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

Robert Bosch GmbH
EPQ & EHM
Wernerstrasse 1
D–70469 Stuttgart-Feuerbach

Diplomarbeit Nr. 3146

Strukturelle Feldtests komplexer ASICs

Dominik Ull

Studiengang:	Informatik
Prüfer:	Prof. Dr. Hans-Joachim Wunderlich
Betreuer:	Dipl.-Inform. Melanie Elm M. Sc. Alejandro Cook Dipl.-Ing. Stefan Döhren Dr. rer. nat. Helmut Randoll
begonnen am:	10. Januar 2011
beendet am:	9. August 2011
CR-Klassifikation:	B.1.3, B.8.11

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kurzfassung

In dieser Ausarbeitung wird ein zerstörungsfreier Befundungstest für Kfz-Steuergeräte vorgestellt.

Hersteller von Automobilelektronik können bisher nicht nachweisen, dass ein an den Kfz-Hersteller ausgeliefertes Steuergerät wirklich fehlerfrei ist, obwohl zur Minimierung des Testaufwands bei der Produktion der enthaltenen ASICs (Application Specific ICs) schon während der Entwicklungsphase des Chipdesigns strukturelle, standardisierte Testmethoden und eingebaute Selbsttests (BIST, Built-In Self-Test) integriert werden.

Es soll nun beispielhaft an einem ASIC aufgezeigt werden, in wie weit diese Methoden beim strukturellen Feldtest von Automobil-Steuergeräten – im Rahmen eines zerstörungsfreien Befundungstests von Feldrückläufern – Verwendung finden. Der Test ermöglicht neben der Klärung teurer Garantieansprüche auch eine verlässliche Informationsquelle für ein eventuelles Redesign.

Durch die Implementierung des JTAG-Protokolls auf den Signalleitungen des steuergerät-internen SPI-Bus können ASIC-interne Selbsttests für Speicher und Analog-Digital-Converter (ADC) im verbauten Zustand ausgeführt werden. Die softwarebasierte Anwendung von Scan Patterns auf Steuergerät-Ebene ermöglicht einen einfachen Scan Test für ASICs ohne Logik-BIST. Es folgt ein Realisierungsvorschlag zur Prüfung aller steuergerät-internen Taktquellen, um die durch Kombination von BIST und softwarebasierten Methoden erreichbare Testabdeckung aufzuzeigen.

Keywords: Befundungstest, Feldtest, Steuergerät, Application Specific Integrated Circuit (ASIC), Built-In Self-Test (BIST), Softwarebasierter Scan-Test

Abkürzungsverzeichnis

ABS Antiblockiersystem
ADC Analog-Digital-Converter
ASIC Application-Specific Integrated Circuit
ATE Automated Test Equipment
ATPG Automatic Test Pattern Generator
BIST Built-In Self-Test, eingebauter Selbsttest
BR Bypass Register
BSC Boundary Scan Cell
BSDL Boundary Scan Description Language
BSR Boundary Scan Register
CANBUS Controller Area Network Bus
CF Coupling Fault
CFI CarPu-Flash-Interface, Boch-intern
CS Chipselect, SPI-Signal
CUT Component-Under-Test
DAC Digital-Analog-Converter
DFT Design For Test
DMA Direct Memory Access
DR Data Register
DSP Digital Signal Processor
DUT Device Under Test
ECK Engineering Clock
EEPROM Electrically Erasable Programmable Read-Only Memory
ESP Elektronisches Stabilitätsprogramm
GPS Global Positioning System
GSM Global System for Mobile Communications
IC Integrated Circuit
ICP In-Circuit Programming
I/O Input / Output
IR Instruction Register
JTAG Joint Test Action Group, reference to IEEE 1149.1 standard
Kfz Kraftfahrzeug
kgV kleinstes gemeinsames Vielfaches (zweier natürlicher Zahlen)
LFSR Linear-Feedback-Shift-Register
MCU Micro Controller Unit
MISO, SO Master-In-Slave-Out, SPI-Signal
MISR Multiple-Input-Signature-Register
MOSI, SI Master-Out-Slave-In, SPI-Signal
MPU Micro Processing Unit
MSC-Bus Micro-Second-Bus, Bosch-internes Interface
NOP No Operation

PLL Phase Locked Loop
PWM Pulsweitenmodulation
RAM Random Access Memory
ROM Read-Only Memory
RST Reset
SAF Stuck-At Fault
SBST Software-Based Self-Test
SCLK, SCK Serial Clock, SPI-Signal
SPI Serial Peripheral Interface
SRAM Static Random Access Memory
TAP Test Access Port
TCK Test Clock
TDI Test Data Input
TDO Test Data Output
TF Transition Fault
TMS Test Mode Select
TRST Test Reset
USB Universal Serial Bus
VHDL Very High Speed Integrated Circuit Description Language
XOR Exklusives Oder

1 Einleitung

> Es war mir immer ein unerträglicher Gedanke, es könne jemand bei Prüfung eines meiner Erzeugnisse nachweisen, dass ich irgendwie Minderwertiges leiste. Deshalb habe ich stets versucht, nur Arbeit hinauszugeben, die jeder fachlichen Prüfung standhielt, also sozusagen vom Besten das Beste war.
>
> *(Robert Bosch, 1918, Grundsätze)*

1.1 Motivation

Seit Jahrzehnten wächst der Anteil von Elektronik im Kraftfahrzeug (Kfz) stetig an, trotz kurzer Stagnation infolge der Automobilkrise 2008 (vgl. Abbildung 1.1). Dieses Wachstum geht im Automobilbereich mit erhöhten Anforderungen an den Einsatzbereich einher. Im Gegensatz zu typischer Konsumelektronik müssen die Komponenten des Kfz in einem stark schwankenden Temperaturbereich (-40 bis +125 C) unter hohem mechanischem Stress wie Vibrationen beim Fahrbetrieb stets fehlerfrei arbeiten. Nur unter höchsten Qualitätsansprüchen (Zero Defect Requirement) kann der Einsatz von Elektronik im Kfz als sicher eingestuft werden. Außerdem unterliegt die Kfz-Elektronik auch den allgemeinen Trends hin zur Miniaturisierung und Integration von (applikationsspezifischen) Schaltkreisen. Neben dem steigenden Elektronikanteil im Kfz hat sich die Integrationsdichte (Gatter pro Fläche) gemäß dem Mooreschen Gesetz in den letzten drei Jahren verdoppelt.

All diese Entwicklungen unterstreichen die Notwendigkeit verlässlicher Testmethoden, die sowohl während der Produktionskette als auch in Feldtests an "Rückläufern“ angewandt werden können. Hierbei unterscheidet man grundsätzlich je nach Intention des Tests zwischen funktionalen Testmethoden, bei denen die Funktion der zu testenden Komponente im Vordergrund steht, und strukturellen Testmethoden, bei der die zugrundeliegenden Strukturen der Komponente überprüft werden. Allgemein zieht man bei elektronischen Komponenten im Sinne einer höheren Testabdeckung und geringeren Testzeit den strukturellen Test vor.

Im Laufe der ASIC-Entwicklung werden heutzutage bereits frühzeitig Teststrukturen in das Design integriert, um den späteren, finanziell schwerwiegenden Testaufwand am Ende der eigentlichen Produktion zu mindern. Im Zuge des Design-For-Test-Prinzips werden standardisierte Testschnittstellen (JTAG) und eingebaute Selbsttests (BIST, Built-In Self-Test) eingesetzt [SC00]. Mithilfe dieser hardwarebasierten, strukturellen Testmethoden wird bei annehmbarem Overhead in relativ kurzer Testzeit eine hohe Abdeckung erzielt.

Auf Steuergerät-Ebene existieren keine dedizierten Teststrukturen, trotzdem lassen sich hier programmierbare Komponenten zum Testen verwenden. Produktionstest beschränken sich bisher auf eine funktionale Prüfung von Leiterbahnen, Lötstellen und diskreten analogen Bauteilen bei unterschiedlichen Temperaturen. Die verbauten ASICs werden an dieser Stelle oft als schon geprüft und fehlerfrei angesehen. Trotz ASIC-interner Teststrukturen wurde eine Befundung von als fehlerhaft deklarierten Steuergeräten bisher nicht automatisiert. Insbesondere bei Verdacht auf defekte ASICs werden diese zwecks Befundung unter Einsatz von Hitze aus dem Steuergerät ausgelötet und an den entsprechenden Zulieferer gesandt. Dieses Vorgehen ist weder zerstörungsfrei noch zur Befundung geeignet, da so neue Defekte aktiviert oder aber bestehende Defekte entfernt werden können.

Beim Einsatz von Run-In-Strategien (stundenweiser Vorbetrieb des Produkts am Ende der Herstellung) können Frühausfälle erfolgreich aktiviert werden. Das anschließende Erkennen der aktivierten Defekte ist auf schnelle Tests mit hoher Abdeckung — also strukturelle Tests — angewiesen. Auf diese Weise können die strukturellen Testmethoden auch zur Qualitätssteigerung in der Steuergeräte-Produktion eingesetzt werden.

Ziel dieser Ausarbeitung ist es, die Einsatzmöglichkeiten der ASIC-internen Teststrukturen auf Steuergerät-Ebene zur zerstörungsfreien Befundung von Feldrückläufern zu evaluieren. Dies erlaubt die Abweisung teurer Garantieforderungen und ebnet den Weg für den Einsatz der strukturellen Testmethoden in allen Testbereichen, vom Run-In in der Produktion bis hin zum Feldtest in der Werkstatt.

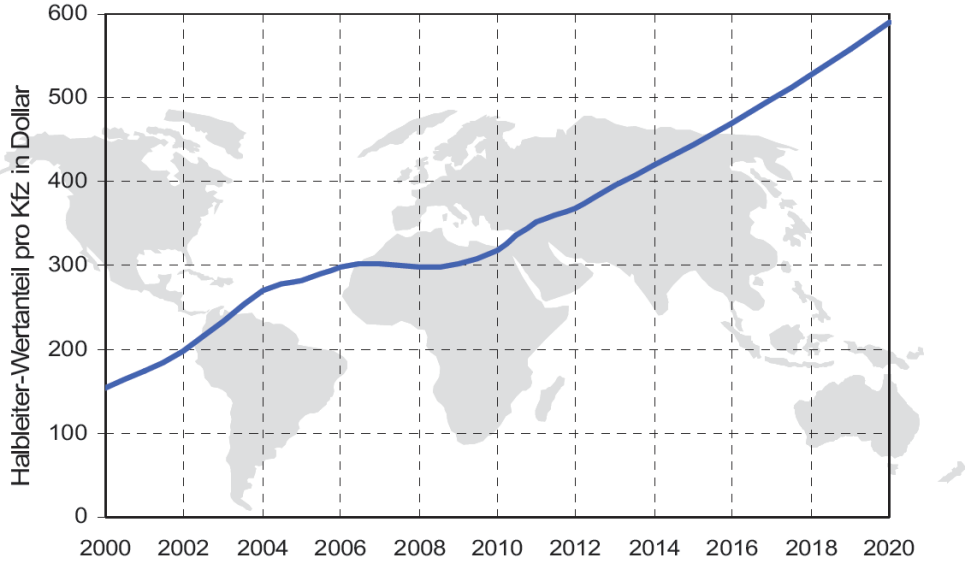


Abbildung 1.1: Halbleiter-Wertanteil im Kfz, Quelle: ZVEI Mikroelektroniktrendanalyse bis 2013 [ZVE10]

1.2 Überblick

Zunächst erklärt Kapitel 2 die Grundlagen der Thematik. Hierzu gehören Defektarten und Fehlermodellierung (Kapitel 2.1), Testbereiche (Kapitel 2.2) und die Funktionsweise von Automobil-Steuergeräten (Kapitel 2.3). Es folgt eine Vorstellung sämtlicher zur Verfügung stehender Teststrukturen auf Steuergerät- (Kapitel 2.4) und ASIC-Ebene (Kapitel 2.5).

