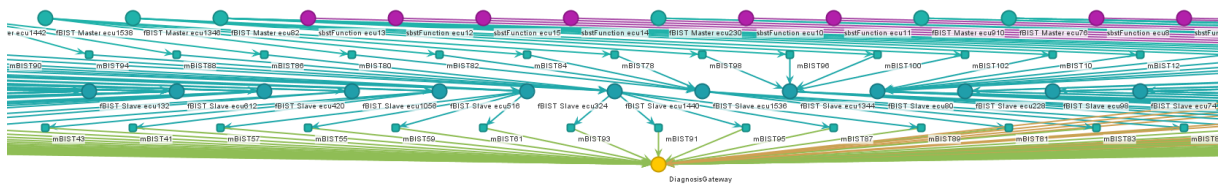


Abbildung X.8: Diagnosefunktionen. Die verfügbaren SBSTs (lila) senden Ergebnisvektoren direkt an eine Funktion zur globalen Auswertung und ggf. Speicherung auf dem zentralen Gateway (gelb). Die BISTs (türkis) sind jeweils durch eine Funktion, die den Speicherort der Testmuster anzeigt, sowie durch die eigentlichen Diagnosefunktionen modelliert. Je nach verwendetem Schema zur transparenten Datenübertragung stehen mehrere Nachrichten mit unterschiedlichen Sendecharakteristika zur Verfügung.

Wie in AP 4.1.4 entwickelt, werden die systemweiten Ablaufpläne an beide Modi der jeweiligen Steuergeräte (den des Normalbetriebs und den des Diagnosebetriebs) angepasst, siehe Abbildung X.9. Hierbei ergibt sich der Vorteil, dass für das Gesamtsystem nur ein Ablaufplan verwendet wird, wodurch die Absicherung des Gesamtsystems weniger komplex wird. Wird im Normalbetrieb jedoch beispielsweise eine Nachricht mit höherer Periode eingeplant, damit eine Diagnosenachricht besser eingeplant werden kann, führt dies zu einer Überreservierung und damit „Verschwendung“ von Bandbreite im Normalbetrieb. Daher ist hierfür eine globale Optimierung der Ablaufplanung und Defragmentierung notwendig. Ist dies möglich, führt diese Variante zum besten Ergebnis für das Gesamtsystem.

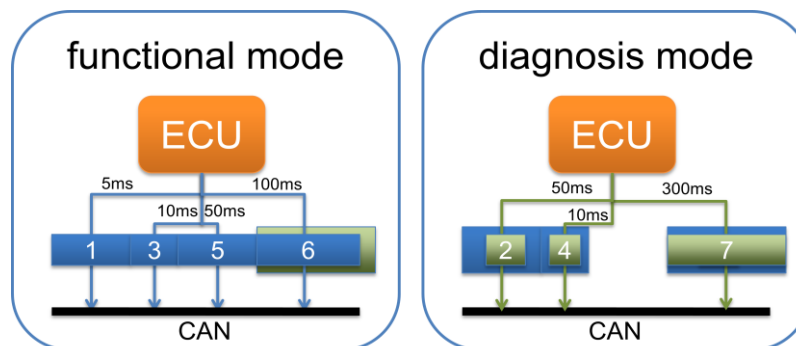


Abbildung X.9: Globale Optimierung unter Berücksichtigung aller Systemzustände

### X.2.2. Aufgabe 4.2.2 Verfahren zur hierarchischen Aktivierung von Diagnoseverfahren, zur graduellen Diagnose auf Systemebene und Zuverlässigkeitsbewertung

In dieser Aufgabe wurde detailliert untersucht, wie die Aktivierung einer erweiterten Diagnosefunktionalität im Kontext des Gesamtsystems bewertet werden kann. Hierzu wird die Aufstartphase eines Steuergeräts detailliert und modelliert. Dabei werden folgende Ziele verfolgt:

1. Identifikation des idealen Zeitpunkts während der Startphase, zu dem Halbleiter-Diagnosen auf den Steuergeräten ausgeführt werden können, siehe X.2.2.1.

2. Entwicklung eines Verfahrens zur Ermittlung der Zeit, die für Halbleiter-Diagnosen auf den Steuergeräten zur Verfügung steht, ohne vorgegebene Deadlines zu verletzen. Damit wird verhindert, dass die gesamte Startphase des Automobils unzulässig verzögert wird. Siehe X.2.2.2.

Durch den Nachweis der Machbarkeit derartiger Diagnosen während der Startphase, siehe X.2.2.3, ergibt sich für den Automobilhersteller die Möglichkeit, derartige Anforderungen in das Lastenheft für Steuergeräte mit aufzunehmen. Somit wird sichergestellt, dass Automobile der nächsten Generation bereits über derartige Diagnosen verfügen.

### **X.2.2.1. Zeitpunkte für Halbleiter-Diagnosen während der Startphase von Steuergeräten**

Für die Einplanung von Halbleiter-Diagnosen bieten sich grundsätzlich mehrere Zeitpunkte mit unterschiedlichen Vor- und Nachteilen an. Hierzu wurden in Absprache mit den Partnern geeignete Zeitpunkte zur Einplanung von erweiterter Diagnose in der Startphase modelliert. Mögliche Zeitpunkte zeigt Abbildung X.10: (a) Vor beziehungsweise zu Beginn der Hardwareinitialisierung, (b) zwischen der Hardware- und Software-Initialisierung, oder (c) nach der Software-Initialisierung, direkt vor dem Normalbetrieb (c). Jeder der drei genannten Möglichkeiten bietet Vorteile, bringt aber auch spezifische Nachteile mit sich, die im Meilensteinbericht zu AP 4.2.2 detailliert erläutert wurden.

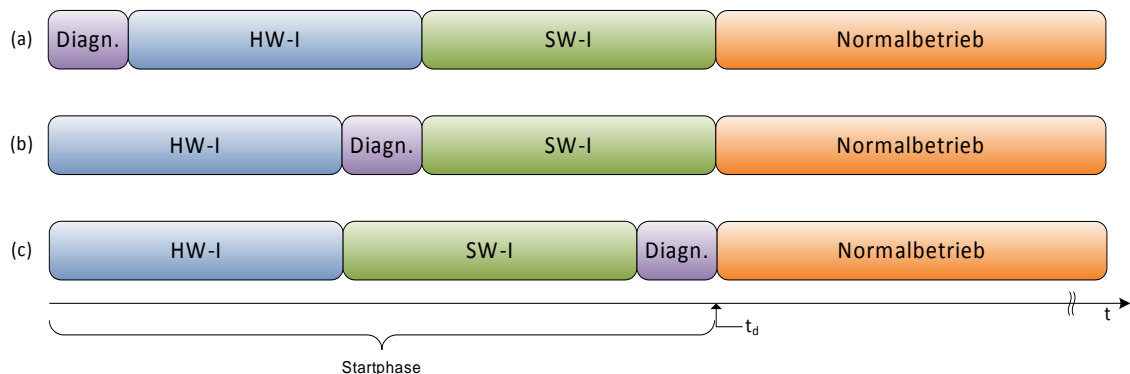


Abbildung X.10: Mögliche Startzeitpunkte für Diagnosen während der Startphase

### **X.2.2.2. Verfügbare Zeit für Halbleiter-Diagnosen während der Startphase von Steuergeräten**

Das Ausführen der Halbleiter-Diagnosen vor der eigentlichen Steuergeräteinitialisierung hat den Vorteil, dass keine Initialisierungsroutine des Steuergeräts doppelt ausgeführt werden muss, um den durch die Diagnosen zerstörten Systemzustand wieder herzustellen. Der Nachteil ist jedoch, dass die einzelnen Initialisierungsroutinen, besonders die der Hardware, zeitlichen Schwankungen unterliegen, was vom Projektpartner Continental bestätigt werden konnte. Plant man nun konservativ um das Einhalten der vorgegebenen Deadlines garantieren zu können, steht weniger Zeit für das Durchführen der Diagnosen zur Verfügung. Dementsprechend können nur weniger aufwändigere Tests durchgeführt

werden, was in einer niedrigeren Fehlerabdeckung resultiert. Aufwändigere Diagnosen bieten eine höhere Fehlerabdeckung, allerdings besteht hierbei das Risiko, dass Deadlines verletzt werden.

Um eine möglichst hohe diagnostische Auflösung erreichen zu können, muss die für die erweiterte Diagnose verfügbare Zeit möglichst genau bestimmt werden. Da bisherige Simulationsverfahren auf Systemebene zeitliche Schwankungen die während der Startphase auftreten können nicht hinreichend berücksichtigen, wurde eine entsprechende Simulation entwickelt und anhand der Startphase des Steuergeräts der Applikation „CVT Getriebe“ evaluiert. Dieses Steuergerät wird auch im Demonstrator, der für das DIANA-Projekt angefertigt wird, verwendet. Die Ausführungszeiten wurden von Continental zur Verfügung gestellt. Die Schwierigkeit bei der Modellierung bestand darin, dass diese Ausführungszeiten teils stark schwanken.

Um zu bestimmen, mit welcher Wahrscheinlichkeit das Ausführen eines Tests dazu führt, dass Deadlines überschritten werden, wurde ein Modell für die Startphase der Applikation „CVT Getriebe“ erstellt. Als Grundlage für das Modell dient die Aktororientierte Systemmodellierungssprache *SystemMoC* und ein auf dem Konzept von *Virtual Processing Components (VPC)* basierender Discrete-Event-Simulator. *SystemMoC* erweitert die Sprache *SystemC* um ein Model-of-Computation. *VPC* erlaubt eine effiziente Performance-Simulation komplexer Steuergerätearchitekturen auf verschiedenen Abstraktionsebenen. Dieses Modell berücksichtigt darüber hinaus die Zeitvarianzen der einzelnen Startphasen eines Steuergeräts, die aufgrund äußerer Einflüsse auftreten können.

Abbildung X.11 zeigt den Aufbau des erstellten Modells der Startphase eines einzelnen Steuergeräts. Jeder Kasten steht für einen Akteur, an den einzelne Performanceparameter, wie beispielsweise die Ausführungszeit, annotiert werden können. Mit diesen annotierten Ausführungszeiten ist es nun möglich, den gesamten Zeitbedarf zu simulieren.

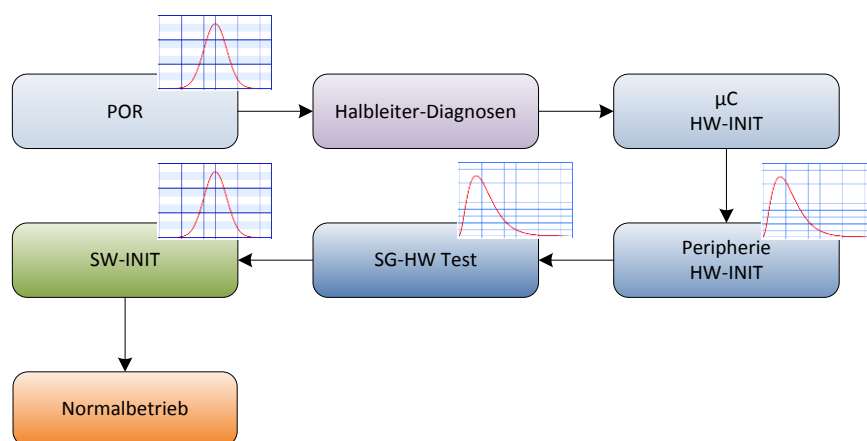


Abbildung X.11: Modell der Startphase mit Wahrscheinlichkeitsverteilungen für die einzelnen Akteure

An die einzelnen Akteure des Modells wurden die bereits diskutierten Ausführungszeiten annotiert. Da die Ausführungszeiten der meisten Akteure nicht statisch sind, sondern Varianzen besitzen, wurden die jeweiligen Varianzen ebenfalls

mit an das Modell übergeben, siehe Abbildung X.11. Die zeitlichen Abweichungen des POR und des SW-INIT Aktors variieren um einen Mittelwert. Es ist davon auszugehen, dass die resultierenden Zeiten meist nahe am Mittelwert liegen werden. Daher wurden diese beiden Aktoren mit einer Normalverteilung annotiert. Der Peripherie HW-INIT und SG-HW Test-Aktor wurden analog dazu mit einer Gammaverteilung annotiert: Die Zeiten variieren nur in die positive Richtung und es ist davon auszugehen, dass die resultierenden Zeiten am häufigsten in der Mitte des minimal und maximal möglichen Wertes liegen werden. Der  $\mu$ C HW-INIT Aktor ist keinerlei zeitlichen Varianzen unterworfen, so dass hier auch keine Verteilung annotiert wurde.

Abbildung X.12 Abbildung zeigt beispielhaft den Aufbau des SW-INIT SystemMoC-Aktors. Sobald ein Startsignal empfangen wird, wird aus dem deaktivierten Zustand (off) des Steuergeräts mittels der Initialisierung in den initialisierten Zustand (initialized) gewechselt. Daraufhin wird das Echtzeitbetriebssystem gestartet und in den aktiven Status (running) gewechselt. Durch ein Signal am Ausgang wird der Normalbetrieb über ein Ende der Startphase informiert.

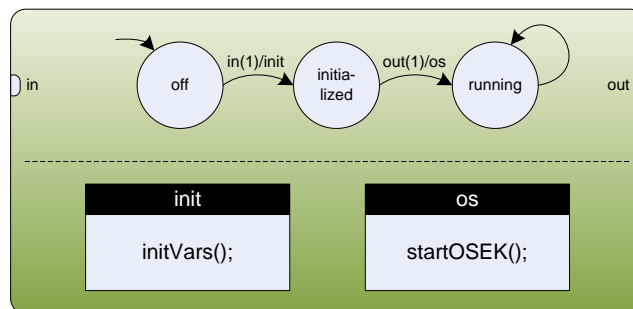


Abbildung X.12: Aktordarstellung von SW-INIT

Das in Abbildung X.11 erstellte Steuergerätemodell kann nun in ein Gesamtsystemmodell integriert werden. Somit lassen sich nicht nur die Startphasen eines einzelnen Steuergeräts, sondern auch Wechselwirkungen zwischen einer beliebigen Zahl von Steuergeräten im Gesamtsystem analysieren. Abbildung X.13 zeigt das Gesamtmodell, bei dem die einzelnen Steuergeräte den Abschluss ihrer Startphase über einen CAN-Bus mit den anderen Busteilnehmern kommunizieren.

### X.2.2.3. Simulationsergebnisse

Insgesamt wurden sechs unterschiedliche Testreihen durchgeführt. In der ersten Testreihe wurde die bisherige Startphase ohne Halbleiter-Diagnosen simuliert. Im Anschluss daran wurden die Halbleiter-Diagnosen in den Testreihen berücksichtigt. Die Dauer der Diagnosen wurde zwischen 5ms, 7,5ms, 10ms und 15ms variiert. In der sechsten Testreihe wurde der SG-HW Test Block um 5ms verkürzt und gleichzeitig Diagnosen von 10ms eingeplant. Die Deadline  $t_d$ , innerhalb derer die Startphase abgeschlossen sein muss liegt bei 200ms. Um einen repräsentativen Simulationswert zu erhalten wurde jede Testreihe 10,000 mal durchlaufen. Die Ergebnisse der einzelnen Testreihen sind im Meilensteinbericht detailliert erläutert.

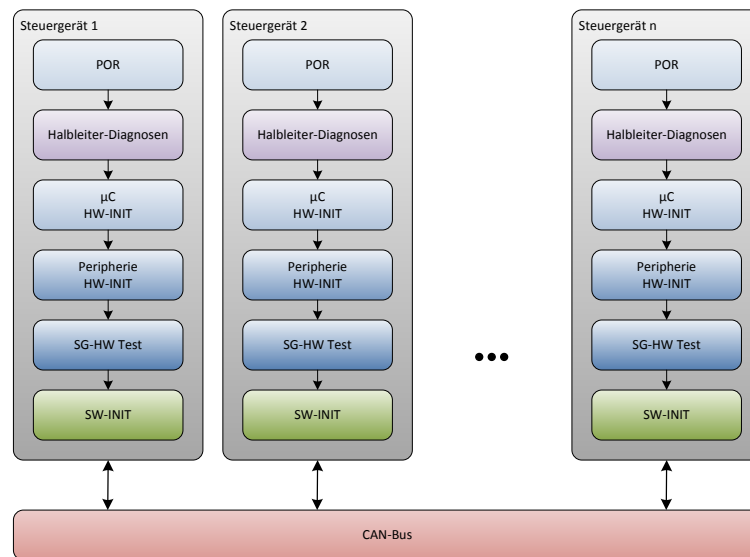


Abbildung X.13: Kommunikation der Steuergeräte über den CAN-Bus

Abbildung X.14 zeigt beispielhaft die Ergebnisse für die Startphase inklusive einer Halbleiter-Diagnose von 7,5ms, 10ms und 15ms.

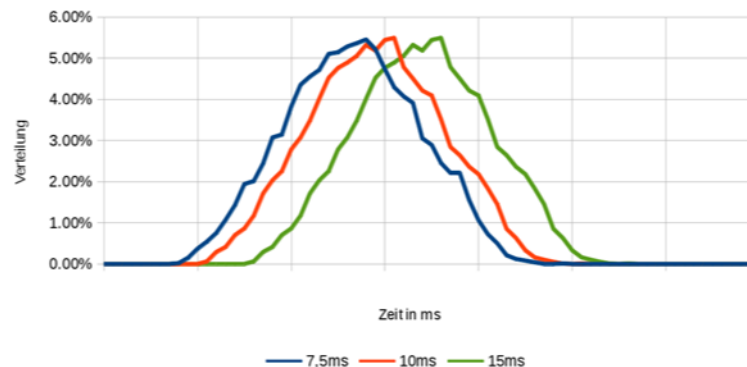


Abbildung X.14: Startphase mit 7,5ms, 10ms und 15ms Halbleiter-Diagnosen

Durch die Simulation der Startphase konnte gezeigt werden, dass:

1. Bereits existierende Startphasen die Deadlines verletzen können.
2. Das Einfügen von Diagnosen ohne das Modifizieren der bisherigen Startphase nur sehr eingeschränkt möglich ist.
3. Bereits kleine Änderungen in der bisherigen Startphase die Einsatzmöglichkeiten für Diagnosen stark verbessert.

Diese Simulationsergebnisse lassen sich nicht nur für die Startphase nutzen. Dadurch, dass die Startphase in vielen Punkten dem normalen Betrieb ähnelt, lässt sich diese Methodik in den regulären Betrieb übernehmen, um dort noch detailliertere Aussagen über den Einfluss von Diagnosen auf die reguläre Funktionalität treffen zu können.

### **X.2.3. Aufgabe 4.2.3 Kompaktierungsverfahren in Chip und im Gerät / Anpassung von Diagnosealgorithmen**

Nachdem die zeitliche Beschränkung für erweiterte Diagnose mit den vorherigen Verfahren bestimmt werden kann, muss nun noch ein Kompaktierungsverfahren zur Verringerung des Kommunikationsaufwandes zwischen einzelnen Chip-Komponenten und dem restlichen System entwickelt werden. Ziel ist dabei, neben einem vertretbaren Hardwareaufwand, vor allem auf eine weitgehende Erhaltung der diagnostischen Auflösung. Um dies zu zeigen, wurde ein Diagnosealgorithmus samt entsprechender Testinfrastruktur für die Logik-Diagnose einer Schaltung mittels stark kompaktierten Mustern entwickelt. Anhand der im Meilenstein 4.1.2 präsentierten Analyse bestehender Chip-Strukturen und der im Meilenstein 4.2.3 vorgestellten Erweiterungen wird nun ein diagnostisches Verfahren definiert, das eine hohe Auflösung mit vertretbaren Kosten ermöglicht.

In X.2.3.1 werden die für den diagnostischen Algorithmus notwendigen Anpassungen kurz vorgestellt. Im Anschluss wird in X.2.3.2 die entsprechende Test- und Diagnoseinfrastruktur erklärt. Weitere Details sind in den Publikationen des Literaturverzeichnisses zu entnehmen.

#### **X.2.3.1. Diagnostischer Algorithmus**

Der vorgeschlagene diagnostische Algorithmus [1] kann sowohl Haftfehler als auch komplexere intermittierende Fehler mit hervorragender Auflösung bestimmen. Hierzu wird das bedingte Haftfehlermodell verwendet.

Der Algorithmus analysiert kompaktierte Test-Signaturen, die aus mehrfachen Testmustern bestehen. Wenn die beobachtete Signatur  $S_{obs}(T_j)$  einer Mustermenge  $T_j$  von der Referenzsignatur  $S_{ref}(T_j)$  abweicht, so werden für jeden Schaltungsknoten  $v$  die bedingten Haftfehler  $(x=p_1 | T_j)_0_v, \dots, (x=p_n | T_j)_0_v$  sowie  $(x=p_1 | T_j)_1_v, \dots, (x=p_n | T_j)_1_v$  untersucht, wobei  $x$  wie oben die Eingänge der Schaltung und  $p_1, \dots, p_n$  die Testmuster in  $T_j$  bezeichnen. Jeder einzelne Fehler  $(x=p_i | T_j)_0_v$  bzw.  $(x=p_i | T_j)_1_v$  bewirkt eine Abweichung  $d(p_i, 0)$  bzw.  $d(p_i, 1)$  von der Referenzsignatur, die vorab berechnet wird. Die linearen Gleichungen

$$c_1 d(p_1, 0) \oplus \dots \oplus c_n d(p_n, 0) = S_{obs}(T_j) \oplus S_{ref}(T_j)$$

und

$$c_1 d(p_1, 1) \oplus \dots \oplus c_n d(p_n, 1) = S_{obs}(T_j) \oplus S_{ref}(T_j)$$

mit den Variablen  $c_1, \dots, c_n \in \{0,1\}$  beschreiben alle möglichen Bedingungen, welche das beobachtete Fehlverhalten mit einem Fehler an Knoten  $v$  erklären können. Ein Fehler am Knoten  $v$  kommt also nur dann als Ursache des Fehlverhaltens in Frage, wenn mindestens eine der beiden Gleichungen eine Lösung hat. Die Position in der sortierten Kandidatenliste ergibt sich dann aus der Anzahl der Sessions, in denen der Knoten  $v$  als möglicher Fehlerort identifiziert wird.

Der Algorithmus kann durch zusätzlichen Speicher mit der STUMPS-Standardarchitektur kombiniert werden. Aus diesem Grund kann er im Rahmen des im Meilenstein 4.1.3 vorgestellten Zielsystems umgesetzt werden.

### **X.2.3.2. Test- und Diagnoseinfrastruktur**

Der im Abschnitt 1 vorgestellte Algorithmus reduziert die Menge an Diagnosedaten, die auf dem Chip gespeichert werden müssen. Dennoch können die Gesamtkosten steigen, wenn die vorgeschlagene Methode nicht bei der Testmustererzeugung berücksichtigt wird.

Üblicherweise bestehen die Testmuster aus Zufallsmustern und deterministischen Mustern. Während ein reiner Zufallstest häufig keine ausreichende Fehlererfassung erzielt, können bei einem rein deterministischen Test hohe Kosten für die Speicherung der komprimierten Eingabedaten entstehen. Ein hybrider Test mit zufälligen und deterministischen Mustern kombiniert die Vorteile beider Ansätze. Pseudo-zufällige Muster sind einfach zu erzeugen und erkennen bereits einen großen Teil an Fehlern, die nicht zwingend an ein vorgegebenes Fehlermodell gebunden sein müssen. Schwer erkennbare Fehler werden im Anschluss daran durch deterministische Muster, welche in komprimierter Form abgespeichert werden, geprüft.