Kapitel 3 enthält die Implementierungen dieser Arbeit. Nach einer Beschreibung des Testaufbaus (Kapitel 3.2) folgt die Durchführung von JTAG-Testkommunikation auf den Signalleitungen des steuergerät-internen SPI-Bus (Kapitel 3.3), die softwarebasierte Anwendung von Scan-Patterns (Kapitel 3.4), ein BIST für Taktsignale (Kapitel 3.5) sowie ein softwarebasierter Abgleich weiterer Taktquellen (Kapitel 3.6). Der Befundungstest (Kapitel 3.7) umfasst zunächst eine funktionalen Vortest zur Validierung des Testaufbaus (Kapitel 3.7.1), um dann die vorherigen Implementierungen in einem Testplan (Kapitel 3.7.2) zusammenzufassen.

Kapitel 4 enthält die Ergebnisse der Implementierung und führt die erreichte Testabdeckung und benötigten Testzeiten beim Befundungstest auf. Die Ausarbeitung schließt mit einem Fazit zu den Anwendungsmöglichkeiten der eingesetzten Testmethoden (Kapitel 5) sowie einem Ausblick auf zukünftige Entwicklungen (Kapitel 6).

2 Grundlagen

2.1 Auftretende Defekte & Fehlermodellierung

Defekt, Fehler und Fehlfunktion

Defekte sind ungewollte Abweichungen zwischen Hardwareimplementierung und urspünglichem Design [MLBa]. Sie werden gemäß Tabelle 2.1 unterteilt [MJH81] und können alle Systemebenen betreffen:

Defekt-Art	ASIC	Steuergerät
Prozessdefekt	Partikel in den Produktionshallen, fehlerhafte Masken-Ausrichtung, zu dünne Isolationsschichten, Bruch des Oxids	Kurzschlüsse, kalte Lötstellen,
Materialdefekt	Risse/Unreinheiten im Silizium, Kristalldefekte	abgebrochene IC-Pins
Alterungsdefekt	Bruch des Dielektrikums	auslaufender Elektrolytkondensator
Verpackungsdefekt	Bond-Defekte, Hohlräume/Risse im Verpackungsmaterial	undichtes Gehäuse, fehlender Aufkleber mit Typbezeichnung

Tabelle 2.1: Defektklassen & Beispiele

Ein Defekt wird durch ein Fehlermodell (engl. Fault) repräsentiert. Er kann — unter gewissen Umständen — zu einem fehlerhaften Systemzustand führen (engl. error), welcher ihn beobachtbar macht. Infolgedessen kann es letztendlich auch zu einer Fehlfunktion (engl. failure) und somit zu einem Ausfall kommen. Ob ein fehlerhafter Zustand zu einer vom Kunden bemerkbaren Fehlfunktion führt, ist systemabhängig. Als Beispiel soll hier ein einfaches Oder-Gatter (Eingänge e_0, e_1; Ausgang a_0) als Gesamtsystem dienen. Das Eingangsignal e_0 hat einen Kurzschluss nach Ground, ist also permanent defekt. Dies wird auf Systemebene als Stuck-At-0 Fault modelliert, d.h. das Signal e_0 hält konstant den Wert 0. Zu einem fehlerhaften Systemzustand kann es hier allerdings nur kommen, wenn das defekte Signal e_0 den Ausgang des Oder-Gatters kontrolliert, d.h. wenn $e_2 = 0$ ist. Erst der Eingangsvektor (1,0) würde hier die Fehlfunktion am Ausgang des Oder-Gatters sichtbar machen. Ein Test zum Erkennen des Fehlers müsste genau diesen Testvektor enthalten.

Fehlermodelle

Eine Testmethode wird nach der benötigten Testzeit und der erreichten Testabdeckung beurteilt. Hierbei bezieht sich die Abdeckung auf ein zugrundeliegendes Fehlermodell; sie stellt dar, wie viele Fehler im Bezug auf alle (nach dem Modell) möglichen Fehler durch den Test erkannt werden. Anzahl und Art der modellierten Fehler hängen von der zu testenden Komponente sowie den Testanforderungen ab. Das klassische Fehlermodell auf Logikebene ist das statische Stuck-At-Modell, welches die beiden Fehlerklassen Stuck-At-0 und Stuck-At-1 umfasst. Eine betroffene Signalleitung ist — unabhängig vom Betriebsmodus der untersuchten Schaltung — statisch auf 0 oder 1 gesetzt [JTA90]. In der Praxis wird dies durch Defekte wie ungewollte, niederohmige Verbindungen zwischen Signal und Versorgungsspannung oder Erdung erzeugt. Bei der Komplexität moderner ICs (Integrated Circuits) ist dieses Modell jedoch nicht mehr ausreichend, um alle auftretenden Fehler modellieren zu können [JLB07]. Es wird daher oft um Schaltverzögerungen (Transition Delay Faults) erweitert, welche während des Betriebs Glitches und Hazards erzeugen können [SM09]. Zur Prüfung solcher Fehler muss At-Speed getestet werden [IP08], d.h. der zeitliche Verlauf der Testsignale muss mindestens mit gleicher Schaltfrequenz wie im Normalbetrieb beobachtet werden.

Das Fehlermodell wird je nach zu testender Komponente angepasst. Bei integrierten Speichern erweitert man die klassischen Stuck-At-Faults (SAFs) um Transition Faults (TFs), welche einen Zustandswechsel der Bit-Zellen zwischen 0 und 1 (oder 1 und 0) verhindern, um Adress-Decoder-Faults (AFs), die sich als fehlerhafte Adressierung von Wortzellen bemerkbar machen, und um Coupling Faults (CFs), bei denen eine Speicherzelle den Inhalt einer weiteren (benachbarten) Zelle beeinflusst [JFL01].

Durch eine gute Testabdeckung bezogen auf das Stuck-At-Modell kann ein Großteil der auftretenden Defekte bereits erkannt werden. Dennoch muss zur Erkennung seltener, z.B. durch Transition-Delay-Faults modellierter Defekte ein ungleich größerer Testaufwand betrieben werden.

Funktionales / Strukturelles Testen

Testmethoden werden als funktional oder strukturell bezeichnet, je nachdem ob funktionales oder strukturelles Wissen über die zu testende Komponente benutzt wird. Der Test gibt demnach Auskunft über die Funktion oder die Struktur. Im Hinblick auf Testabdeckung und vor allem Testzeit sind im Bereich von Hardware-Tests strukturelle Methoden gegenüber den funktionalen vorzuziehen. Letztere werden daher eher für eine Designverifikation und seltener für einen Produkttest eingesetzt. Dass funktionale Tests bei Digitalelektronik sehr zeitaufwändig sein können zeigt folgendes Beispiel: Wenn ein 64-bit-breiter Zähler funktional geprüft werden soll, müssten dessen Ausgänge in 2^{64} verschiedenen funktionalen Zuständen beobachtet werden. Erst nach Prüfung der gesamten Wahrheitstabelle ist hier die korrekte Funktionalität aufgezeigt. Ein struktureller Test dagegen kann den Zähler in sehr viel weniger Testzuständen auf die modellierten Strukturfehler der einzelnen Flipflops hin prüfen. Das verwendete Fehlermodell kann beim funktionalen Testen immer nur auf funktionaler Systemebene definiert sein. Erst das strukturelle Testen erlaubt eine hardware-orientierte Modellierung, beispielsweise mit Stuck-At-Faults.

Ausfallwahrscheinlichkeiten & Burn-In-Verfahren

Die Zuverlässigkeit eines Produkts lässt sich anhand der Ausfallwahrscheinlichkeit in Abhängigkeit von der Lebensdauer messen. Häufig wird dieser Zusammenhang als sogenannte „Badewannenkurve“ modelliert. Dabei werden drei statistische Weibull-Verteilungen überlagert, die jeweils die Frühausfälle, die konstanten Ausfälle im Laufe der vorgesehenen Lebensdauer sowie die altersbedingten Fehlerfälle darstellen. Die allgemeine Weibull-Verteilung ist durch folgende Dichtefunktion definiert[DK98]:

$$f(t) = \frac{b}{n} \cdot \left(\frac{t}{n}\right)^{b-1} \cdot \exp\left(-\left(\frac{t}{n}\right)^{b}\right)$$

Hierbei dient die Variable n dem Strecken der Funktion auf der X-Achse (Lebensdauer), Variable b legt die Form der Kurve fest. Die frühzeitigen Ausfälle (f1, $b = 0.1$) werden durch eine ab dem Zeitpunkt t=o stark sinkende Auftrittswahrscheinlichkeit modelliert. Je länger ein Produkt funktioniert desto unwahrscheinlicher wird der Ausfall. Die konstante Ausfallrate (f2, $b = 1$) ist fast unabhängig vom Auftrittszeitpunkt und kann als nahezu konstant über die Lebensdauer angesehen werden. Altersbedingte Ausfälle (f3, $b > 1$) häufen sich gegen Ende der Lebensdauer: Je älter ein Produkt ist, desto wahrscheinlicher wird ein derartiger Ausfall. Bild 2.1 zeigt die drei parametrisierten Weibull-Verteilungen ($n = 1$, $b = 0.1/1/10$) sowie deren Summe (f4), welche den typischen Verlauf der sogenannten der Produktausfälle zeigt. In der Anwendung dieses Modells werden die einzelnen Parameter statistisch erhoben.

Anhand dieser Verteilung lässt sich gut verdeutlichen, dass ein kurzzeitiger Betrieb des Produkts am Ende der Herstellungskette den Großteil der Frühausfälle abfangen kann [Cha06]. Ein derartiges Vorgehen wird im ASIC-Bereich als Burn-In bezeichnet, während man auf Steuergerät-Ebene vom sogenannten Run-In spricht. Der große Mehraufwand des stundenweisen Vorbetriebs mit extern zugeschalteten Lasten lohnt natürlich nur bei vollständigen Testmethoden, die beim Run-in aktivierte Defekte noch vor der Auslieferung detektieren können. Dies kann auf ASIC-Ebene mit in Kapitel 2.5.3 beschriebenen BIST-Methoden bzw. auf Steuergerät-Ebene mit einem Boundary Scan Test (vgl. Kapitel 2.5.1) erfolgen.

Im Lebenszyklus eines Steuergeräts werden mehrmals unterschiedliche Komponenten aus unterschiedlichen Motiven heraus getestet. Bild 3.1 markiert die hier interessanten Testbereiche unterteilt in strukturelle und funktionale Testmethoden.

2.2 Testbereiche

ASIC-Produktion

Zunächst einmal werden die ASICs als eigenständiges Produkt schon während ihrer Fertigung geprüft (A). Dies geschieht sowohl am offenen Silizium als auch am verpackten Bauelement. Die digitalen

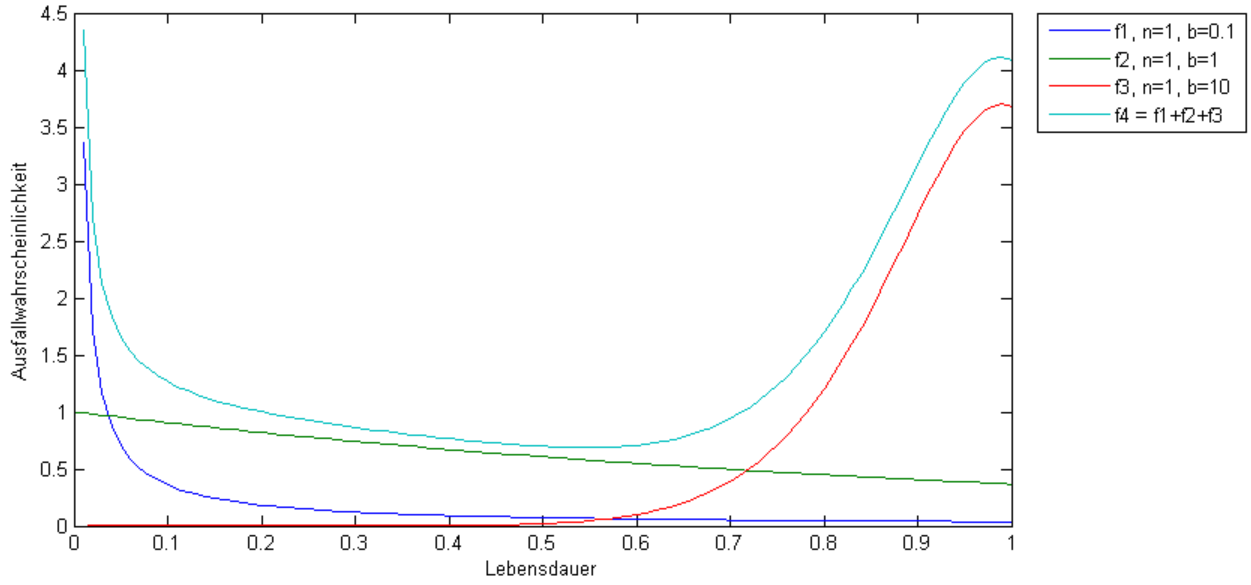


Abbildung 2.1: zeitlich verteilte Ausfallwahrscheinlichkeit ("Badewannenkurve")

Schaltkreise können hier schnell und genau strukturell getestet werden, während analoge Anteile meist mithilfe externer Beschaltungen einer aufwändigen funktionalen Prüfung unterzogen werden. Die digitalen Bereiche sind aufgrund der starken Miniaturisierung erfahrungsgemäß anfälliger für Defekte, welche schon durch Staubpartikel in den Produktionshallen entstehen können.