Beim Selbsttest mit hybriden Mustern reduzieren die Zufallsmuster einerseits die Kosten für den deterministischen Test. Andererseits decken sie aber auch viele Fehler außerhalb des vorgegebenen Fehlermodells ab. Um beide Vorteile auszunutzen, zielt der vorgeschlagene Diagnoseansatz auf einen hybriden Test mit langen Zufallsmusterfolgen ab. Die Herausforderung dabei ist, die Speicherkosten für Diagnose so weit wie möglich zu reduzieren und gleichzeitig eine hohe Defekterfassung zu erhalten. Die Hauptidee besteht darin, diejenigen Bereiche der Zufallsmusterfolge zu bestimmen, die mit einem hochauflösenden Diagnoseverfahren analysiert werden müssen, sowie diejenigen Bereiche, die mit weniger aufwendigen Verfahren behandelt werden können. Das Verfahren kann ebenso bei sehr langen pseudo-vollständigen Testmustern verwendet werden, um eine bessere Defekterfassung zu erreichen.

Das vorgeschlagene Kompaktierungsverfahren kann verwendet werden, um den Kommunikationsaufwand zwischen Chip und Steuergerät zu minimieren. Wenn jeder Chip die vorgestellte Test- und Diagnoseinfrastruktur unterstützt, brauchen nur noch hochkompaktierte Signaturen nach Außen übertragen werden, um Diagnose durchzuführen. Allerdings können die Herstellungskosten der Chips aufgrund des zusätzlichen Speichers zunehmen. Um die Gesamtkosten zu minimieren, können die entsprechende Test- und Diagnosedaten der Zielschaltung mithilfe eines funktionalen Busses wie z. B. CAN übertragen werden (MS 4.1.3). Ebenso kann die benötigte Test- und Diagnoseinfrastruktur wiederverwendet werden, um mehrere Schaltungen zu diagnostizieren. Hierzu müssen die Test- und BIST-Controller entsprechend angepasst werden.

### **X.2.4. Aufgabe 4.2.4 Detaillierte Analysen mittels ganzheitlicher Optimierung**

Die große Anzahl an Alternativen, die im Projekt DIANA für die erweiterte Diagnose entwickelt wurden, erfordert eine automatisierte Einbringung und Auswahl in das automobiler Gesamtsystem. Hierzu wird auf den vorgestellten Modellen eine ganzheitliche Optimierung, siehe Abbildung X.15, durchgeführt, die die vorhandenen Freiheitsgrade berücksichtigt und global optimiert. Folgende Freiheitsgrade werden dabei im Folgenden berücksichtigt:

- Abbildungsalternativen der Systemfunktionalität,
- verwendete Steuergeräte,
- Auswahl der Feldbusse und der Netzwerktopologie,
- Routing der Nachrichten,
- ob und welcher BIST auf einem Steuergerät implementiert wird,
- ob und mit welcher Periode der SBST auf einem Steuergerät eingeplant wird,
- Speicherort der Testmuster,
- Vergabe der Prioritäten für System- wie Diagnosefunktionen und -nachrichten

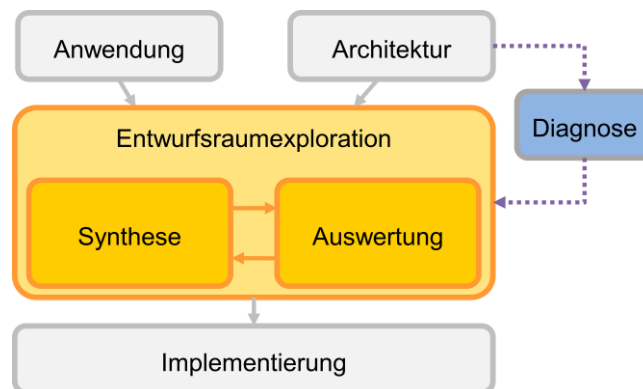


Abbildung X.15: Auf Basis der Modelle für die Anwendung, die Architektur und die erweiterte Diagnosefunktionalität kann die Entwurfsraumexploration durchgeführt werden.

Bei der Optimierung auf Systemebene müssen verschiedenste Anforderungen berücksichtigt werden. Wird beispielsweise bei der Ablaufplanung ein SBST zu häufig eingeplant, kann dies im Extremfall zusammen mit der Anwendungsfunktionalität zu einer Überlastsituation auf dem Steuergerät führen. Entsprechende Alternativen werden durch eine Zeitanalyse detektiert und aus den Lösungskandidaten entfernt.

Für den Systemdesigner stehen mehrere Zielgrößen im Konflikt. Entsprechend wird eine Mehrzieloptimierung durchgeführt, die den Suchraum in Hinblick auf verschiedene Zielgrößen gleichzeitig optimieren kann. Als Resultat erhält man eine Menge von Paretopunkten, die die Trade-Offs zwischen den einzelnen Zielgrößen darstellen. Ein *Paretopunkt* ist eine Implementierung, für die keine andere Implementierung gefunden wurde, welche in allen Zielgrößen besser ist. Die Paretopunkte liegen hierbei auf einer Hyperebene im n-dimensionalen Suchraum. Die im Rahmen von DIANA umgesetzte Entwurfsraumexploration unterstützt die gleichzeitige Optimierung in Hinblick auf die folgenden Zielgrößen (weitere können nach Bedarf ergänzt werden):

- **Testabdeckung:** Für jedes Steuergerät wird versucht, Diagnosefunktionen mit hoher Testabdeckung auszuwählen. Sind mehrere auf einem Steuergerät gewählt, wird als pessimistische Abschätzung nur die Testabdeckung der besten Diagnosefunktion berücksichtigt. Gemittelt wird über alle Hardwarekomponenten, für die Diagnose planbar wäre.



- **Verbesserung der Absicherung per SBST:** Je öfter die Hardware einer Anwendungsfunktion vor ihrem periodischen Ablauf von einem SBST überprüft werden kann, desto besser ist die zeitliche Auflösung.
- **Zeit zur Testdurchführung:** Die Zeit, die zur Übertragung der Testmuster, Ausführung des Tests und Übertragung der Ergebnisse benötigt wird – und damit die Zeit, die eine Steuergerät länger aktiv ist bevor es in den Teilnetzbetrieb geht oder die Nachlaufphase endet – sollte möglichst kurz sein.
- **Energieverbrauch:** Jede Ausführung von System- oder Diagnoseanwendung verbraucht Energie. Dies ist zu minimieren.
- **Kosten:** Neben den Kosten, die alternative Steuergeräte mit Hardwareunterstützung zur erweiterten Diagnose zu den Kosten des Gesamtsystems beitragen, werden hier außerdem die Kosten für die Speicherung der Testmuster berücksichtigt.

### X.2.5. Ergebnisse

Die entwickelte Simulation von Steuergerätearchitekturen erlaubt den Nachweis von Reserven in der Aufstartphase von Steuergeräten, die zur erweiterten Diagnose verwendet werden können. Dabei berücksichtigt die Simulation Unsicherheiten in der Ausführungszeit durch den Start des Steuergeräts selbst, wie auch der Peripherie und der Umgebung. Dies ermöglicht erstmals statistische Aussagen über das Aufstartverhalten zu treffen und Trade-offs zwischen diagnostischer Auflösung und der Wahrscheinlichkeit, den Start unter allen möglichen Umständen vor anvisierten Deadlines zu bewältigen, zu identifizieren.

Neben der Feststellung der für die Diagnose nutzbaren Zeit wurden Verfahren entwickelt, um den zusätzlichen Speicher zur Vorhaltung der Diagnosemuster zu minimieren. Diese basieren auf der STUMPS-Architektur und erlauben es, die Testmuster sowohl geeignet zu codieren als auch bei gleichartigen Komponenten die Muster zentral vorzuhalten und, ohne den Betrieb des restlichen Systems zu stören, zu übertragen. Durch diese Mehrfachverwendung kann eine weitere Reduktion des Speicheraufwands erreicht werden.

Durch die werkzeuggestützte Integration von SBST und BIST-Verfahren in das Gesamtsystem können bestehende Systeme mit minimalem Aufwand nachgerüstet werden. Dabei werden sowohl das Scheduling und der Speicher am Steuergerät wie auch die Kommunikation über die Feldbussysteme berücksichtigt. Die entwickelten Werkzeuge erlauben außerdem, zukünftige Systeme mit erweiterter Diagnose auszurüsten und dabei von Beginn an ein diagnosegewahres System hinsichtlich mehrerer Zielgrößen wie Kosten, etc. zu entwickeln. Da alle Verfahren automatisiert sind, ermöglicht dies eine umfassende Entwurfsraumexploration, die alle für den Entwickler wichtigen Entwurfsentscheidungen und Zielgrößen berücksichtigen kann. Die resultierenden Vorschläge zur Umsetzung der diagnosegewahren E/E Architektur erhöhen die Konfidenz der Entwickler, die richtige Systemimplementierung weiterzuverfolgen.

### **X.3. Aufgabe 4.3 Methoden und Algorithmen zur optimalen Diagnoseauflösung für analoge und gemischt digital-analoge Baugruppen**

#### **X.3.1. Motivation**

Die Diagnose im System erfordert geeignete Zugangspunkte in der zu analysierenden elektronischen Schaltung. Damit die Analyse unter Echtzeitbedingungen auch wirtschaftlich betrieben werden kann, wird heute in der Mikroelektronik gefordert, dass die verwendeten Diagnosegeräte einfach und flexibel zu handhaben sind. Für elektronische Bauteile, die in einem komplexen System wie z. B. einem Steuergerät aus der Automobiltechnik verbaut worden sind, ist es deshalb wichtig, zuverlässig und reproduzierbar alle notwendigen Diagnose- und Analysedaten rasch zu extrahieren. Es ist also zweckmäßig die Zeit für die Aufnahme der Diagnosedaten aus dem Bauteil möglichst gering zu halten, um dann erst in einer Nachbearbeitung – offline und kostengünstig – die tatsächliche Diagnose und die Extraktion der Fehlerursachen durchzuführen. Damit werden für die Diagnose und Analyse von analogen und digital-analog gemischten Bauteilen die Bildung von Signaturen gefordert, welche alle Anforderungen an eine hohe Fehlererkennung und damit die Diagnose von Fehlverhalten von elektronischen Bauteilen erfüllen und gewährleisten können.

#### **X.3.2. Ziel**

Entwicklung von Methoden und Algorithmen zur optimalen Diagnoseauflösung (Trennung der Antworten der Diagnose von übertragungsbedingten Störeinflüssen) für analoge und gemischt digital-analoge Baugruppen

#### **X.3.3. Teil 1**

Die Anforderungen an eine hohe Fehlererkennung und damit eine zweckmäßige und wirtschaftliche Diagnose von Fehlverhalten von elektronischen Bauteilen kumulieren in der Fähigkeit, die Diagnose und Analyse mit genau der noch geforderten Prüfschärfe durchführen zu können, welche dann zu dem optimalen und damit minimalen Ressourcen- und Kostenverbrauch führt. Der von Continental bereitgestellte und vorgegebene Prüfling, Continental Demonstrator, bestehend aus Sensor, Übertragungskanal und Analog-Digital-Wandler (ADC) für die Digitalisierung und Kompression der analogen Mess- bzw. Testwerte muss so diagnostiziert werden können, dass mögliche Fehlerursachen später wieder aus diesen komprimierten Daten extrahiert werden können. Der Prüfling muss also durch ein geeignetes Messverfahren aus seiner Umgebung herausgeschnitten werden können.

##### **X.3.3.1. Ergebnisse**

Die messtechnischen Zugänge (Signalpfade) zu den internen Beobachtungs- und Steuerungsknoten müssen damit hinsichtlich ihrer elektrischen Eigenschaften extrahiert und berücksichtigt werden können, so dass auch mit vergleichsweise einfachem Prüfinstrumentarium und Messprozeduren eine hohe Diagnoseschärfe und damit genaue und möglichst eindeutige Fehlerlokalisierung erreicht werden kann.

Ausgehend vom aktuell zu ermittelnden Stand der Technik wurden die Anwendbarkeit und insbesondere Einschränkungen existierender Verfahren zur Charakterisierung von Sensorik (Gutverhalten, Schlechtverhalten) und der Eliminierung von parasitären Einflüssen auf Prüfsignale in der vorliegenden Anwendung analysiert. Da es sich bei dem vorhandenen Sensorcluster von Continental, um ein hochmodernes und komplexes Cluster mit komplett vergossener und verbauter Sensorik handelt, ist der Zugang von außen nicht gegeben. Die Signalübertragungstrecke vom Sensor zum Analog-Digital-Wandler wird durch den Kabelbaum realisiert. Dieser ist ebenfalls nicht zugänglich. Damit müssen alle Charakteristiken aus der digitalen Signatur des Analog-Digital-Wandlers gewonnen werden können.

Das Verhalten des Wandlers stellt damit das Nadelöhr für die Bewertung der Übertragungstrecke und des Sensors dar, ein statisches nichtlineares Bauelement. Die Übertragungstrecke wird dazu in seinem Übertragungsbereich als lineares zeit-invariantes System (LTI-System) betrachtet. Die Unterscheidung zwischen Gutverhalten und Schlechtverhalten kann damit mittels Limits und Toleranzen (Toleranzbänder) eingegrenzt werden.

Das erforderliche Modell des ADC, die Nachbildung seiner statischen Übertragungskennlinie, wurde dazu erforscht. Eine Untersuchung der Genauigkeit und Auflösung von Konvertern für statische Signale, stationäre Signale und dynamische Signale im Betrieb ist damit möglich. Die Parametrisierung dieses Wandlermodells erfolgt dann später mithilfe der in der Realität gemessenen Signatur.

Es wurde weiterhin zusammen mit Continental herausgearbeitet, dass die wesentliche Systemschnittstelle zum Sensorcluster eine digitale Schnittstelle ist. Damit besteht die Herausforderung in der Diagnosefähigkeit mit Hilfe der digitalen zu erzeugenden Signaturen. Wesentliches Bauteil ist hier der im Mikroprozessor verbaute Analog-Digital-Wandler. Mithilfe dieses Wandlers werden im Betrieb die wichtigen Sensordaten aufgenommen.

#### **X.3.4. Teil 2**

Die genaue Untersuchung des Gesamtsystems, die daran anschließende Definition der für den Test geeigneten Abstraktionsebene, hat zu einem Systemmodell mit den wichtigsten Grundblöcken für eine gemischt analog-digitale Baugruppe, das ausgewählte Sensorcluster, geführt. Die Diagnose dieses Systems erfordert geeignete Zugangspunkte in der zu analysierenden elektronischen Schaltung. Da das Gesamtsystem aus einer Reihenschaltung von Grundblöcken aufgebaut ist und seine Grundblöcke selber eine Verhaltensbeschreibung der Realität darstellen, ist der Zugang ohne integrierte Testpunkte aber nur im Sinne eines Blackbox Tests möglich. Für die Diagnose im Betrieb bedeutet dies die Auswahl eines geeigneten Verfahrens. Es soll aus den erhaltenen Daten alle Rückschlüsse auf die Funktionsfähigkeit der Baugruppe beziehungsweise der einzelnen Grundblöcke selbstständig ziehen können. Die Daten müssen sowohl im Betrieb unter Einsatzbedingungen aufgenommen also auch ausgewertet werden können. Da davon ausgegangen werden muss, dass jedes Steuergerät seinen eigenen charakteristischen Datensatz erzeugt, muss aus den erhaltenen Daten das jeweilige individuelle Gutverhalten beziehungsweise charakteristische Einbauverhalten unter

den jeweiligen Einsatzbedingungen und Betriebsbedingungen, denen das Steuergerät ausgesetzt ist, aufgenommen und detektiert werden können. Die Auswertung der aufgenommenen Daten muss den Anforderungen an die Bereitstellung solcher Informationen im Steuergerät und im Gesamtsystem genügen. Dies können sowohl Abweichungen über einen großen längeren Zeitraum als auch kurze Spitzen im Betrieb sein.

Eine solche Methode muss es also ermöglichen, aus einer Menge von Messungen die Entscheidung zu treffen, die einen solchen Diagnosefall zweifelsfrei entscheiden. Eine solche Fragestellung reduziert auf solche Systeme und Verfahren, welche eine Klassifizierung zwischen guten und schlechten Kandidaten (Messungen) ermöglichen. Ein erster Schritt in die Einarbeitung in ein solch komplexes Fachgebiet, ist die Beschäftigung mit und Untersuchung von linearen Klassifizierern. Solche Klassifizierungsverfahren erlauben in Bezug auf eine vorher festgelegte Hyperebene (Trennebene) zwischen guten und schlechten Instanzen auf der einen und auf der anderen Seite der Hyperebene zu unterscheiden. Das Verfahren muss einwandfrei funktionieren, im Sinne des Engineering Konvergenz aufweisen und in endlicher Zeit zu einem Ergebnis gelangen. Die Einsatzbandbreite muss gewährleistet sein, dass theoretische Verständnis der zu Grunde liegenden Algorithmen vorhanden sein. Deshalb war eine Aufarbeitung der in der Literatur bestehenden Veröffentlichungen in der Form nötig, in welcher die wesentlichen Inhalte direkt und anschaulich aufgenommen werden können. Damit soll Vertrauen für diese mathematisch hoch komplexen Verfahren geschaffen und gewonnen werden. Insbesondere der Anwender würde sich sonst schwer tun, sich für solche Verfahren in sicherheitskritischen Systemen zu entscheiden.

#### **X.3.4.1. Ergebnisse**

In dieser Phase wurde die Untersuchung von geeigneten Verfahren für die Klassifizierung von solchen Test und Diagnoseverfahren vorangetrieben und ein neues und kostengünstiges Testverfahren vorgestellt. Die Auswahl des Verfahrens wurde dabei durch die Randbedingungen der Anforderungen an eine at-speed und on-line (im Betrieb) Messung bestimmt. Das Verfahren nutzt zur Klassifizierung eine geringe Anzahl von geeigneten Tests. Diese müssen im Vorfeld in Abhängigkeit der Trainingsmenge bestimmt werden. Zu jedem dieser Teststimuli wird dann das Antwortsignal zu einem bestimmten Zeitpunkt gegen einen Grenzwert getestet, weshalb man pro Teststimuli nur einen Größenvergleich hat. Die eigentliche Arbeit ist also bei dem Verfahren im Vorfeld zu leisten. In jedem Fall muss eine Trainingsmenge generiert werden, welche das auf Fehler zu untersuchende Steuergerät nach Pass und Fail zu klassifizieren hat. Diese Trainingsmenge, genauer die Impulsantworten der einzelnen Messreihen, werden dann im Impulsantwortraum betrachtet und partitionieren alle passed und failed Instanzen. Die Einbettung des Sensors in die Umgebung des Mikroprozessors wird so geeignet nachgebildet. Die trennenden Hyperebenen werden in dem vorgestellten Verfahren automatisch gewählt. Man ist also nicht darauf angewiesen, alle failed Instanzen wie bei vergleichbaren Verfahren in verschiedene Cluster aufzuteilen.

Ein zweites wichtiges Ergebnis dieser Phase ist die Erarbeitung einer weiteren geeigneten Methode, um die Implementierung von hochauflösenden analogen

Testpunkten in nicht zugänglichen Teilen von Schaltungen vorzuschlagen. Diese Methode, ein hochauflösender Ringoszillator, verwendet dabei ein Minimum an rein digitaler Logik. Der Ringoszillator kann Flächen sparend und sehr kostengünstig an beliebigen Testpunkten zugeschaltet werden. Die Methode erzeugt dabei eine digitale Signatur des analogen Testpunktes. Sie erfüllt die Anforderungen an eine robuste Übertragung der Informationen zum Mikroprozessor.

### **X.3.5. Teil 3**

Damit liegen alle wichtigen Verfahren vor, um sowohl Blackbox- als auch dezidierte in-Situ-Messungen und Tests durchführen zu können. Der Aufbau von Antastvorrichtungen entfällt daher, da kein Zugang von Außen ermöglicht werden kann.

#### **X.3.5.1. Ergebnisse**

Es wurde die digitale Signatur eines nicht bekannten realen Stimulus untersucht und bewertet, sowohl in Hinblick auf seine statischen und dynamischen Wesensanteile. Dazu wurde die von Conti gelieferte Soft- und Hardware geeignet programmiert, um die dedizierte on-Chip-Analyse mit Hilfe des eingebauten Mikroprozessors selbst umzusetzen.

Ein wichtiges Teilziel war damit die Realisierung eines Tests der Konverterarchitektur im Mikroprozessor selbst. Die Verwendung des analogen Fehlersimulator AFSIM der Fraunhofer-Gesellschaft (FHG) in Dresden, um Fehlerbilder des ADC nachbilden zu können, wurde aus Zeit- und Prioritätsgründen nicht nachgegangen. Der Abgleich mit den Erfahrungen von Conti bezüglich der verwendeten Betriebsmodi und Auswertungen auch unter Berücksichtigung der Conti-Spice-Datenbanken wurde durchgeführt. Da die zur Verfügung gestellten realen Daten aber noch keine Vergleiche mit Fehlermodellen und Fehlerdaten beinhalten, wurde das Konzept dahingegen erweitert, dass die Erzeugung des Teststimulus im Betrieb erfolgen muss, die Bewertung der Testantworten dieses bekannten realen Stimulus ebenfalls.

Als geeigneter Testfall wurde von Conti die Daten und Erfahrungen zu einer realen Testschaltung zur Messung von defekten Kondensatoren zur Verfügung gestellt. Aufbauend darauf wurde eine Schaltung für die Aufnahme und Diagnose der statischen als auch dynamischen Charakteristiken eines Filters entworfen, und für den Einsatz in einem Automotive Steuergerät validiert. Defekte bzw. offene Kondensatoren werden dabei über die Messung einer Sprungantwort mit Hilfe des eingebauten Microcontrollers vom Typ TC1797 detektiert.

Damit ist es nun möglich, das dem on-Chip Mikroprozessor vorgeschaltete Anti-Aliasingfilter autark auszumessen und vollständig zu verifizieren. Prototyp, Kapazitätsarray mit Möglichkeiten zur Fehlernachbildung und Integration in das Steuergerät von Conti wurden bewerkstelligt. Der mit Continental zusammen erstellte Demonstrator zeigt den erfolgreichen Selbsttest und Diagnose fehlerhafter Kapazitäten, Stimulus und Auswertung erfolgen dabei im Mikroprozessor selbst. Ein möglicher Einsatz im Betrieb unter Echtzeitbedingungen ist damit verifiziert. Die Kennwerte des Selbsttests für die Berücksichtigung der Diagnoseabdeckung in der FMEA in Abstimmung mit den Aufgabe 3.1 und 3.2 liegen damit vor.

### **X.3.6. Zusammenfassung**

Die Auswahl und Verwendung eines lernenden Diagnoseverfahrens erlaubt einen Blackbox Test unter Echtzeitbedingungen. Die Trennung der Messwerte kann damit mit der nötigen hohen Prüfschärfe und möglichst wirklichkeitsgetreu erfolgen. Alle Entscheidungslimits sollen mithilfe dieser Methodik erst unter Einsatzbedingungen im Einsatz und unter Echtzeitbedingungen erzeugt werden können. Zusammen mit dem Auftraggeber Continental auch in Zusammenarbeit mit Aufgabe 3.1 und 3.2 wurden folgende Ergebnisse erzielt:

- Es wurde ein für die Diagnose und Test geeignetes Modell für den Analog-Digital-Wandler (ADC) erstellt.
- Für den Einbau von analogen Testpunkten wurde das Konzept eines Ringoszillators erarbeitet.
- Für die Entdeckung von Diagnoselücken im Anti-Aliasing Filter vor dem AD-Wandler des Microcontrollers wurde ein Selbsttest auf der Conti-Steuergeräteelektronik entwickelt.