Steuergeräte-Produktion

Bei der anschließenden Steuergeräte-Herstellung liegt das Hauptaugenmerk auf funktionaler Komponenten- und Leitungsprüfung. Schon beim Platzieren eines Widerstands prüft der Greifarm dessen Abweichung vom Sollwert. ICs werden durch visuelle Prüfung der Beschriftung abgeglichen. Am Ende der Produktion kommt dann ein aufwändiger, programmierbarer funktionaler Tester nach der Prüfablaufvorschrift (PAV) zum Einsatz (B). Ziel dieser Prüfung ist die Verifizierung, ob Verbindungen zwischen den einzelnen Schaltmodulen des Steuergeräts fehlerfrei ausgeführt sind. Die PAV beschränkt sich hierbei auf die Durchgangsprüfung (Verbindungstest) in unterschiedlichen Temperaturbereichen.

Der Feldtest

Nach der Kfz-Fertigung wird das Steuergerät nur noch im Garantie- oder Regressfall beim Feldtest geprüft (C). Dieser Überbegriff fasst alle Tests an bereits ausgelieferten Produkten zusammen.

Dies kann entweder firmenintern zur Befundung oder aber auch in Autowerkstätten zu Reparaturzwecken geschehen. Der Fokus liegt dabei nicht mehr nur auf der Prüfung sondern auch auf der möglichst genauen Diagnose / Fehlerlokalisierung. Die Identifikation fehlerhafter Steuergeräte kann hier zur Abwendung falscher Garantieforderungen genutzt werden, die genaue Diagnose hilft bei einer Reparatur, bei eventuellen Chip-Redesigns oder auch bei Fragen der Verantwortlichkeit und Produkthaftung.

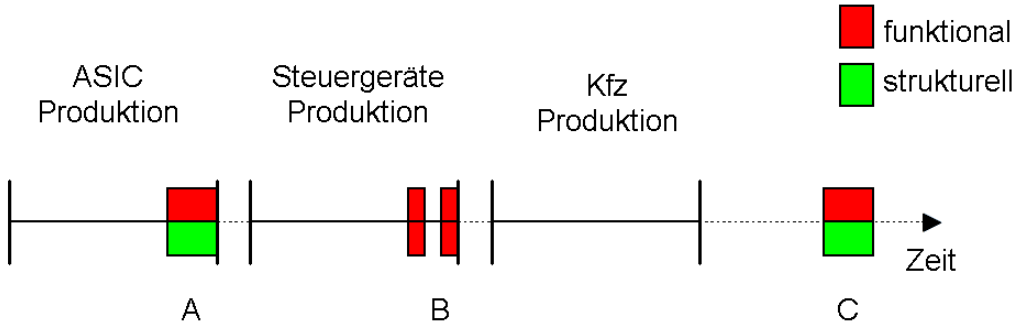


Abbildung 2.2: Testbereiche Steuergeräteproduktion

In der Praxis ist der Feldtest bisher auf das Auslesen eines internen Fehlerspeichers im Zuge einer Kfz-Reparatur beschränkt. Hierzu werden Pocket PCs oder auch Embedded Systeme mit spezieller Software zur Verfügung gestellt. Diese Diagnoseart setzt voraus, dass sich ein möglicher Fehler nicht im Steuergerät, speziell nicht im Fehlerspeicher, dem Hauptprozessor oder dem Kommunikationsinterface befindet. Die anschließende Fehlerdiagnose ist auf Fehlerfälle begrenzt, in denen die Steuergeräte-Software einen Eintrag in den Fehlerspeicher — einen internen Electrically Erasable Programmable Read-Only Memory (EEPROM) — vorsieht. Der firmeninterne Befundungstest ist flexibler; hier können Entwicklungswerkzeuge und auch das Wissen über ASIC-interne Teststrukturen genutzt werden.

2.3 Steuergeräte im Automobilbereich

Steuergeräte als Kontrolleinheiten des Kfz steuern und regeln viele Abläufe, von der zeitkritischen Zündung und Einspritzung im Motorblock bis hin zu Funktionen des Fahrgastkomforts. Abbildung 2.3 verdeutlicht auf Elektronik basierende Funktionen eines typischen Gesamtfahrzeugs, aufgeteilt in die Bereiche Antriebsstrang (Motor, Getriebe, Batterie), Karosserie & Innenraumkomfort (Licht, Spiegel, Kimaanlage, Amaturenbrett, Sitze), Sicherheit (Airbags, ABS, ESP) und Fahrwerk (Federung, Servolenkung). Während rechenlastige Hauptprogramme auf leistungsstarken Mikrocontrollern (MCUs, Micro Controller Units) bzw. Mikroprozessoren (MPUs, Micro Processing Units) die Abläufe koordinieren werden spezielle Ansteuerungen und Sensorauswertungen oft ausgelagert, um dann in ASICs realisiert zu werden. Bei komplexen Regelungen kann der entsprechende ASIC auch Controller oder DSPs (Digital Signal Processors) samt eigener Firmware enthalten.

Die in dieser Arbeit untersuchte Art von Steuergerät fällt in den Bereich des Antriebsstrangs (Powertrain) und dient somit der Steuerung und Regelung der Motoreinspritzung. Neben einem Hauptcontroller sind bis zu 10 Module enthalten, die jeweils aus einem ASIC und seiner externen Beschaltung bestehen. Die Module stellen Vorlagen dar, die je nach Anwendung und Ausstattung des Kfz in das entsprechende Steuergerät integriert werden.

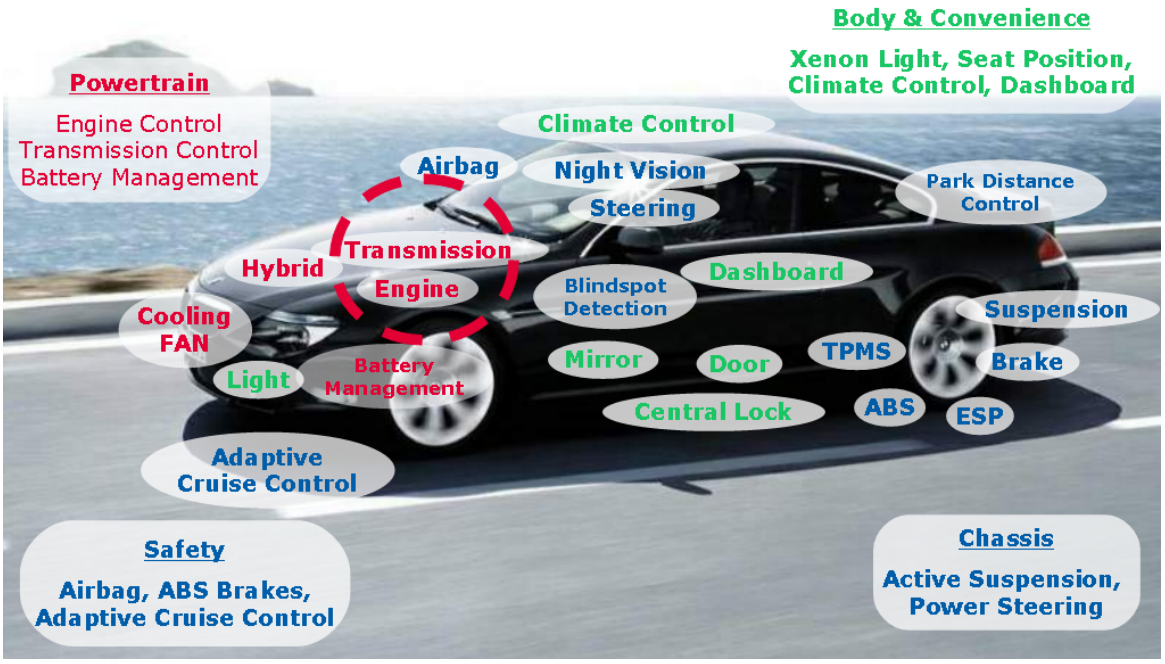


Abbildung 2.3: Elektronische Funktionen des Kfz, Quelle [Hilo7]

2.4 Interne Teststrukturen auf Steuergerät-Ebene

Auch wenn das Steuergerät selbst — im Gegensatz zu den verbauten ASICs — keine dedizierten Teststrukturen enthält, so lassen sich dennoch programmierbare Komponenten zum Testen verwenden. Hierzu zählt der vom Hauptcontroller aus ansteuerbare, systemweite SPI-Bus als Kommunikationsmittel genauso wie alle frei programmierbaren Controller, oder der BOSCH-interne MSC-Bus.

2.4.1 Serial Peripheral Interface (SPI)

Einleitung

Der Überbegriff Serial Peripheral Interface umfasst eine Reihe nicht-patentierter Varianten eines seriellen synchronen 1-Bit-Protokolls zur Kommunikation zwischen Microcontrollern und Peripheriegeräten. Nachdem es erstmalig 1979 von Motorola im M68HC03 Mikrocontroller als reine Hardwareimplementierung vorgestellt wurde [Lee09], zogen andere IC-Hersteller mit vergleichbaren Protokollen nach (vgl. National Semiconductor, Microwire). Die frei einsetzbare Ansteuerungsvariante hat sich — auch aufgrund des Vorteils einer schnellen Kommunikation bei vergleichsweise kleinem Verdrahtungsaufwand — großflächig durchgesetzt. Heutzutage wird eine große Anzahl an ICs per SPI angesprochen. Hierzu zählen Speicher-ICs (EEPROM, SRAM, Serial Flash, SD-Karten), externe Peripherie zur Funktionserweiterung (DACs, ADCs, Ethernet, GSM, GPS) als auch Controller (Displaycontroller, DSPs, kleinere Mikrocontroller).

Signalansteuerung

Das Interface besteht aus 4 Signalen. Der Master einer Kommunikation steuert 3 Signale an, der Slave antwortet auf einem Signal. Dabei befindet sich zwischen MOSI (Master-Out-Slave-In) und MISO (Master-In-Slave-Out) ein in den Slave integriertes Shiftregister, welches mit SCLK getaktet wird. MOSI stellt den Eingang, MISO den Ausgang des Shiftregisters dar. Per Chipselect (CS) kann das Register aktiviert bzw. deaktiviert werden. Der folgende Signalverlauf (Bild 2.4) stellt eine typische SPI-Kommunikation dar. Man erkennt, dass der Master vor der eigentlichen Kommunikation das Chipselect-Signal des Slaves aktiv (low) schaltet. Nach dieser Startkondition wird mit jedem Taktschlag ein Bit in beiden Richtungen transferriert. Es wird immer eine feste Anzahl von Bits übertragen, die zusammen einen Datenframe bilden. Je nach Implementierung können einzelne Bits eines Datenframes auch als Parity Bits zur Erkennung von Übertragungsfehlern definiert sein.