### **X.3.7. Literatur**

- [1] M. F. Toner, G. W. Roberts, A BIST Scheme for an SNR Test of a Sigma Delta ADC, International Test Conference (ITC), 17-21 Oct 1993, 805-814.
- [2] H. Mattes, S. Kirmser, S. Sattler, Mixed-Signal Mikrocontroller Selbst-Test, 19. ITG/GI/GMM Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen, Erlangen 2007, pp. 40-47

## **X.4. Aufgabe 4.4 Software basierter Selbsttest für Test und Diagnose digitaler Logik in Automotive Systemen**

### **X.4.1. Ziel**

Das Ziel von Aufgabe 4.4 ist die Entwicklung strukturbasierter Test- und Diagnosemethoden für Prozessorarchitekturen. Dies umfasst die Erzeugung von Testmustern, die bei Prozessoren übliche Testanwendung in Form softwarebasierter Selbsttests (SBST), sowie die auf Testantworten basierende Logikdiagnose. Darüber hinaus wurden Konzepte erarbeitet, wie sie in zukünftigen Prozessoren zur Verbesserung der strukturellen Test- und Diagnostizierbarkeit umgesetzt werden können.

Im Gegensatz zu bekannten softwarebasierten Ansätzen wird hier nicht nur die Funktion des Zielprozessors, sondern auch dessen Struktur getestet. Dies ermöglicht die Modellierung struktureller Fehler, und somit auch die Bestimmung einer strukturellen Fehlerabdeckung. Außerdem weisen strukturelle Methoden im Vergleich zu funktionalen Testmethoden grundsätzlich höhere Fehlerabdeckung auf. Im Bezug auf strukturelles Testen stellt die softwarebasierte Testanwendung eine Herausforderung dar: Das softwarebasierte Ansteuern bzw. Auslesen von prozessorinternen Signalen ist durch die funktionalen Eigenheiten des Instruktionssatzes beschränkt. Zusätzlich wird die Fehlererkennung (und -diagnose)

durch indeterministische Schaltteile erschwert, wie sie in modernen Prozessoren zwecks Performancesteigerung integriert sind. Um Fehler in solchen Strukturen abzudecken, werden speziell angepasste Testverfahren benötigt.

Die strukturelle Logikdiagnose ist der Vorgang, für erhaltene Testantworten einer defekten Schaltung die wahrscheinlichste Fehlerstelle zu ermitteln. Im vorgestellten Lösungsansatz wird eine sequentielle Pipelinestruktur des Zielprozessors zunächst in eine kombinatorische Darstellungsform überführt. Auf diese Weise können bestehende kombinatorische Diagnosealgorithmen (wie POINTER, [1]) angewandt werden, welche hohe diagnostische Auflösungen erreichen. Die Diagnosefähigkeit wird stark davon beeinflusst, wie gut die Testmuster die Unterscheidung der Fehlerstellen ermöglichen. Zur Verbesserung der diagnostischen Auflösung werden im erarbeiteten Konzept daher zwei Phasen mit sich ergänzenden Testmustergruppen bei der Diagnose berücksichtigt. In der ersten Phase findet kein Forwarding zwischen Pipeline-Stufen statt. Eine Pipeline kann als äquidistante sequentielle Schaltung aufgefasst werden, eine kombinatorische Diagnose kann ohne Berücksichtigung der Pipelineregister stattfinden. Bei diesem Vorgang wird bereits ein Großteil einfach zu erkennender Haftfehler abgedeckt. In der zweiten Phase wird Forwarding explizit durchgeführt, um die rückführenden Signale ebenfalls abzudecken und im Fehlerfall diagnostizieren zu können. Ein wichtiger Vorteil des POINTER-Diagnosealgorithmus ist, dass er unabhängig vom verwendeten Fehlermodell ist. So können selbst Fehler lokalisiert werden, die bei der Testmuster-generierung nicht berücksichtigt wurden.

#### **X.4.2. Methoden**

Moderne Mikroprozessoren bestehen typischerweise auf mehreren Pipeline-Strukturen, wie in Abbildung X.1 dargestellt. Jede einzelne Pipeline verarbeitet eine bestimmte Art von Instruktionen, wie beispielsweise Rechenoperationen, oder Speicherzugriffe. Diese werden der Pipeline von der Dekodierungseinheit zugewiesen. Die Pipelines enthalten mehrere Verarbeitungsstufen, die jeweils aus einem sequentiellen und einem kombinatorischen Schaltteil bestehen. Die sequentiellen Elemente stellen die internen Pipeline-Register dar, die kombinatorischen Teilschaltungen führen die entsprechenden Operationen auf den Registerinhalten durch. Zwischen nicht-aufeinanderfolgenden Pipeline-Stufen können zusätzliche kombinatorische Verbindungen existieren, die sog. Forwarding-Signale. Sie stellen geschwindigkeitssteigernde Abkürzungen im Datenpfad dar.

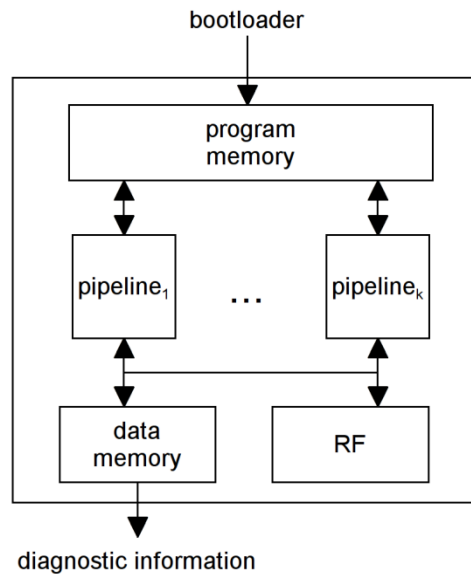


Abbildung X.16: Prozessorarchitektur

Bei der Durchführung struktureller softwarebasierter Selbsttests werden die Prozessorkomponenten (u.a. die Pipelines) einzeln behandelt. Zunächst dienen existierende ATPG-Tools (Automatic Test Pattern Generator) dazu, eine Menge von Testmustern zu erzeugen. Dazu wird ein Fehlermodell (z.B. Haftfehler), sowie ein Optimierungsziel (Testabdeckung, diagnostische Auflösung) festgelegt. Man unterscheidet zwischen sequentiell und kombinatorischem ATPG. Das sequentielle ATPG ist mit hohem Rechenaufwand verbunden, und stößt mit kommerziellen Programmen an seine Grenzen. Je nach Werkzeug ergeben sich eventuell weitere Einschränkungen für die Testanwendung, größere Testdatenmengen oder längere Testroutinen. Die Umwandlung struktureller Testmuster in ein ausführbares Selbsttest-Programm geschieht in Abstimmung mit den bei der Mustergenerierung eingesetzten Nebenbedingungen. Ein Testmuster stellt eine Sequenz von Instruktionen dar, welche bei der Testanwendung ausgeführt wird. Die Ergebnisse dieser Instruktionen in den Registern werden zwecks späterer Offline-Diagnose im Speicher abgelegt.

Die kombinatorische POINTER-Diagnose wurde an die softwarebasierte Testdurchführung angepasst. Zunächst einmal arbeitet der Diagnosealgorithmus auf einer kombinatorischen Darstellung der ursprünglichen Pipeline. Dieser wird konstruiert, indem die Pipelinestruktur „entrollt“ wird, d.h. kombinatorische Teilschaltungen der Pipeline werden entsprechend ihres Auftretens ohne Pipelineregister verbunden. Bei Forwarding-Signalen werden kombinatorische Schaltteile dupliziert. Hierbei werden nicht nur Gatter, sondern auch Fehlerorte dupliziert. Die Diagnose muss demnach so angepasst werden, dass sie die Evidenzen für mehrere duplizierte Fehlerorte zusammenfasst.

Desweiteren wurden auch Hardware-Konzepte zur Verbesserung der Diagnosefähigkeiten erarbeitet, u.a. die Einführung von speziellen Testbefehlen, die Vorteile eines integrierten Signaturregisters für die Diagnose, als auch die Erweiterung der softwarebasierten Testmethode für Peripheriekomponenten von



Prozessoren. Die Kompaktierung von Testantworten zur Minimierung des benötigten Speicherplatzes und der Bandbreite wurde ebenfalls behandelt.

#### **X.4.3. Ergebnisse**

Die entwickelten Test- und Diagnosemethoden wurden anhand der Integer-Pipeline des TriCore-Prozessors TC1797 für das Haftfehlermodell evaluiert. Es wurde eine Menge von 650 Testmustern mit 92.23% Fehlerabdeckung erzeugt. Die Größe des dazugehörigen Testprogramms liegt bei 40KB.

Anschließend wurden 500 verschiedene Fehler (Haftfehler, Brückfehler, Verzögerungsfehler) simuliert und diagnostiziert. Tabelle X.1 enthält die erreichten diagnostischen Auflösungen, die je nach Defektmechanismus zwischen 91.3% und 100.0% schwanken. In der linken Ergebnisspalte (Rank = 0) gilt ein Fehler nur dann als diagnostiziert, sofern der diagnostizierte Fehlerort übereinstimmt. In der rechten Spalte zählt ein Fehler als korrekt diagnostiziert, wenn der korrekte Fehlerort einer der 5 wahrscheinlichsten Stellen ist.

Fault Type	Diagnostic Resolution	
	Rank = 0	Rank < 5
Stuck-At	86.2%	100.0%
Dom. AND-Bridge	87.0%	94.6%
Dom. OR-Bridge	85.9%	96.5%
Slow-to-Rise Delay	81.7%	91.5%
Slow-to-Fall Delay	87.0%	91.3%

Tabelle X.1: Diagnostische Auflösung

#### **X.4.4. Demonstrator-Emulation der SBST-Methode**

Zur Demonstration der erarbeiteten SBST-Methode wurde die CHIPit-Platinum-Emulationsmaschine von Synopsys ausgewählt. Sie ist mit 3 Virtex5-FPGAs (xc5vlx330), einigen MByte SDRAM und einer LCD-Anzeige ausgestattet. Die Kommunikation zwischen der Emulationsmaschine und einem Host-Computer erfolgt per UMR-Bus, ein speziell zu diesem Zweck von Synopsys entwickeltes Protokoll. Die Maschine ermöglicht die Emulation großer Schaltungen. Der Aufbau des Systems, welches zur Validierung der an der Universität Stuttgart entwickelten SBST-Methode dient, wird in Abbildung X.17 dargestellt.

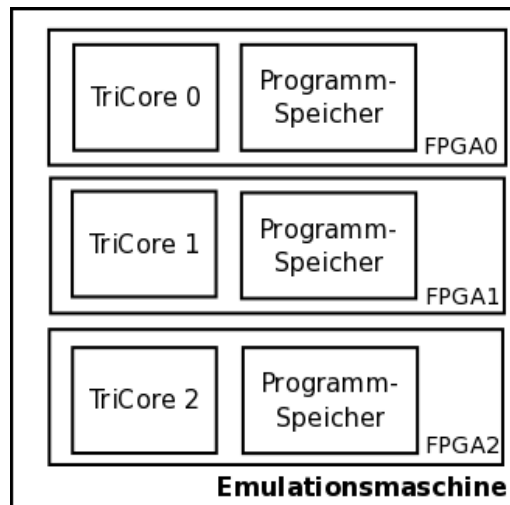


Abbildung X.17: Mehrere TriCore-Processoren in der Emulationsmaschine

Das System umfasst drei TriCore-Processoren, wobei ein einzelnes FPGA jeweils einen Prozessor samt Programm-Speicher enthält. Obwohl der Prozessor nur knapp 20% der Fläche eines FPGAs benötigt, und mehrere Clock-Domains vorhanden sind, begrenzen die zur Verfügung stehenden Taktpuffer die Anzahl der TriCore-Processoren pro FPGA. Allerdings sind drei TriCore-Processoren ausreichend, um die entwickelten Diagnose-Methoden zu validieren. Per UMR-Bus hat man vom Host-Computer aus Zugriff auf jedes der drei FPGAs. Diese Schnittstelle ermöglicht bidirektionale Kommunikation, über die das direkte Beschreiben des Programmspeichers und die Ansteuerung des Reset-Signals stattfindet. Es können zwei verschiedene Testmodi ausgewählt werden. Beim Werkstatt-Test wird das Testprogramm in den Programmspeicher übertragen und ausgeführt. Im Anschluss wird das Ergebnis vom Host-Computer ausgelesen und angezeigt. In Folge kann ein identifiziertes, fehlerhaftes Steuergerät als Reparaturmaßnahme ausgetauscht werden. Bei der Testdurchführung im Betrieb stellt der Host-Computer eine andere funktionale Komponente des Gesamtsystems Automobil dar, die als Testdatenquelle dient. Das Testprogramm wird nur im Leerlauf des Prozessors durchgeführt, um die bestehende Systemfunktionalität nicht zu unterbrechen. Falls bei diesem Betriebstest ein Fehler gefunden wird, geht das Gesamtsystem in einen sicheren Zustand über.

#### X.4.5. Literatur/Referenzen

- [1] S. Holst and H.-J. Wunderlich, "Adaptive debug and diagnosis without fault dictionaries," *Journal of Electronic Testing*, vol. 25, no. 4-5, pp. 259–268, 2009.

## XI. Abkürzungen

A	Auftretenswahrscheinlichkeit
A/D	Analog / Digital
ADC	Analog-Digital-Converter
ADE	Analog Design Environment
AD-Wandler	Analog-Digital-Wandler
AES	Advanced Encryption Standard
aFSIM	analoger Fehlersimulator des Fraunhofer IIS/EAS
ALU	Arithmetic Logic Unit
AP	Arbeitspaket
API	Application Programming Interface
ASIC	Application-Specific Integrated Circuit
ASIL	Automotive Safety Integrity Level nach ISO 26262 Norm
ATE	Automatic Test Equipment
ATPG	Automatic Test Pattern Generation
B	Bedeutung (Fehlerschwere)
BB	Basis-Block
BISR	Buit-in Self Repair
BIST	Built In Self Test
CAN	Controller Area Network
CC	Communication Controller
CCP	CAN Calibration Protocol
CDC	Clock Domain Crossing
CPC	Cross Parity Check
CPU	Central Processing Unit
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CR	Carrier Sense Multiple Access / Collision Resolution
CTL	Core Test Language
CVT	Continuously Variable Transmission
DAC	Digital-Analog-Converter
DA-Wandler	Digital-Analog-Wandler
DC	Diagnostic Coverage
DC fault model	Direct Current fault model
DCB	Don't Care Bit
DD	Detectability Degree
DED	Double Error Detection
DFT	Design for Testability
DM	Detektierbarkeitsmaß
DMA	Direct Memory Access
DNL	differentielle Nichtlinearität
dpm	Defects per Million
DSP	Digital Signal Processor

DUT	Device Under Test
E	Entdeckungswahrscheinlichkeit
ECC	Error Code Correction
ECCD Register	Error Code Correction Detection Register
ECCS Register	Error Code Correction Safety Register
ECU	Electronic Control Unit
EDA	Electronic Design Automation
EDT	Embedded Deterministic Test
EEPROM	Electrically Erasable Programmable Read-Only Memory
EGAS	Standardisiertes E-Gas Überwachungskonzept für Benzin und Diesel Motorsteuerungen
EMC	Electromagnetic Compatibility
EMV	Elektromagnetische Verträglichkeit
ESD	Electrostatic Discharge
ETRR Register	Error Tracking Register
FIFO	Speicher nach dem First In-Fist Out-Prinzip
FLEXRay	Serielles, deterministisches und fehlertolerantes Feldbussystem
FMEA	Failure Mode and Effects Analysis
FMEDA	Failure Modes Effects and Diagnostic Analysis
FPGA	Field Programmable Gate Array
FPI Bus	Flexible Peripheral Interconnect Bus
FSBST	Functional Software Based Self Test
FSM	Finite State Machine
FTDMA	Flexible Time Division Multiple Access
GALS	Globally Asynchronous Locally Synchronous
GL	Gate-Level
GPTA	General Purpose Timer Array
GUI	Graphical User Interface
HM	Hauptmeilenstein
HW	Hardware
I/O	Input/ Output
IC	Integrated Circuit
IDDq	Versorgungsstrom im Ruhezustand
IEC	International Engineering Consortium
INL	integrale Nichtlinearität
IP	Intellectual Property
ISA	Instruction Set Architecture
ISCAS	IEEE International Symposium on Circuits and Systems
ISO	International Organization for Standardization
JTAG	Joint Test Action Group
KFZ	Kraftfahrzeug
LCSU	Lockstep Control Unit
LED	Licht emittierende Diode
LFM	Latent Fault Metric

LFSR	Linear Feedback Shift Register
LIN	Local Interconnect Network
LMB	Local Memory Bus
LSL	Lower Specification Limit
LTL, UTL	Lower Test Limit
LVS	Layout-Schematic Vergleich
MAC	Multiplier Accumulator
MBIST	Memory Built In Self Test
MC	Microcontroller (see context)
MC	Muller-C-Element (see context)
MISR	Multiple Input Signature Register
MUX	Multiplexer
NMI	Non Maskable Interrupt
NVM	Non Volatile Memory
OBD	On-Board-Diagnose
OEM	Original Equipment Manufacturer
OpAmp	Operationsverstärker
OPV	Operationsverstärker
OSEK	Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug
OTCG	Opcode Test Code Generator
P&R	Place & Route
PC	Program Counter
PCP	Peripheral Control Processor
PLL	Phase-Locked Loop
POD	Probability of Detection of a Fault
PWM	Pulse-Width Modulation
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RLB	Re-configurable Logic Block
RPZ	Risikoprioritätszahl
RTL	Register Transfer Level
Rx	Receiver, allg. bez. Empfangskomponente
SA	Stuck-at
SBST	Software Based Self Test
SC	Scan-Controller
SCM	Safety Concept Manager
SEC	Single Error Correction
SEDED	Single Error Correction, Double Error Detection
SEooC	Safety Element out of Context
SFF	Safe Failure Fraction
SFR	Scan Feed Register
SIL	Safety Integrity Level according to IEC61508 standard
SMART	SRAM Memory Automatic Repair Toolbox

SNR	Signal-Rausch-Verhältnis
SoC	System on Chip
SOPC	System On a Programmable Chip
SPFM	Single Point Faults Metric
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SRS	Software Requirement Specification
STIL	Standard Test Interface Language
SW	Software
TAP	Test Access Port (JTAG)
TC	TriCore
TCK	Test Clock (JTAG)
TCU	Transmission Control Unit
TDI	Test Data Input (JTAG)
TDL	Texas Instruments Test Description Language
TDMA	Time Division Multiple Access
TDO	Test Data Output (JTAG)
THD	gesamte harmonische Verzerrung
TL	Testlänge
TMR	Triple Modular Redundancy
TMS	Test Mode Select (JTAG)
TP	Test Prozessor
TUE	gesamter nichtangepasster Fehler
Tx	Transmitter, allg. bez. Senderkomponente
USB	Universal Serial Bus
USIF	Universal Scan-Test Interface
USL	Upper Specification Limit
UTL	Upper Test Limit
VDA	Verband der Automobilindustrie
VHDL	Very High speed integrated circuits Description Language
VLIW	Very Long Instruction Word Processor
Y	Yield
$\Delta\Sigma$ -ADC	Delta-Sigma-Digital-Analog-Wandler
$\mu$ C	Microcontroller

## XII. Anhang

### XII.1. Veröffentlichungen und Konferenzbeiträge

- C. Gleichner, T. Koal, H. T. Vierhaus, "Effective Logic Self Repair Based on Extracted Logic Clusters", IEEE Conference on Signal Processing Algorithms, Architectures, Arrangements, and Applications 2010, Poznan, Polen, Sept. 2010
- T. Koal, D. Scheit, H. T. Vierhaus, "Schwachstellen und Engpässe bei Verfahren zur Fehler-Kompensation und Selbstreparatur für hochintegrierte Schaltungen", 4. GMM/GI/ITG- Fachtagung "Zuverlässigkeit und Entwurf", Kreuth, Sept. 2010
- A. Cook, M. Elm, H.-J. Wunderlich, U. Abelein, "Structural In-Field Diagnosis for Random Logic Circuits", 23. GI/GMM/ITG-Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen", Passau, Feb. 2011
- T. Koal, H. T. Vierhaus, "Optimal Spare Utilization for Reliability and Mean Lifetime Improvement for Logic Built-in Self Repair", 1<sup>4th</sup> IEEE Int. Symposium on Design and Diagnostics of Electronic Circuits and Systems, Cottbus, April 2011
- A. Cook, M. Elm, H.-J. Wunderlich, U. Abelein, "Structural In-Field Diagnosis for Random Logic Circuits", 16<sup>th</sup> IEEE European Test Symposium, Trondheim, Norwegen, Mai 2011
- A. Cook, S. Hellebrand, T. Indlekofer, H.-J. Wunderlich, "Robuster Selbsttest mit Diagnose", 5. GMM/GI/ITG-Fachtagung "Zuverlässigkeit und Entwurf", Hamburg, Sept. 2011
- T. Koal, H. T. Vierhaus, "Rekonfigurierbare Logik für Ausbeute-Optimierung und Verschleiß-Kompensation", 5. GMM/GI/ITG- Fachtagung "Zuverlässigkeit und Entwurf", Hamburg, Sept. 2011
- T. Koal, M. Schölzel, H. T. Vierhaus, "On the Feasibility of Built-in Self Repair for Logic Circuits", 26<sup>th</sup> IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, Okt. 2011
- A. Cook, S. Hellebrand, T. Indlekofer, H.-J. Wunderlich, "Diagnostic Test of Robust Circuits", 20<sup>th</sup> IEEE Asian Test Symposium, New Delhi, India, Nov. 2011.