Obwohl das SPI-Protokoll voll-duplex ausgelegt ist wird es in den meisten Anwendungen halb-duplex eingesetzt. Während die Signalleitungen eine gleichzeitige Kommunikation in beiden Richtungen gestatten, kann vom Ablauf her gesehen der Slave erst auf eine Anfrage des Masters antworten, nachdem diese komplett übertragen wurde. Demnach werden zwei Datenframes benötigt, zunächst um dem Slave einen Befehl zu übertragen und anschließend zum Auslesen der Antwort. Durch dieses Frage-Antwort-Spiel arbeitet das Protokoll dann nur noch halb-duplex.

2.4.2 SPI-Bus

Die Anordnung und Verbindung des SPI Masters und den vorhandenen Slaves wird durch die gewählte Bus-Topologie festgelegt. Hierbei unterscheidet man die typischen Varianten von Kaskaden- und Sternschaltung, dargestellt in Bild 2.5. Bei der Kaskadenschaltung werden mehrere Slaves so hintereinander verschaltet, dass sich ein einziges langes Shiftregister ergibt. Die Chipselect-Leitung

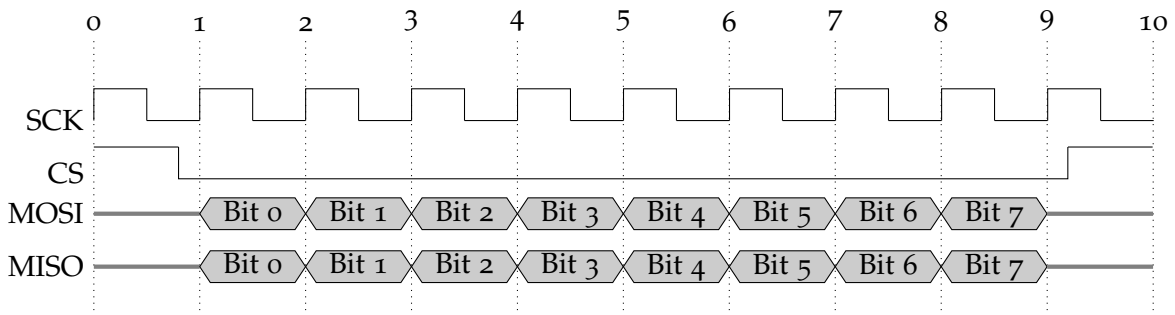


Abbildung 2.4: Signalverlauf SPI Datenframe

ist global, d.h. die Slaves können nur gemeinsam, nicht aber einzeln aktiviert werden. Bei der Sternschaltung sind die Signale MISO und MOSI global. Die Slaves werden einzeln durch unterschiedliche Chipselect-Signale aktiviert und deaktiviert. Meist wird als Topologie die Sternschaltung eingesetzt, der höhere Verdrahtungsaufwand bei getrennten Chipselect-Signalen wiegt oft weniger schwer als der Geschwindigkeitsnachteil bei der Kaskadenschaltung.

Zur Verringerung des Verdrahtungsaufwands bei mehreren Slaves kann das SPI-Protokoll durch Multiplexing erweitert werden, so dass mehrere Slaves gleichzeitig senden oder empfangen können [Holoo]. Das im hier untersuchten Steuergerät eingesetzte Adress-Multiplexing erlaubt in der Sternschaltung, mehrere Slaves an einem einzigen Chipselect zu betreiben. Dabei werden einige Bits des Datenframes als Adressbits aufgefasst. Mehrere gleichzeitig per Chipselect aktivierte Slaves gleichen die empfangenen Adressbits mit der ihnen zugeordneten Adresse ab und antworten erst dann auf dem MISO-Signal, wenn sie adressiert wurden.

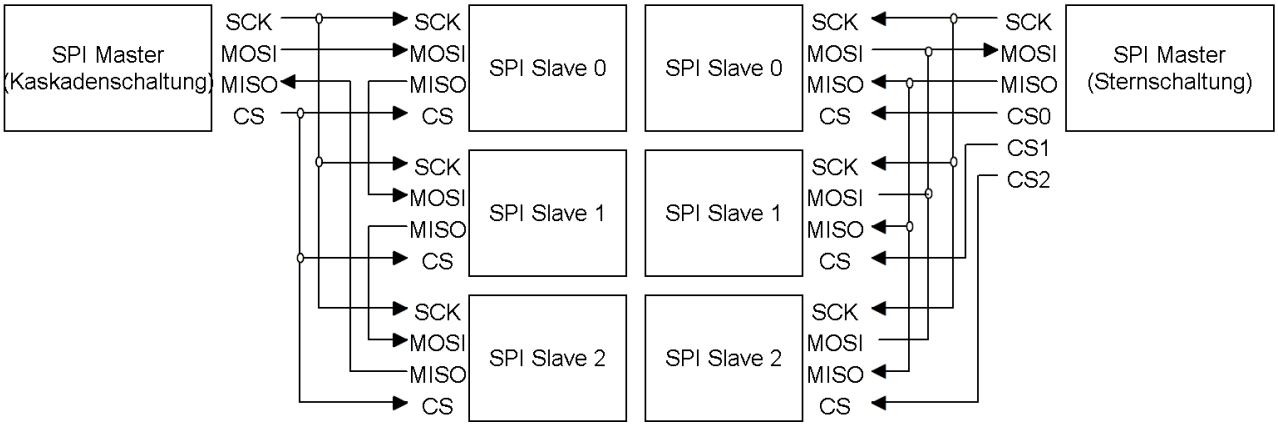


Abbildung 2.5: SPI Topologie: Kaskadenschaltung / Sternschaltung

2.4.3 Programmierbare Komponenten

Auf den frei programmierbaren Controllern des Steuergerät können strukturelle softwarebasierte Testmethoden entwickelt werden. Diese sind meist langsamer als in Hardware implementierte BIST-Methoden (vgl. Kapite 2.5.3), kommen dafür jedoch — sofern das Testprogramm von außen eingespielt werden kann — ohne Hardware-Overhead aus. Im Bereich leistungsstarker Mikrocontroller wird bereits mit softwarebasierten Selbsttests gearbeitet [CHPW05]. Sobald der Controller das ihm zugeordnete Testprogramm erfolgreich durchgeführt hat, kann er als fehlerfrei betrachtet werden — und als Testcontroller für weitere softwarebasierte Tests zum Einsatz kommen. Eine derart aufgebauter, hierarchischer Testplan erzielt auf Systemebene eine hohe Testabdeckung. Dies ist insbesondere für den Produktionstest interessant; im Feldtest muss zunächst geklärt werden, von wo die Testprogramme eingespielt werden. Dies kann entweder durch ein externes Diagnosegerät oder aus integriertem Speicher erfolgen. Im hier entwickelten, firmen-internen Befundungstest werden Entwicklungswerkzeuge eingesetzt, um Testsoftware im Hauptcontroller eines Steuergeräts auszuführen.

2.5 Integrierte Teststrukturen auf ASIC-Ebene

Auf Ebene der ASICs werden zur Minimierung des Testaufwands bei der Produktion mehrere Teststrukturen integriert. Dies umfasst eine standardisierte Testschnittstelle (JTAG), über die dann weitere eingebaute Selbsttests (Built-In Self-Test, BIST) ausgeführt werden.

2.5.1 JTAG IEEE 1149.1

Einleitung

1990 gründeten die Experten von über 200 Herstellern die sogenannte Joint Test Action Group (JTAG). Ziel dieser Vereinigung war die Standardisierung von Testmethoden, welche noch im gleichen Jahr im IEEE 1149.1-1990 Standard (Standard Test Access Port and Boundary Scan Architecture, [JTA90]) publiziert wurden. Der Standard enthält sowohl für das ASIC- als auch für das Elektronikdesign Richtlinien, welche unter dem Begriff "Design for Test"(DFT) zusammengefasst einen späteren Produkttest vereinfachen. Die standardisierte Testschnittstelle ist vielfach einsetzbar und erlaubt neben der Durchführung von Boundary Scan Tests und Built-In Self-Tests auch die Programmierung von ICs im verbauten Zustand (ICP, In-Circuit Programming). Der Standard wurde im Jahre 1995 um die Boundary Scan Description Language (BSDL) zur Beschreibung des Verhaltens standardkonformer ICs erweitert. Die IC-spezifische BSDL-Datei führt Pin-Bezeichnungen, Register, Registergrößen und unterstützte Instruktionen in VHDL-Syntax (Very High Speed Integrated Circuit Description Language) auf. Das Bauteil kann so schnell und einfach in automatisierten Test-Entwicklungsumgebungen einbezogen werden.

Boundary-Scan-Architektur

Die Boundary-Scan-Architektur eines beispielshaften ICs ist in Bild 2.6 dargestellt. Zur Ansteuerung der zusätzlichen Testschaltung wird der IC zunächst um einen Test Access Port (TAP) erweitert. Dieser Zustandsautomat wird über die Eingangssignale TCK (Test Clock) und TMS (Test Mode Select) gesteuert und erlaubt Operationen bezogen auf das Instruktionsregister und das aktuell ausgewählte Datenregister. Ein optionales Reset-Signal (TRST, Test Reset) versetzt den TAP in seinen Startzustand. Je nach Zustand des TAPs ist eines der JTAG-Register als Shiftregister zwischen dem

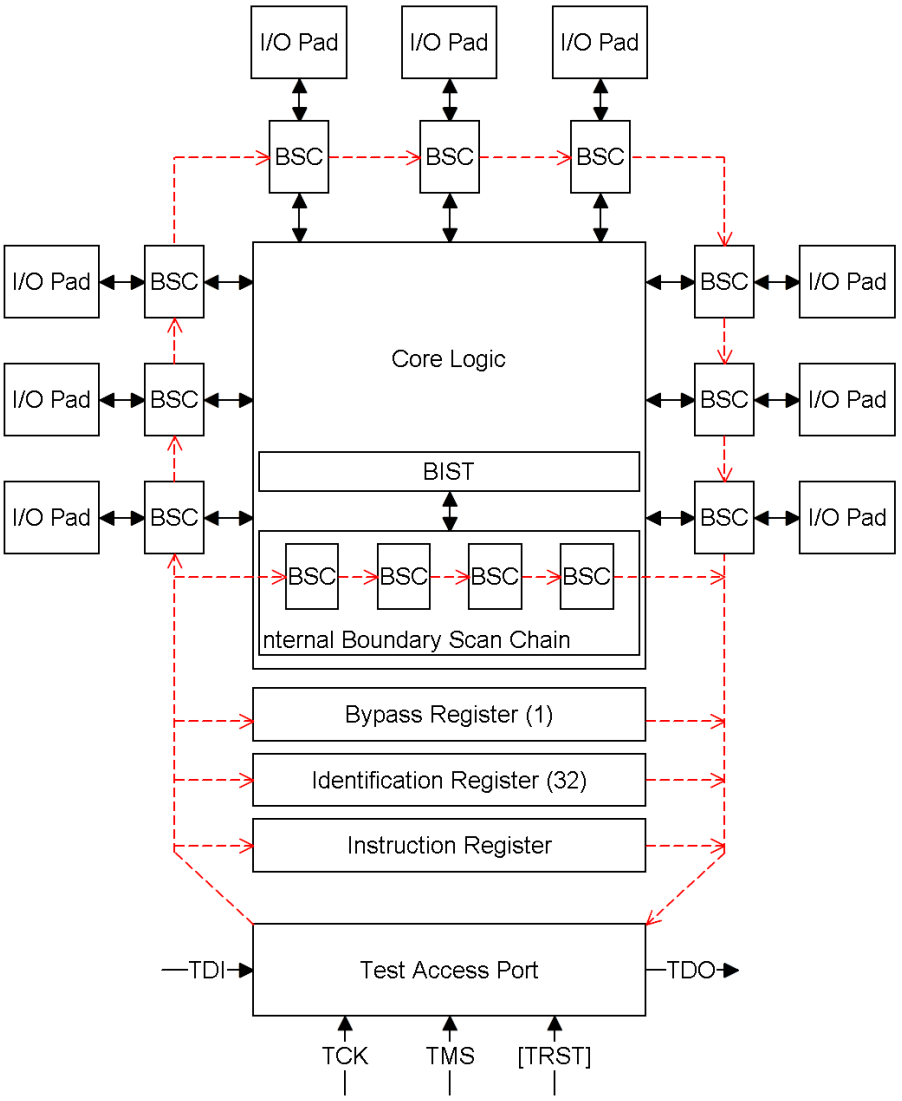


Abbildung 2.6: Boundary Scan Design

Eingang TDI und dem Ausgang TDO verfügbar. Das Instruktionsregister dient dabei der Ausführung IC-eigener Funktionen und Testmethoden, durch die auch das aktuelle Datenregister festgelegt wird. Das Identifikationsregister wird beispielsweise durch die IDCODE-Instruktion ausgewählt. Es enthält eine 32-bit-breite Identifikationsnummer und dient zunächst einmal der Verifizierung, mit welchem IC die Testkommunikation stattfindet.

Die JTAG-Register bestehen alle aus mehreren seriell verbundenen Boundary-Scan-Zellen. Dies ist insbesondere für das Boundary-Scan-Register von Bedeutung, welches alle Scan-Zellen der digitalen I/O-Pins umfasst. So können die Zustände aller digitalen Pins gesetzt und gelesen werden. Interne Boundary-Scan-Register dienen der Kommunikation mit zusätzlichen Teststrukturen, wie beispielsweise den später vorgestellten Built-In Self-Test-Methoden (BIST, vgl. Kapitel 2.5.3).

Bild 2.7 zeigt eine typische Boundary-Scan-Zelle. Digitale Eingangspins werden an DATA_IN angeschlossen, DATA_OUT stellt die Verbindung zur internen Chiplogik her. Durch das MODE-Signal kann das DATA_IN-Signal nach DATA_OUT durchgeschaltet werden, so dass die Zelle transparent wird. Das erste Flipflop der Scanzelle (DFF A) kann mithilfe von CAPTURE den Zustand von DATA_IN annehmen. Im SHIFT-Modus wird dieses Flipflop dann als Bit-Zelle des Shiftregisters betrieben. Durch das UPDATE-Signal wird der Zustand des ersten Flipflops (DFF A) in das zweite Flipflop (DFF B) übertragen. Je nach MODE-Signal wird dieses Signal dann statt DATA_IN über DATA_OUT der internen Logik als Eingangssignal präsentiert.

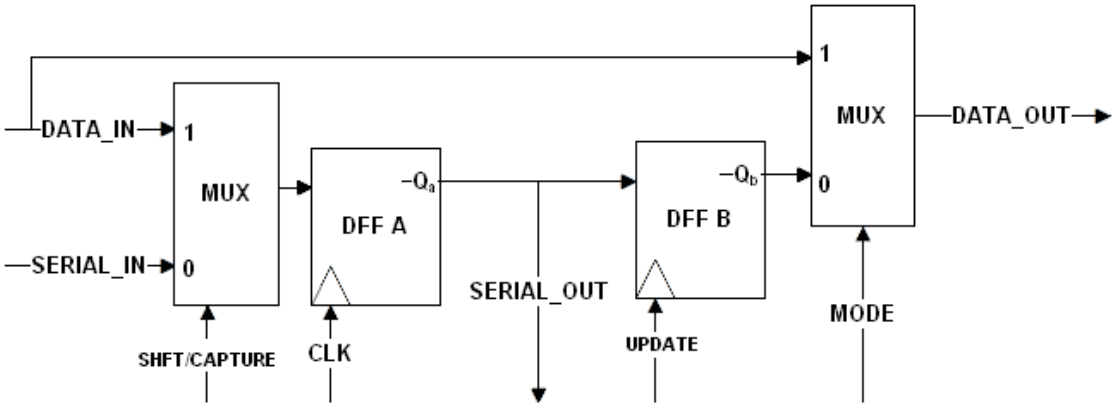


Abbildung 2.7: Implementierung einer Boundary-Scan-Zelle

Der Test Access Port (TAP)

Bild 2.8 stellt den Zustandsautomaten des TAP dar. Neben den Zuständen RESET und IDLE beziehen sich alle anderen entweder auf das Datenregister DR (linker Zweig) oder auf das Instruktionsregister IR (rechter Zweig). Die Übergänge des Automaten passieren bei einer steigenden Flanke auf dem TCK-Signal in Abhängigkeit vom aktuellen Wert des TMS-Signals. Der TRST-Eingang ist deshalb

optional, weil der RESET-Zustand von jedem anderen Zustand durch das Einspielen der TMS-Sequenz 11111 ebenfalls erreichbar ist. Zur Ausführung einer JTAG-Instruktion wird der TAP vom Zustand RESET durch die Sequenz 01100 in den Zustand SHIFT_IR gebracht. In diesem Zustand ist das Instruktionsregister zwischen TDI und TDO seriell verfügbar. Die typischerweise 5 bis 8 Bit große Instruktion wird mit ebenso vielen Takten per TDI in das Instruktionsregister übertragen, während der TAP im SHIFT_IR-Zustand verbleibt. Im Anschluss wird der TAP durch die Sequenz 110 in den Zustand IDLE versetzt. Erst hier wird die geshiftete Instruktion ausgeführt. Sofern weitere Kommunikation nötig ist, kann die Instruktion ein ihr zugeordnetes Datenregister auswählen. Im SHIFT_DR-Zustand kann dieses gelesen und beschrieben werden. Vor jedem SHIFT-Vorgang muss der TAP den jeweiligen CAPTURE-Zustand passiert haben. In diesem wird das entsprechende Register parallel mit dem Inhalt geladen, der im Shiftzustand an TDO beobachtet werden kann. Im UPDATE-Zustand am Ende einer Shiftoperation wird der geshiftete Inhalt vom Register parallel ausgegeben.

Topologie auf Board-Ebene

Zur Verbindung mehrerer 1149.1-konformer ICs auf Board-Ebene sieht der Standard eine serielle Topologie (Bild 2.9) vor. Alle verfügbaren TAPs werden durch die globalen TCK- und TMS-Signale synchron angesteuert. Die seriellen Anschlüsse der ICs (TDI/TDO) werden hintereinandergeschaltet, so dass sich eine sogenannte Scan-Kette (engl. Scan-Chain) bildet. Ein JTAG-Anschluss auf Board-Ebene reicht damit aus, um die komplette Testlogik aller ICs anzusprechen. Mithilfe der BYPASS-Instruktion können einzelne ICs zur Verkürzung der Scan-Ketten-Länge in den Bypass-Modus versetzt werden. So ist als Datenregister das 1-Bit-breite Bypass-Register festgelegt, wenn der entsprechende IC vom aktuellen Testvorgang ausgeschlossen wird.

JTAG-Instruktionen

Folgende Instruktionen sind im Standard vorgesehen:

BYPASS : Das Bypass-Register wird als Datenregister ausgewählt.

EXTEST : Der Zustand der digitalen Ausgänge der internen Logik wird im CAPTURE_DR-Zustand in das Boundary-Scan-Register geladen. Dieses kann im SHIFT_DR-Zustand beschrieben und ausgelesen werden. Im folgenden UPDATE_DR-Zustand wird der neue Inhalt des Scan-Registers an den digitalen Ausgängen des ICs ausgegeben.

INTEST : Im Gegensatz zur EXTEST-Instruktion wird hier die Kommunikation zwischen den digitalen Eingangspins und den Eingängen der internen Logik manipuliert. Im CAPTURE_DR-Zustand nimmt das Scan-Register den Zustand der digitalen IC-Eingänge an. Nach erfolgter Shift-Operation wird der Zustand des Scan-Registers dann an die interne Logik weitergegeben.

HIGHZ : Bei dieser Instruktion werden alle Digitalausgänge des ICs auf hohe Ausgangsimpedanz umgeschaltet.

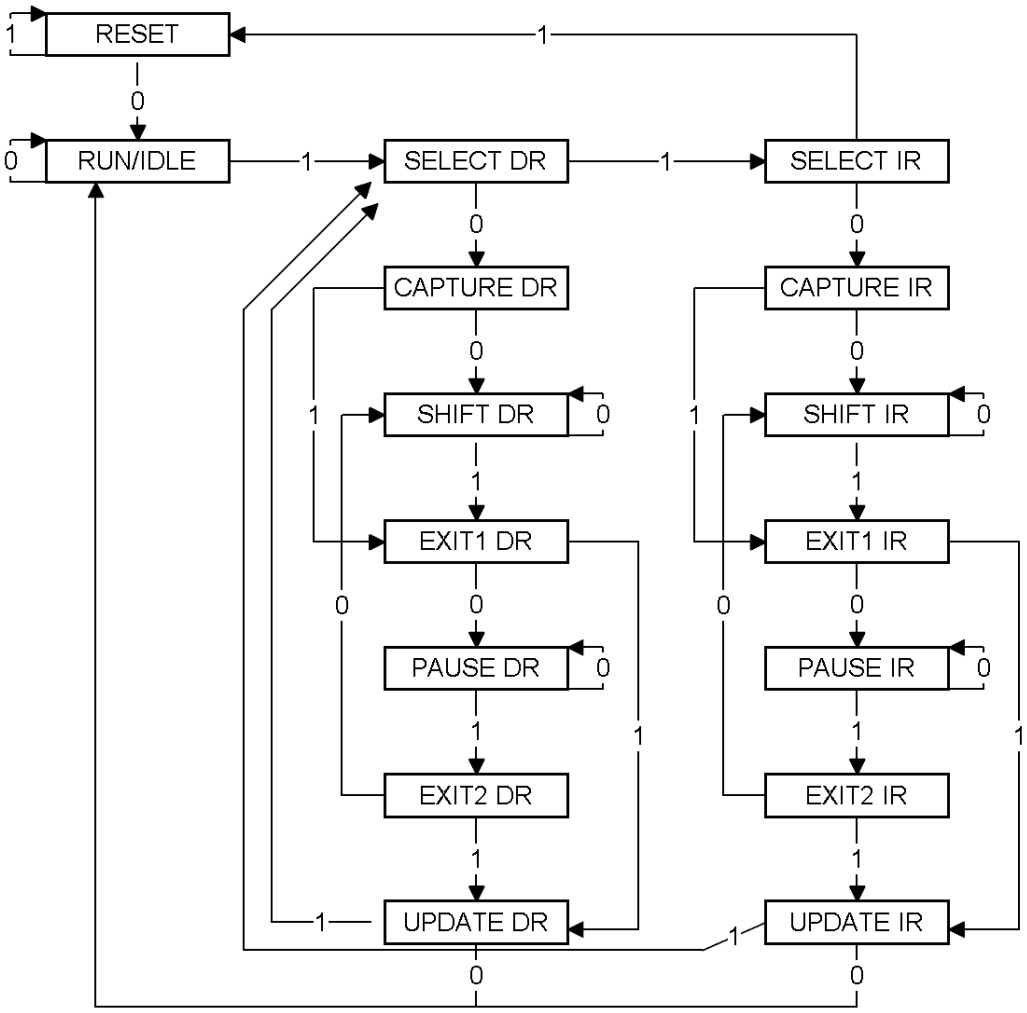


Abbildung 2.8: Der Test Access Port (TAP) Zustandsautomat

IDCODE (optional) : Sofern vorhanden aktiviert dies das Identifikationsregister. Ansonsten wird das Bypass-Register ausgewählt.