- A. Cook, S. Hellebrand, H.-J. Wunderlich, "Eingebaute Selbstdiagnose mit zufälligen und deterministischen Mustern", 24. GI/GMM/ITG-Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen", Cottbus, Feb. 2012
- J. Al-Eryani, S. Sattler, "A Novel High-Speed Analog-to-Digital Converter for Low-Cost in-Situ Testing", 24. GI/GMM/ITG-Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen", Cottbus, Feb. 2012
- C. Gleichner, P. Engelke, R. Kothe, H. T. Vierhaus, "Anbindung scanbasierter Tests an Standard-Schnittstellen: Möglichkeiten und Grenzen", 24. GI/GMM/ITG-Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen", Cottbus, Feb. 2012
- A. Cook, S. Hellebrand, M. E. Imhof, A. Mumtaz, H.-J. Wunderlich, "Built-in Self-Diagnosis Targeting Arbitrary Defects with Partial Pseudo-Exhaustive Test", 13<sup>th</sup> IEEE Latin American Test Workshop, Quito, Ecuador, April 2012
- T. Koal, M. Ulbricht, H. T. Vierhaus, "Combining On-Line Fault Detection and Logic Self Repair", 15<sup>th</sup> IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems, Tallinn, Estland, April 2012
- L. Grobelny, "Digitale Fehlermodelle und ihre Leistungsfähigkeit in der Praxis", Dresdner Arbeitstagung Schaltungs- und Systementwurf, Dresden, Mai 2012
- M. Gulbins, W. Vermeiren und S. Redlich, "Fehlerdiagnose für Analog-Digital- und Digital-Analog-Wandler in einer Loop-Back-Struktur", Dresdner Arbeitstagung für Schaltungs- und Systementwurf, Dresden, Mai 2012
- A. Cook, S. Hellebrand, H.-J. Wunderlich, "Built-in Self-Diagnosis Exploiting Strong Diagnostic Windows in Mixed-Mode Test", 17<sup>th</sup> IEEE European Test Symposium, Annecy, Frankreich, Mai 2012
- T. Koal, M. Ulbricht, P. Engelke und H. T. Vierhaus, "Selbstreparatur für Logik-Baugruppen mit erweiterten Fähigkeiten für die Kompensation von Fertigungsfehlern und Frühausfällen", Dresdner Arbeitstagung für Schaltungs- und Systementwurf, Dresden, Mai 2012
- M. Eberl, M. Glaß, J. Teich und U. Abelein, "Considering Diagnosis Functionality during Automatic System-Level Design of Automotive Networks", 49<sup>th</sup> Design Automation Conference, San Francisco, USA, Juni 2012



- C. Gleichner, H. T. Vierhaus und P. Engelke, "Scan Based Tests Via Standard Interfaces", 15<sup>th</sup> Euromicro Conference on Digital System Design, Izmir, Türkei, Sept. 2012
- T. Koal, M. Ulbricht, P. Engelke und H. T. Vierhaus, "Logic Self Repair Architecture with Self Test Capabilities", 6. GMM/GI/ITG- Fachtagung "Zuverlässigkeit und Entwurf", Bremen, Sept. 2012
- M. Lindig, Th. Klotz, B. Straube, "A Graph-theoretical Reduction Technique for Prime Implicant Table", 10<sup>th</sup> International Workshop on Boolean Problems, Freiberg, Sept. 2012
- A. Cook, D. Ull, M. Elm, H.-J. Wunderlich, H. Randoll, S. Döhren, "Reuse of Structural Volume Test Methods for In-System Testing of Automotive ASICs", 21<sup>st</sup> IEEE Asian Test Symposium, Niigata, Japan, Nov. 2012
- Y. Can, S. Sattler, T. Hölzer, W. Spengler, "Testschaltung zur Messung von defekten Kondensatoren", 25. GI/ITG/GMM Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen", Dresden, Feb. 2013
- L. Grobelny, "Nutzung der vollständigen Beschreibung der Schaltungsstruktur für die automatische Testpatterngenerierung", 25. GI/ITG/GMM Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen", Dresden, Feb. 2013
- M. Gulbins, W. Vermeiren und S. Redlich, "Fehlerdiagnose für ADCs und DACs in einer Loop-Back-Struktur unter Einbeziehung von Parameterschwankungen", 25. GI/ITG/GMM Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen", Dresden, Feb. 2013
- T. Koal, M. Ulbricht, P. Engelke und H. T. Vierhaus, "Kombinierte On-Line-Fehlerkompensation und Selbstreparatur für Logik-Baugruppen", 25. GI/ITG/GMM Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen", Dresden, Feb. 2013
- F. Reimann, M. Glaß, J. Teich und U. Abelein, "Szenarienbasierte Integration von Diagnosefunktionalität in E/E Architekturen", 3. GMM-Fachtagung "Automotive meets Electronics", Dortmund, Feb. 2013
- T. Koal, M. Ulbricht, P. Engelke, H. T. Vierhaus, "On the Feasibility of Combining Fast Fault Detection and Self Repair for Logic Circuits", 16<sup>th</sup> IEEE Int. Symposium on Design and Diagnostics of Electronic Circuits and Systems, Karlovy Vary, Tschechische Republik, April 2013

- T. Koal, M. Ulbricht, H. T. Vierhaus: „Virtual TMR Schemes Combining Fault Tolerance and Self Repair“, Proc. Euromicro-Conference on Digital System Design (DSD 2013), Santander, Sept. 2013, IEEE CS Press
- T. Koal, M. Ulbricht, P. Engelke, H. T. Vierhaus: „Combining Fault Tolerance and Self Repair in a Virtual TMR Scheme“, 7. GMM/GI/ITG- Fachtagung „Zuverlässigkeit und Entwurf“, Dresden, Sept. 2013

### **Diplom- Bachelor- und Master-Arbeiten**

- Stefan Kulke: „Extraktion regulärer Module aus irregulären Gatter-Netzlisten auf der Basis struktureller Vorgaben“, Diplomarbeit (Informatik), März 2012
- Davide Dicorato: „Controller for re-configurable logic blocks with self-testing and self-repair functions“, University of Cagliari, Italien, Oktober 2012 (betreut an der BTU Cottbus, Lehrstuhl Technische Informatik)
- Stephanie Röder: „Analyse und Weiterentwicklung eines adaptiven Selbsttests für VLIW-Prozessoren“, Master-Arbeit, Studiengang Informations- und Medientechnik, März 2013, BTU Cottbus
- Domink Dentl: „CAN-Bus gestützte Fehlerdiagnose von integrierten Schaltkreisen in kommerziellen Getriebesteuerungsgeräten, Bachelorarbeit, Dezember 2012, Institut für Technische Informatik, Universität der Bundeswehr München
- Andreas Krüger: „JTAG-basierte Konfiguration und Fehlerdiagnose eines FPGAs für Automotive-Anwendungen“, Masterarbeit, Juli 2012, Institut für Technische Informatik, Universität der Bundeswehr München

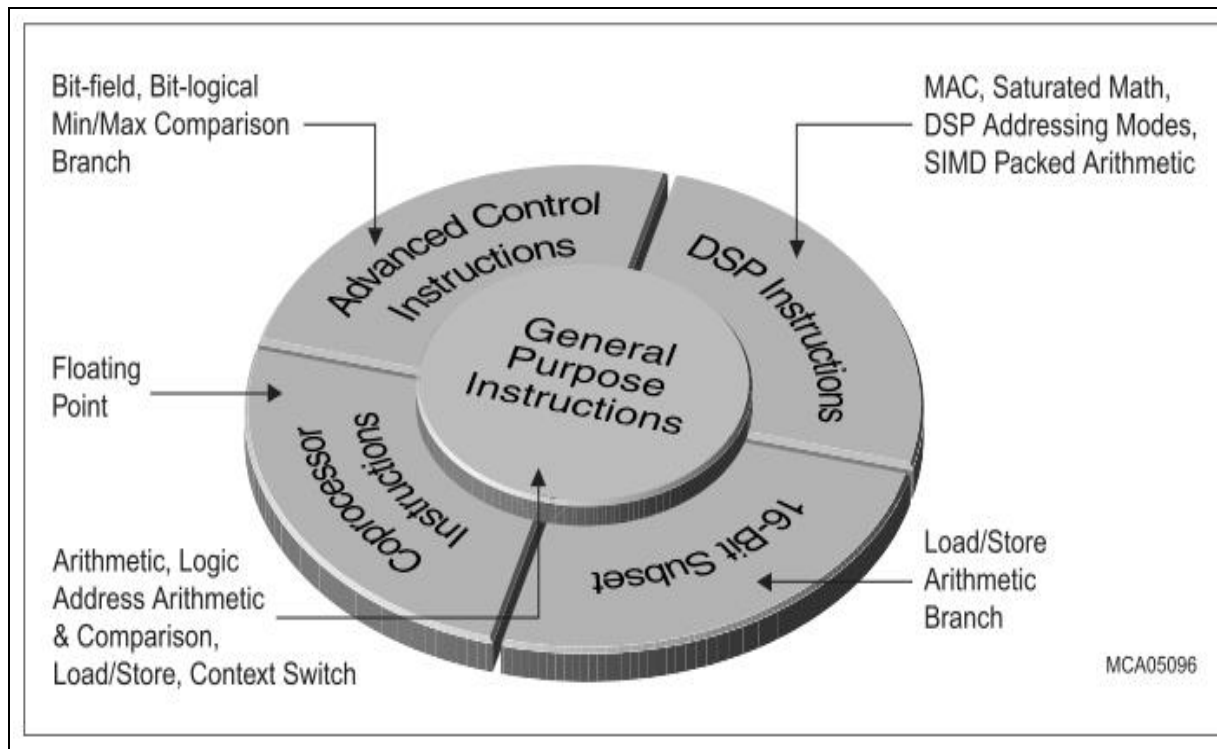
## XII.2. Patente und –anmeldungen

Titel	Ausfertigungsdatum
Selbstkalibrierender Signalgenerator	24.10.2011
Vorrichtung zum Converter-Test in Echtzeit	24.10.2011
Vorrichtung zur Berechnung von Offset und Gain für Converter in Echtzeit	18.01.2012
Non Destructive Test für SRAMs in Applikationen	23.05.2012
Prüfbitkompaktierung für Cross-Parity-Codes	05.12.2012
Verbesserung von Non Destructive Test for SRAMs in Applikationen	05.12.2012
Sichere Speicherung durch interne Betriebssicherstellung	09.04.2010
Schaltungsanordnung zum Online-Test	28.03.2011

## XII.3. Anhang: Einführung in die Architektur des verwendeten Microcontrollers

### XII.3.1. *TriCore*<sup>®</sup> *Microcontroller Architektur*

Die *TriCore*<sup>®</sup> Architektur ist die erste vereinheitlichte Single-Chip 32-bit Microcontroller DSP-Architektur, die für Echtzeit Anwendungen in Embedded Systems optimiert wurde. Die *TriCore* Befehlssatz Architektur (ISA) kombiniert die Echtzeit-Fähigkeit eines Microcontrollers (MC) mit der Rechenleistung eines Digitalen Signal Prozessors (DSP) und dem guten Preis- / Leistungs-Verhältnis eines Reduced Instruction Set Computers (RISC) in einem kompakten programmierbaren Rechenkern.



Architektur eines TriCore Microcontrollers

Der Befehlssatz unterstützt einen durchgehenden 32-bit Adressraum von 4 GByte mit optionaler virtueller Adressierung und memory-mapped I/O-Strukturen. Die Architektur erlaubt einen weiten Bereich von Anwendungen, von skalaren bis superskalaren Applikationen. Sie erlaubt die Implementierung von verschiedenen System-Architekturen einschließlich Multi-Processing und hat somit ein hohes Maß an Flexibilität in der Praxis.

Die Architektur unterstützt 16-bit und 32-bit Befehlsformate, die 16-bit Befehle sind eine Untermenge der 32-bit Befehle, ausgewählt nach der Häufigkeit ihrer Anwendung. Sie reduzieren den Speicherbedarf von Programmcode und Arbeitsspeicher, sie erhöhen die Rechenleistung und reduzieren den Energieverbrauch des Microcontrollers.

Das Echtzeit-Verhalten wird weitgehend bestimmt durch die Reaktionszeit auf Interrupts und auf Taskwechsel. Die TriCore<sup>®</sup> Architektur minimiert die Interrupt Reaktionszeit durch die Vermeidung von langen Multi-Zyklus Befehlen. Die meisten Befehle werden in einem Takt ausgeführt. Der Einbau einer flexiblen hardware-unterstützten Interrupt-Struktur unterstützt auch schnelles Umschalten zwischen verschiedenen Anwendungen.

Die ausgewählte TriCore<sup>®</sup> Architektur ist für die nachfolgend beschriebene Entwicklung und den praktischen Nachweis der Fehlerabdeckung (DC) für ein FSBST Testprogramm geeignet.

### **XII.3.2. Architektur des TriCore<sup>®</sup> Microcontrollers TC1387**

Der TC1387 kombiniert drei leistungsfähige Technologien auf einem Silizium Chip, wodurch er neue Bereiche der Rechenleistung, der Geschwindigkeit und der Wirtschaftlichkeit im Bereich der embedded systems erreicht.

- Reduced Instruction Set Computer (RISC) Architektur
- Digitaler Signal Prozessor (DSP) Befehle und Adressierungsarten
- On-Chip Programmspeicher und Arbeitsspeicher und Peripherie Einheiten

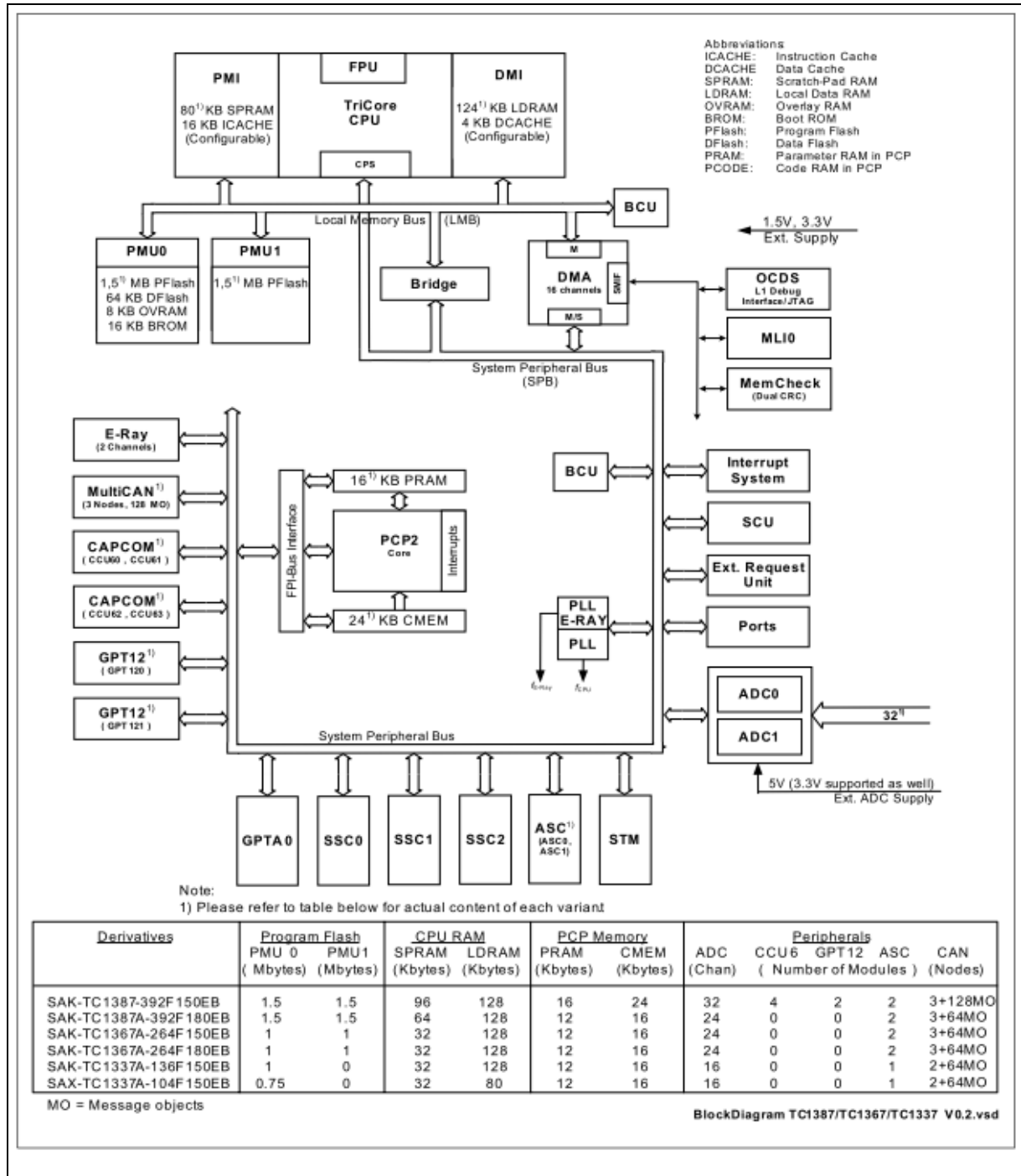
Die DSP Befehle und Adressierungsarten bieten die erforderliche Rechenleistung, um effizient komplexe Signale aus der realen (physikalischen) Welt zu analysieren. Die RISC Architektur mit seiner Laden / Speichern Fähigkeiten bietet hohe Rechenbandbreite bei geringen Systemkosten. Die on-Chip Programmspeicher und Arbeitsspeicher sind so entwickelt und dimensioniert, um selbst den höchsten Anforderungen an Echtzeit Rechenleistung in embedded systems zu genügen. Zusätzliche Hochleistungs-Funktionen des TC1387 beinhalten folgende Komponenten:

- Effiziente Speicherorganisation: Zwischenregister und Caches für Befehle und Daten
- Serielle Kommunikation – flexible synchrone und asynchrone Betriebsarten
- Peripheral Control Processor – selbständige Datenbefehle und Interrupt Behandlung
- DMA Controller – DMA Befehle und Interrupt Behandlung
- Universal Timer
- Analog-Digital-Wandler
- Hochleistungs-Bus on-Chip
- on-Chip debugging und Emulations Möglichkeiten
- Flexible Verbindungen zu externen Komponenten
- Flexibles Verlustleistungs-Management

Der TC1387 ist ein Microcontroller mit einer TriCore CPU, Programmspeicher und Datenspeicher, Daten- und Adress-Bussen, Bus-Verwaltung und Bus-Steuerung, einem Interrupt-Controller, einem Peripheral Control Processor (PCP), einem DMA Controller und verschiedenen Peripherie-Elementen. Innerhalb des TC1387 sind alle Peripherie-Elemente mit der TriCore CPU über den Flexible Peripheral Interconnect (FPI) Bus und den Local Memory Bus (LMB) verbunden. Mehrere I/O Leitungen an den TC1387 Ports sind reserviert für diese Peripherie Elemente, um mit der externen Welt zu kommunizieren.

### Blockdiagramm des TriCore<sup>®</sup> Microcontrollers TC1387

Die folgende Abbildung zeigt das Block-Diagramm des TC1387 Microcontrollers



Blockdiagramm des TC1387

## Berichtsblatt

1. ISBN oder ISSN keine	2. Berichtsart (Schlussbericht oder Veröffentlichung) Schlussbericht
3. Titel Schlussbericht zum BMBF-Verbundprojekt „DIANA“ (Durchgängige Diagnosefähigkeit in Halbleiterbauelementen und übergeordneten Systemen zur ANALyse von permanenten und sporadischen Elektronikausfällen im Gesamtsystem Automobil)	
4. Autor(en) [Name(n), Vorname(n)] Obermeir Hermann; Grobelny, Lothar; Abelein, Ulrich; Hoelzer, Thomas	5. Abschlussdatum des Vorhabens 31.05.2013
	6. Veröffentlichungsdatum 12.12.2013
	7. Form der Publikation Broschüre
8. Durchführende Institution(en) (Name, Adresse) 1) Infineon Technologies AG, 81726 München (Großkunden-PLZ ohne Str.) 2) Audi AG, 85045 Ingolstadt (Großkunden-PLZ ohne Str.) 3) Continental AG, Sieboldstrasse 19, 90411 Nürnberg 4) Zentrum Mikroelektronik Dresden AG, Grenzstrasse 28, 01109 Dresden	9. Ber. Nr. Durchführende Institution nicht zutreffend
	10. Förderkennzeichen 16M3188 (VDI)
	11. Seitenzahl 294
12. Fördernde Institution (Name, Adresse)  <b>Bundesministerium für Bildung und Forschung (BMBF) 53170 Bonn</b>	13. Literaturangaben 42
	14. Tabellen 56
	15. Abbildungen 83
16. Zusätzliche Angaben keine	
17. Vorgelegt bei (Titel, Ort, Datum) TIB Hannover	
18. Kurzfassung <p>Für den potentiellen Käufer eines Automobils spielt heute, neben den zu erwartenden Kosten und vielen anderen Faktoren, die Zuverlässigkeit eine maßgebliche Rolle bei der Kaufentscheidung. Kosten und Zuverlässigkeit werden wiederum ganz wesentlich durch die im Fahrzeug verwendeten elektronischen Systeme bestimmt. Nicht zuletzt durch gestiegene Sicherheits- und Umweltschutzanforderungen hat die Komplexität elektronischer Systeme in den vergangenen Jahren immer weiter zugenommen. Die eindeutige Identifikation fehlerverursachender Komponenten wird jedoch in solchen komplexen, verteilten Systemen immer schwieriger, ist aber gleichzeitig essentiell für hohe Qualität und sichere Systemverfügbarkeit.</p> <p>Im Rahmen des BMBF-Verbundprojekts DIANA wurden erweiterte Diagnosefähigkeiten in Halbleiterbauelementen, Steuergeräten und dem Gesamtsystem der Automobilelektronik erforscht, um permanent oder sporadisch auftretende Fehler im Automobil schneller und zielgerichteter auffinden zu können. Dabei wurden Methoden und Verfahren entwickelt, die eine effiziente und unter den Rahmenbedingungen unterschiedlicher Anwendungsfälle wirtschaftliche Diagnose solcher Fehler erlauben. Diese Lösungen zur Fehlerdiagnose von hochkomplexen Steuergeräten der Automobilelektronik wurden so gestaltet, dass sie beginnend mit der Fehlermodellierung, über die Schaltungsentwicklung sowie den Produktionstest beim Halbleiterhersteller, und über die Umsetzung der in den Halbleiterbauelementen integrierten Verfahren beim Steuergerätehersteller, schließlich beim Endanwender zur Diagnose und Fehlerbeseitigung eingesetzt werden können. Dank dieses durchgängigen Ansatzes über alle Glieder der Wertschöpfungskette, bereitet DIANA den Weg für eine übergreifende Diagnosemethodik für die Automobilelektronik. Zu den für diese Diagnosemethodik essentiellen Ergebnissen des Projektes, gehören unter anderem Verfahren, die unter Berücksichtigung der Schaltungsstruktur im System eine exakte Diagnose von Fehlern in digitaler Logik sowie in Flash- und SRAM-Speichern ermöglichen. Auch für die Diagnose von Fehlern in gängigen analogen Komponenten, wie beispielsweise Analog-Digital-Wandler, konnten vollkommen neue Verfahren erarbeitet werden.</p>	
19. Schlagwörter Diagnose, Design for Test	
20. Verlag keiner	21. Preis keiner

## Document Control Sheet

1. ISBN or ISSN none	2. type of document (e.g. report, publication) project final report
3. title project final report for BMBF funded project „DIANA“ (full German title: Durchgängige Diagnosefähigkeit in Halbleiterbauelementen und übergeordneten Systemen zur ANALyse von permanenten und sporadischen Elektronikausfällen im Gesamtsystem Automobil)	
4. author(s) (family name, first name(s)) Obermeir Hermann; Grobelny, Lothar; Abelein, Ulrich; Hoelzer, Thomas	5. end of project May 31st, 2013
	6. publication date Dec 12th, 2013
	7. form of publication brochure
8. performing organization(s) (name, address) 1) Infineon Technologies AG, 81726 München 2) Audi AG, 85045 Ingolstadt 3) Continental AG, Sieboldstrasse 19, 90411 Nürnberg 4) Zentrum Mikroelektronik Dresden AG, Grenzstrasse 28, 01109 Dresden	9. originator's report no. none
	10. reference no. 16M3188 (VDI)
	11. no. of pages 294
12. sponsoring agency (name, address)  <b>Bundesministerium für Bildung und Forschung (BMBF) 53170 Bonn</b>	13. no. of references 42
	14. no. of tables 56
	15. no. of figures 83
16. supplementary notes none	
17. presented at (title, place, date) TIB Hannover	
18. abstract For todays cars, not only the selling price will influence the purchase decision. Reliability and longevity are becoming major arguments. Both depend on the quality of electronic components used in the car. DIANA aims on extending diagnosability abilities to address the complexity of electronic systems used in cars. Special focus has been set on extending 'classical' diagnosis during system startup phase to normal operation. To be able to do so, new models and methods have been developed to allow exchange of diagnosis information between all levels of the system.	
19. keywords Diagnose, Design for Test	
20. publisher none	21. price none