Abweichungen vom Standard

Aufgrund der großen Anzahl an im 1149.1 Standard enthaltenen Forderungen existieren nicht-konforme ICs. Dies kann unbeabsichtigt oder auch aus Gründen der Inkompatibilität zwischen Standard und Produktanforderungen geschehen. Es folgt eine kurze Auflistung der häufigsten Verletzungen des Standards [Rob94]:

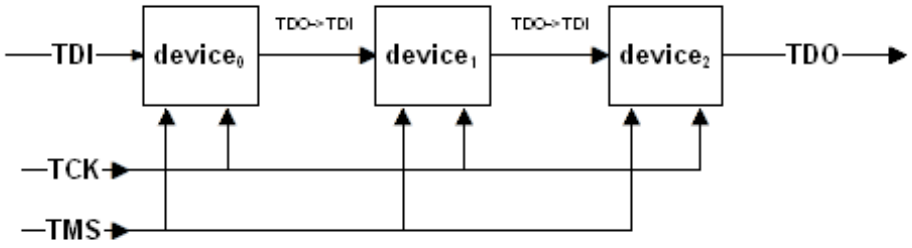


Abbildung 2.9: Scan Chain

Digitale Pins ohne Boundary Scan Zellen: Oftmals wird nicht jeder digitale Pin mit einer Boundary Scan Zelle ausgestattet. Diese Form der "milden" Abweichung kann — sofern der betroffene digitale Pin auf Systemebene als analog gedeutet wird — toleriert werden.

Falsch implementierte Boundary Register: Fehlende Update Latches oder nicht als dedizierte Testlogik eingesetzte Boundary Register können beim Shiften der enthaltenen Daten zu unvorhergesehenen Effekten führen. Derartige Abweichungen müssen bei der Testentwicklung berücksichtigt werden.

Taktanforderungen: Der Standard verbietet die Notwendigkeit von Taktsynchronizationen zwischen Testtakt und Systemtakt der getesteten Komponente. Einige ICs sind zur korrekten Ausführung von implementierten Testmethoden dennoch auf eine spezielle Taktung des Systems oder auf Takt-Synchronizationen angewiesen.

Die hier aufgeführten Nicht-Konformitäten bergen das Riskio von zerstörungsbehafteten — aber nach dem Standard erlaubten — Testoperationen. Sie verkomplizieren außerdem eine automatisierte Testentwicklung auf Board-Ebene, da die zum Standard gehörende BSDL-Sprache derartige Abweichungen nicht vorsieht.

2.5.2 Erweiterung IEEE 1149.7

Im letzten Jahr wurde eine Erweiterung des IEEE 1149.1 JTAG Standards unter der Bezeichnung IEEE 1149.7 (Standard for Reduced-pin and Enhanced-functionality Test Access Port and Boundary-Scan Architecture, [IEE10]) veröffentlicht. Diese JTAG-Erweiterung bietet neben einer variableren Topologieauswahl und verringertem Verdrahtungsaufwand (mindestens 2 statt 4 Signale) auch einen optimierten Scan-Durchsatz sowie für Debugging-Zwecke vorteilhafte Funktionserweiterungen. Eine 1149.1-TAP-Schnittstelle kann durch das Ergänzen eines P1149.7-Adapters einfach erweitert werden. Außerdem können 1149.7-TAPs und 1149.1-TAPs auf Boardebene koexistieren bzw. in einigen Fällen sogar durch die gleichen TAP-Signale angesteuert werden. Umgangssprachlich wird dieser Standard

auch als Concurrent-JTAG bezeichnet, da in der Star-2-Topologie die gleichzeitige Übertragung von Scandaten und Debugging-Informationen möglich ist [Ley09].

Bild 2.10 demonstriert die für den 1149.1-Standard typische serielle Topologie im Vergleich zu den neuartigen Star-4- und Star-2-Topologien. Besonders die Star-2-Topologie hat mit 2 globalen Signalen den geringsten Verdrahtungsaufwand. Die Neuerungen im Vergleich zum 1149.1-Standard sind nach Kompatibilitätsklassen sortiert. In der Klasse T0 wird zunächst volle Kompatibilität zu 1149.1-konformen TAPs hergestellt. Ein IC der Klasse T1 enthält zusätzlich 1149.7-spezifische Instruktionen und Register. In T2 sind Scanformate zur optimierten Datenübertragung in seriellen 1149.1-Topologien vorgesehen. T3 bringt Scanformate zur Ansteuerung und direkten Adressierung von TAPs in der Star-4-Topologie mit sich. Die T4-Klasse umschreibt weitere Scanformate, die eine paketorientierte Datenübertragung in der Star-2-Topologie ermöglichen. Die letzte Kompabilitätsklasse T5 ermöglicht das Übertragen von zusätzlichen Daten während einer Übertragung von Scandaten.

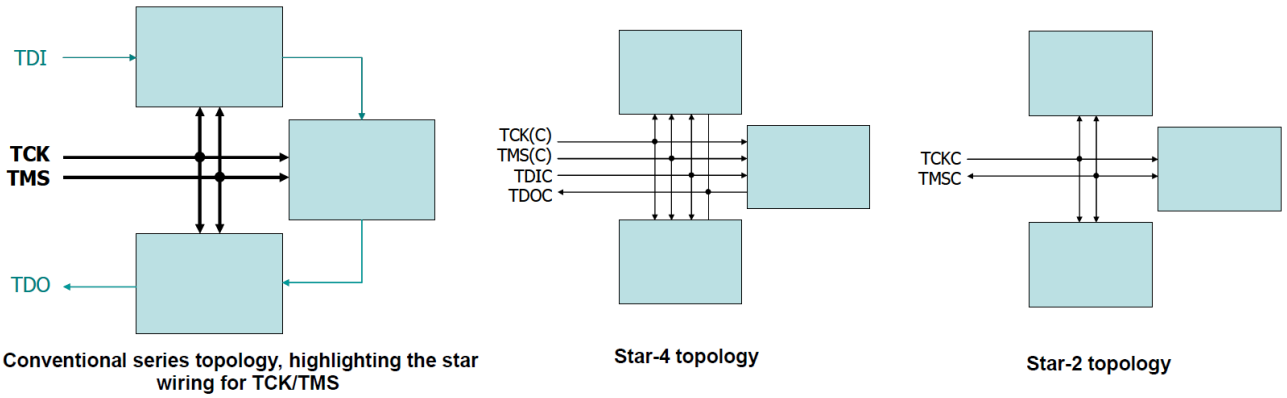


Abbildung 2.10: IEEE 1149.7: serielle, Star-4- und Star-2-Topologie, Quelle [Ley09]

2.5.3 Built-In Self-Test (BIST)

Einleitung

Durch einen hardwarebasierten, eingebauten Selbsttest (Built-In Self-Test, BIST) können sich ASIC-Komponenten selbst strukturell testen. Sowohl die Generierung der Testsignale als auch die Überprüfung der Antwortsignale wird dabei auf dem IC („on-chip“) durchgeführt. Hierfür ist zusätzliche Testlogik im Chipdesign notwendig. Vorteile dieser Test-Strategie sind eine starke Verringerung des Testaufwands, die Einsparung teurer externer Tester (ATE, Automated Test Equipment) und deren Wartung, ein vereinfachtes Testdesign mit automatisierter Entwicklungssoftware, sowie eine höhere Fehlerabdeckung in kürzerer Testzeit. Zusätzlich erlauben BIST-Verfahren einen At-Speed-Test, der auch Transition Delay Faults abdeckt [MLBb]. Je nach implementierter BIST-Architektur kann außer der getesteten Komponente auch die Testlogik selbst geprüft werden. Aufgrund fehlender Notwendigkeit für externes Testequipment können BIST-Testmethoden in jeder Produktionstufe und

in allen Testbereichen, auch im Feld ausgeführt werden. Eine Parallelisierung des Tests ist ebenfalls problemlos umsetzbar.

Klassifizierung von BIST-Methoden

BIST-Methoden werden wie folgt klassifiziert (Bild 2.11): man unterscheidet zunächst zwischen Online-Verfahren, die während des Normalbetriebs der zu testenden Schaltung durchgeführt werden, und Offline-Verfahren, die einen speziellen Testmodus benötigen, in dem die zu testende Komponente nicht funktionstüchtig ist. Dabei geschehen nur die Concurrent-Online-Verfahren wirklich während des funktionalen Betriebs; Non-concurrent-Online-Methoden werden zwar im Normalbetrieb durchgeführt — jedoch nur, wenn die zu testende Komponente gerade nicht benötigt wird. Bei den häufiger anzutreffenden Offline-Verfahren unterscheidet man zusätzlich zwischen funktionalen und strukturellen Testmethoden.

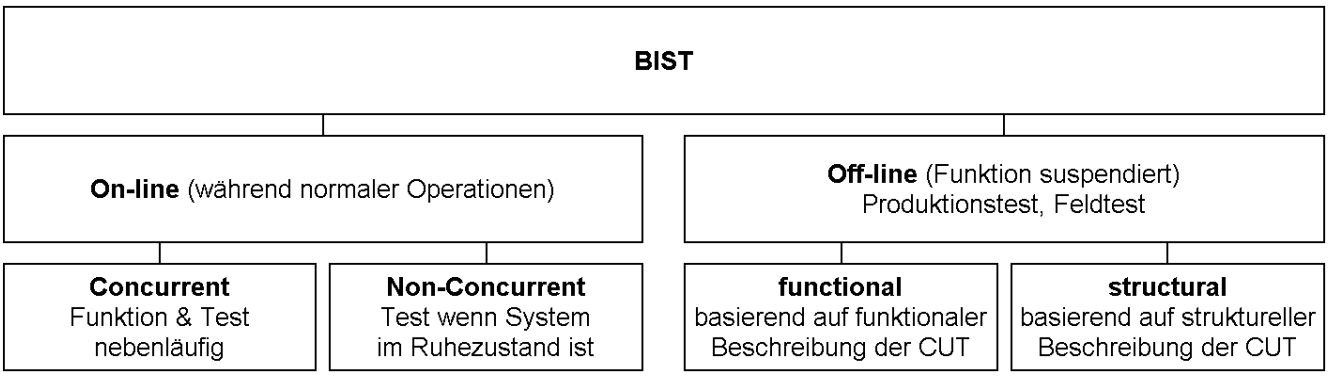
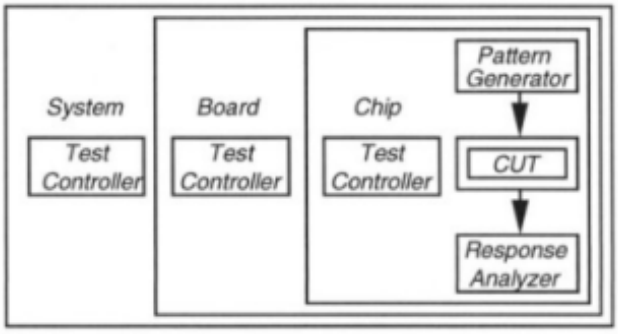


Abbildung 2.11: Arten von BIST-Methoden, Quelle [Moh]

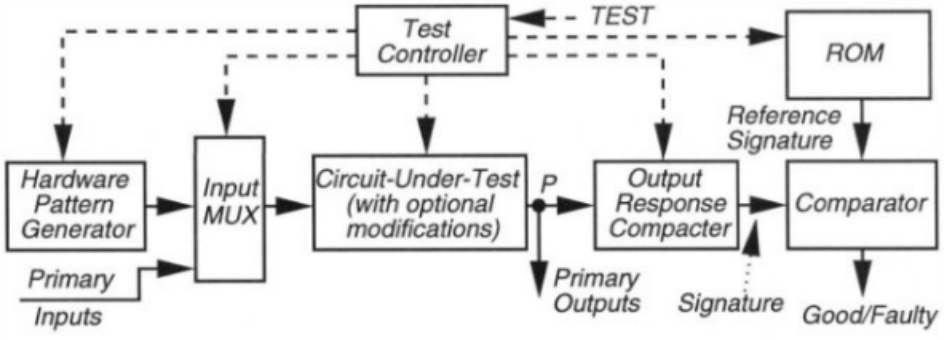
Bild 2.12 verdeutlicht den durch Testcontroller auf allen Ebenen gesteuerten Testprozess. Die Testdaten für die zu testende Komponente (Component-Under-Test, CUT) werden im Chip durch einen Pattern-Generator erstellt. Die Testausgänge werden — ebenfalls im Chip — von einem Response-Analyzer verarbeitet. Am Ende des Prozesses entscheidet der Testcontroller des Chips in Anhängigkeit vom Response-Analyzer über das Testergebnis. Dieses wird im Anschluss jeweils an den Testcontroller auf nächsthöherer Systemebene weitergereicht. Eine Testschnittstelle zum Testcontroller des Chips ist durch den 1149.1-Standard meist gegeben. Durch eine JTAG-Instruktion wird der eingebaute Selbsttest angestoßen, das Ergebnis liegt dann nach kurzer Wartezeit in einem Datenregister vor. Hierbei ist keine Diagnoseinformation enthalten; es wird lediglich ein Wahrheitswert über das Ergebnis des durchgeführten Tests zurückgegeben. Eine detailliertere Darstellung der Teststrukturen auf Chipebene ist in Bild 2.13 gegeben. Es zeigt zusätzlich die Umschaltung der Eingangssignale (Input Mux) auf den Testdatengenerator sowie einen Komparator, der die beim Test erzeugte, kompaktierte Anwort-Signatur mit einem Referenzwert aus einem ROM-Speicher vergleicht. Die hier dargestellte Testarchitektur fällt aufgrund der Umschaltung der Eingangssignale in den Bereich der Non-Concurrent- oder sogar Offline-Verfahren.

BIST Testprozess



BIST hierarchy.

Abbildung 2.12: BIST Process auf Systemebene, Quelle [MLBb]



BIST process.

Abbildung 2.13: BIST Process auf IC-Ebene, Quelle [MLBb]

Memory-BIST

Eine typischerweise mit BIST-Testmethoden ausgestattete Komponente sind integrierte Speicher. Diese umfassen aufgrund eines stetig wachsenden Speicherbedarfs viele Einzeltransistoren und führen bei Defekten zu schweren, nicht abschätzbaren Fehlfunktionen. Besonders im Hinblich auf den „Zero Defekt Tolerance“-Grundsatz im Automobilbereich werden hier Online-Verfahren eingesetzt. Im untersuchten ASIC wird beispielsweise bei jedem Systemstart ein Non-Concurrent March-C-Test auf allen Speicherbereichen durchgeführt. Im Hinblick auf weitere Testanwendungen ist dieser Test

sogar programmierbar, d.h. die eingebauten Teststrukturen erlauben auch ausführlichere March-Tests. Ein solcher Test besteht aus mehreren, sogenannten March-Elementen. Diese bewirken das Schreiben oder Lesen eines Wertes in allen Speicherzellen. Zusätzlich ist jedem March-Element eine Richtung zugeordnet, in der die Speicherzellen durchlaufen werden. So lassen sich Stuck-At, Transition und Coupling-Fehler abdecken. Der zusätzliche Platzbedarf ist hier besonders gering, da zur Durchführung von March-Tests lediglich ein Speicherinterface sowie ein Testcontroller benötigt werden. Concurrent-Speicher-BISTs bringen nicht nur einen leicht höheren Platzaufwand sondern auch den großen Nachteil von erhöhten Zugriffszeiten auf Speicherzellen im Normalbetrieb mit sich. Eine Methode ohne expliziten Platzaufwand stellen sogenannte softwarebasierte Selbsttests (SBST, Software-Based Self-Test) dar. Diese sind jedoch auf einen existierenden Controller angewiesen, benötigen mehr Testzeit und vergrößern den Speicherbedarf der Software. In einem typischen Beispiel für Memory-BIST [MLBb] wird der Testalgorithmus für 4Mb DRAM in Hardware auf dem Chip realisiert. Während sich der Platzbedarf um 1% erhöht, verringert sich die Testzeit um 2-3 Größenordnungen, da der Test in Betriebsgeschwindigkeit stattfindet.

Logic-BIST

Unter Logic-BIST bzw. Random-Logic-BIST versteht man eingebaute Selbsttests für beliebige kombinatorische und sequentielle Logikschaltungen. Die sequentiellen Anteile werden dabei oft in Scanketten zur Verfügung gestellt, um die internen kombinatorischen Anteile testbar zu machen. Bei der Generierung der Testdaten unterscheidet man zwischen einer kompletten (engl. exhaustive), einer pseudo-zufälligen (engl. pseudo random) und einer deterministischen Vorgehensweise [RD98]. Wenn eine komplette Testdatengenerierung aufgrund der großen Anzahl an Testsignalen ausscheidet, ist ein deterministische Verfahren — bei denen algorithmisch ein Satz von Testsignalen mit hoher Testabdeckung gesucht wird — den pseudo-zufälligen Verfahren vorzuziehen.

Als Komponente zur Testdatengenerierung wird meist auf ein LFSR (Linear Feedback Shift Register) zurückgegriffen. Wie in Bild 2.14 dargestellt handelt es sich hierbei um ein einfaches Shiftregister, welches um Rückführungen (Feedbacks) erweitert ist. Alle zurückgeführten Bits ergeben nach einer exklusiven Oder-Verknüpfung das nächste Eingangsbit. Die Rückführungen charakterisieren dabei das Generatorpolynom des Shiftregisters. Sofern dieses Polynom irreduzibel ist, es also nicht in weitere irreduzible Polynome zerlegt werden kann, wiederholt sich die ausgegebene Bitsequenz eines n-Bit-LFSR erst nach 2^{n-1} Bits. Das hier dargestellte Beispiel besitzt Rückführungen am 4., 5., 6. und 8. Bit. Dies korrespondiert mit dem irreduziblen Generatorpolynom $x^8 + x^6 + x^5 + x^4 + 1$ und gibt eine maximale Sequenz von 128 Bits aus. Auch in der Kryptographie werden LFSRs gern als Pseudo-Zufallszahlengeneratoren eingesetzt, da die von ihnen ausgegebene Bitsequenz gute statistische Eigenschaften aufweist. Der einzige zu meidende Zustand des Registers ist, wenn alle Bits gleich 0 sind, da eine exklusive Oder-Verknüpfung mehrerer Nullen auch immer Null ergibt. Das LFSR kann diesen Zustand somit nicht verlassen und erzeugt eine reine Null-Sequenz. Generatorpolynom und Startzustand eines LFSRs reichen aus, um die generierte Bitsequenz zu berrechnen. Der Einsatz eines LFSRs ist somit nicht zwingend an eine pseudozufällige Testdatengenerierung gekoppelt. Zur deterministischen Testdatengenerierung können mehrere LFSR-Startzustände mit Laufzeiten bestimmt werden. Das LFSR wird dann im Test zu den bestimmten Zeitpunkten neu initialisiert (LFSR

Reseeding) [SN06]. Weitere Optimierungen bei der Testdatengenerierung können durch Bit-Flipping [HJW96] oder auch Bit-Fixing [NAT01] erreicht werden. In diesen Fällen wird das Eingangssignal einer Scan-Kette durch Zusatzlogik zeitweise invertiert (Flipping) oder gesetzt (Fixing). Werden alle Bits des generierenden LFSR parallel als Eingänge für mehrere Scan-Ketten genutzt, so kann die statistische Korrelation zwischen den Scan-Signalen durch einen sogenannten Phase Shifter verringert werden [JR00]. Hierbei dienen unterschiedliche XOR-Kombinationen der LFSR-Bits als Eingänge der Scan-Ketten.

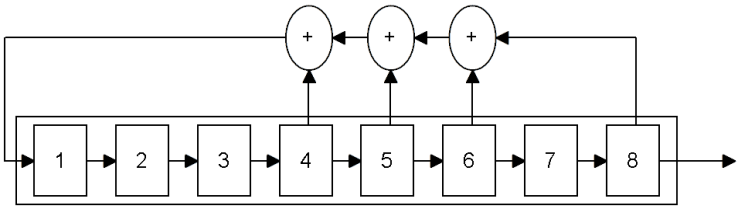


Abbildung 2.14: Linear Feedback Shift Register (LFSR)

Zur Überprüfung der Antwortsignale wird ein Multiple-Input-Signature-Register (MISR, vgl. Bild 2.15) eingesetzt. Es gleicht vom Aufbau her einem LFSR, weist jedoch Eingänge an den Rückführungen auf, die für die Testantworten vorgesehen sind. Am Ende des Tests enthält es eine von den Antworten abhängige Signatur. Falls die zu prüfenden Signale zeitweise unbekannt sind (Signalzustand X, Don't Care), muss dem MISR eine sogenannte X-Masking-Logik [XS11] oder X-Canceling-Logik [Tou07] vorgeschaltet werden. So wird verhindert, dass ungekannte Signalzustände die Signatur beeinflussen.

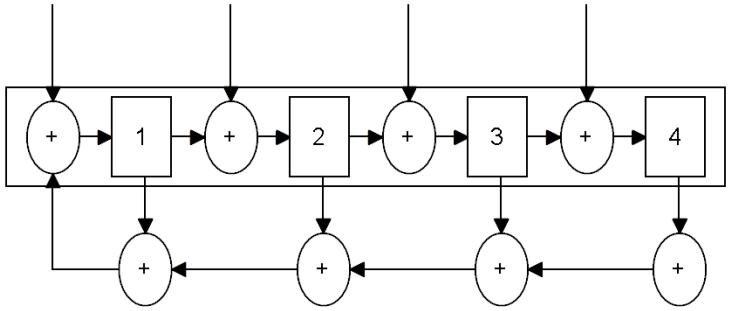


Abbildung 2.15: Multiple Input Signature Register (MISR)

3 Implementierung

3.1 Überblick

Das folgende Kapitel beschreibt die Implementierung einzelner Verfahren, die gemeinsam den Befundungstest darstellen. Zunächst einmal erlaubt die JTAG@SPI-Protokollansteuerung (Kapitel 3.3), JTAG-Kommunikation mit einzelnen ASICs über den steuergerät-internen SPI-Bus durchzuführen. Dies bildet die Grundlage für die Ausführung eingebauter BIST-Methoden bezüglich Speicher und ADC. Für ASICs ohne internen Logik-BIST wird im Anschluss ein softwarebasierter Scan-Pattern Test (Kapitel 3.4) implementiert, der vom Hauptcontroller des Steuergeräts aus durchgeführt wird. Zwei weitere Methoden zeigen das Zusammenspiel von (hardwarebasiertem) BIST und strukturellen, softwarebasierten Tests: der in Kapitel 3.5 vorgestellte BIST zur Taktüberprüfung (des Hauptcontrollers) kann durch den softwarebasierten Abgleich weiterer Taktsignale auf die PLL-Schleifen von ASICs erweitert werden (Kapitel 3.6).

Am Ende werden diese Methoden im Befundungstest (Kapitel 3.7) zusammengefasst. Ein kurzer funktionaler Vortest steigert die Diagnosemöglichkeit bei falschem Testaufbau oder fehlerhaften Testschnittstellen.

3.2 Testaufbau

Bild 3.1 stellt den Entwicklungsaufbau dar. Das Steuergerät wird von einem Labornetzteil mit $\sim 12V$ bei $\sim 0,25A$ versorgt. Der Laptop als Testcontroller hat über das Bosch-interne CFI (CarPu Flash Interface) Zugriff auf den CAN-Bus des Steuergeräts, welches durch die sogenannte CFI Box in einen Testmodus versetzt wird. Mithilfe des ASC@CAN-Protokolls können bis zu 255 Byte lange ASCII-kodierte Nachrichten zwischen Laptop und Hauptcontroller im Steuergerät ausgetauscht werden. In der hier eingesetzten Konfiguration ist dieser Kommunikationskanal wegen dem CAN-Transceiver auf eine Übertragungsrate von 4 MBit/Sek. beschränkt. Neben einfachen Steuerungsnachrichten kann auf diese Weise auch ein Programm in den RAM des Hauptcontrollers übertragen und ausgeführt werden. Nach Initialisierung der Verbindung wird also zunächst ein Firmwareprogramm auf den Hauptcontroller übertragen; im Anschluss lassen sich dann parametrisierte Unterfunktionen dieses Testprogramms ausführen. Der Hauptcontroller selbst ist ein 32-bit Mikroprozessor, der mit 80 Mhz getaktet wird. Er verfügt über eine 4-stufige Pipeline, eine Floating-Point-Einheit sowie 16Kb Instruktionscache und 8Kb Datencache. Pro Instruktion werden im Schnitt 2 Takte benötigt, dies kann jedoch aufgrund der Pipeline abweichen. Die im RAM des Controllers ausgeführte Firmware kann samt Variablen maximal 32Kb belegen. Das Kompilieren der Firmware auf dem Laptop übernimmt

ein kommerzieller, in die Cygwin-Laufzeitumgebung eingebetteter Cross-C-Compiler für TriCore Controller (High-Tec).

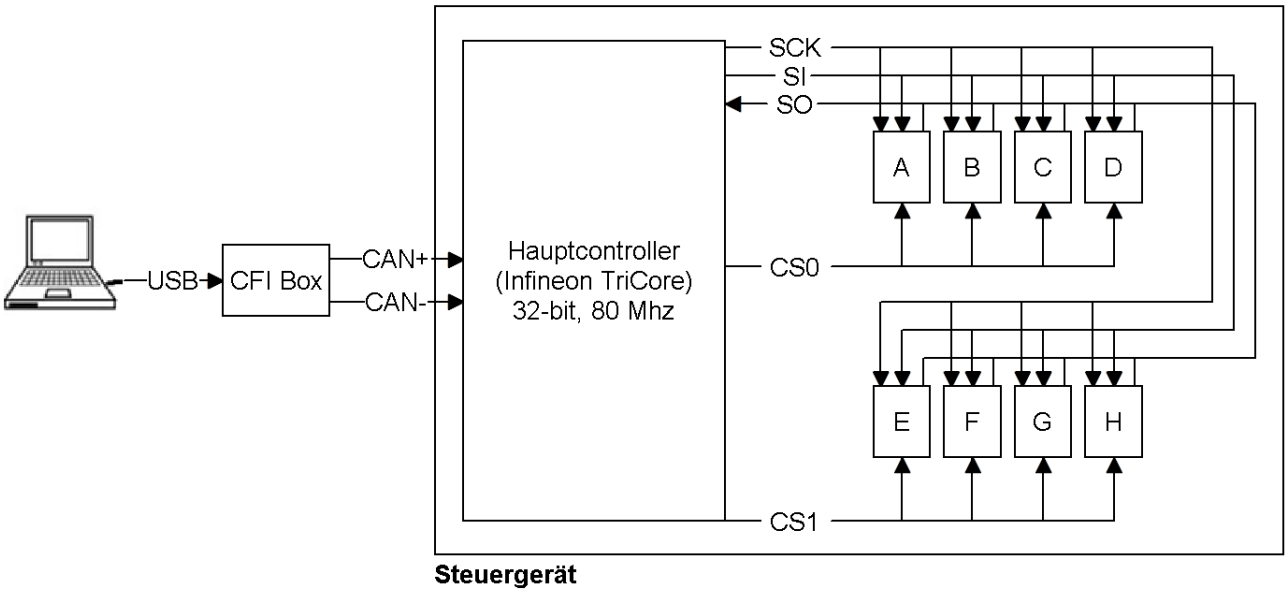


Abbildung 3.1: Entwicklungsaufbau

Innerhalb des Steuergeräts stellt der Hauptcontroller den Master des SPI-Busses dar. Einzelne ASICs (A-D, E-H) sind als Slaves angebunden und werden in einer durch Adress-Multiplexing erweiterten Sternschaltung angesprochen. Dabei werden bis zu 4 Slaves gleichzeitig über ein Chipselect-Signal aktiviert (vgl. ASIC A-D durch CS0). Die ersten zwei Bits des nun beginnenden Datenframes dienen der weiteren Adressierung. Ein Slave treibt demnach erst dann das MISO-Signal, wenn diese zwei Bits des Datenframes bereits übertragen wurden und mit der eigenen Adresse übereinstimmen. Die ASIC-spezifische SPI-Adresse ist entweder in einem internen EEPROM hinterlegt oder auch fest verdrahtet. Diese Maßnahme verringert die Anzahl benötigter Chipselect-Signale um den Faktor 4. Der Verdrahtungsaufwand im Layout des Steuergeräts ist ebenfalls kleiner. Dafür werden zwei von 16 Bits eines Datenframes reserviert, die Übertragungsrate wird demnach um $\frac{1}{8}$ erhöht.

3.3 Protokollansteuerung JTAG@SPI

Im folgenden soll der interne SPI-Bus zur JTAG-Kommunikation mit den ASICs genutzt werden. Dazu enthält der hier untersuchte ASIC SPI-Befehle, die das serielle Interface abschalten und die internen JTAG-Signale {TCK, TMS, TDI, TDO} auf den Pins des seriellen Anschlusses {SCK, CS, MOSI, MISO} verfügbar machen. Probleme ergeben sich hier aufgrund der Nutzung des Chipselect-Signals in Kombination mit Adress-Multiplexing. Es muss verhindert werden, dass die JTAG-Kommunikation

mit einem zu testenden ASIC (A) von anderen am gleichen Chipselect betriebenen ASICs (B-D) nicht als SPI-Datenframes interpretiert (und evtl. auf dem MISO-Signal beantwortet) wird.

3.3.1 Entwurf

Der JTAG@SPI-Testmodus muss zunächst durch die SPI-Befehle ENABLE_TAP und SWITCH_IF aktiviert werden. Als Parameter werden mehrere Schlüsselwerte übertragen. Dieses Vorgehen gilt nicht etwa der Verhinderung von Reverse Engineering, sondern der Sicherheit. Ein wichtiger Aspekt bei der Einführung von Offline-Testmodi ist, dass diese unter keinen Umständen per Zufall aktiviert werden. Ein unvorhergesehender, im Fahrbetrieb ausgelöster Testmodus würde sich wie ein Komplettausfall der entsprechenden Komponente verhalten. Bei älteren ICs wurde dieses Problem bisher durch eine hardwarebasierte Deaktivierung aller Testmodi verhindert. Wenn das TRST-Signal des JTAG-TAPs als Pin verfügbar ist, kann dieser hierzu fest auf Ground verdrahtet werden.

Im JTAG@SPI-Modus bestimmt die Zuweisung der JTAG-Signale zu den SPI-Signalen maßgeblich die Geschwindigkeit der JTAG-Kommunikation. Hierbei sind zwei Varianten denkbar:

SPI Signal	JTAG Signal
CS	TCK
SCK	TMS
MOSI	TDI
MISO	TDO

Tabelle 3.1: JTAG@SPI: Signal-Mapping A

SPI Signal	JTAG Signal
CS	TMS
SCK	TCK
MOSI	TDI
MISO	TDO

Tabelle 3.2: JTAG@SPI: Signal-Mapping B

Vorteil der ersten Variante (Mapping A) ist, dass das Chipselect-Signal durch Kopplung mit dem Testtakt (TCK) in jeder Taktperiode deaktiviert (und wieder aktiviert) wird. Dies sorgt bei anderen ASICs mit gleichem Chipselect-Signal für eine Blockierung der SPI-Kommunikation. Gleichzeitig

hat dieses Mapping einen großen Nachteil: ein halber JTAG Takt muss immer der notwendigen Holding-Zeit des Chipselect-Signals (250ns) auf SPI-Seite entsprechen. Hieraus ergibt sich für den JTAG-Takt eine minimale Periodendauer von 500ns und somit eine maximale Taktfrequenz von 2 Mhz.

Das im hier untersuchten ASIC bereits eingesetzte Mapping B dagegen erlaubt eine andere Ansteuerungsmöglichkeit: da das Chipselect-SPI-Signal asynchron arbeitet, kann es ohne Änderung des JTAG-Taktsignals TCK kurzzeitig deaktiviert und wieder aktiviert werden. Im SPI-Modus betriebene ASICs gehen anschließend von einem neuen Datenframe aus. Diese Aktion wird von hier an als asynchroner Chipselect-Reset bezeichnet. Bezogen auf die Geschwindigkeit ist auch hier die Chipselect-Holding-Zeit die ausschlaggebende Größe. Während beim Mapping A jeder JTAG-Takt automatisch einen neuen SPI-Datenframe einleitet, muss beim Mapping B die JTAG-Kommunikation pro Periode zur asynchronen Deaktivierung/Aktivierung des Chipselect-Signals angehalten werden. Dennoch ermöglicht Mapping B auch eine schnellere Ansteuerung: sobald auf SPI-Frames wartende ASICs nämlich die ersten zwei Adressbits des SPI-Datenframes empfangen haben und diese nicht mit der eigenen Adresse übereinstimmen, ignorieren sie den Rest des Datenframes bis zum erneuten Deaktivieren des Chipselect-Signals. Dies ist insbesondere bei langen Shiftvorgängen vorteilhaft, da hier das TMS/CS-Signal konstant auf 0 bleibt, der Chipselect (auch für bis zu 3 weitere ASICs) ist damit aktiv. Das Ziel dabei ist, die JTAG-Kommunikation auf SPI-Ebene möglichst als an den zu testenden ASIC adressierte Datenframes aussehen zu lassen.

3.3.2 SPI-Blocker

Auch im Hinblick auf weitere Anwendungsmöglichkeiten (vgl. softwarebasierter Scan Test, Kapitel 3.4) ist die hier vorgeschlagene Lösung möglichst generisch gehalten. Der vorgestellte SPI Blocker ermöglicht jegliche nicht per Chipselect getaktete Testkommunikation auf dem SPI-Bus zwischen Hauptcontroller und einem ausgewählten ASIC. Im eingeschalteten Zustand verhindert der Blocker, dass weitere im SPI-Modus am gleichen Chipselect betriebene ASICs Datenframes empfangen können. Hierzu werden zufällig, durch die TJAG-Kommunikation zustandekommende, nicht an den zu testenden ASIC adressierte SPI-Frames nach dem ersten Adressbit bzw. vor dem zweiten Adressbit durch Chipselect-Resets abgebrochen.

Bild 3.2 zeigt das Blockschaltbild des in Software realisierten SPI Blockers. Die Ansteuerung der SPI-Ausgangssignale SCLK, CS und MOSI wird den Einstellungen entsprechend abgewandelt. Diese Funktionalität wird durch die enable-Variable aktiviert bzw. deaktiviert. Bei enable=false werden die SPI-Signale direkt und ohne Veränderung auf den Signalleitungen des seriellen Interface ausgegeben. Die mindelay-Variable (32-bit) legt eine minimale, pro JTAG-Takt einzuhaltene Verzögerung (in ns) fest. Die eigentliche Übertragungsgeschwindigkeit hängt zusätzlich noch von der Ausführungszeit ab, die der Controller bei der softwarebasierten Ansteuerung benötigt. Durch die mode-Einstellung kann der SPI Blocker in Normalmodus oder JTAG-Modus betrieben werden. Letzterer umfasst eine auf das JTAG-Protokoll zugeschnittene Optimierung, die später anhand von Signalverläufen verdeutlicht wird. In der Firmware wird der SPI Blocker statt den Makros zum Setzen der SPI-Ausgangssignale aufgerufen. Das interne serielle Interface ist dabei abgeschaltet. Als Parameter dient ein Signalvektor

für {SCK, CS, SI}. Zur Erkennung von gesetzten, aber noch nicht ausgeführten, steigenden Flanken auf dem SCK-Signal ist in der Software eine Variable mit dem vorherigen SCLK-Zustand vorgesehen. Der Eingang der seriellen Master-Schnittstelle (MISO) wird hier nicht einbezogen. Bei SPI-Übertragungen ist das serielle Interface aktiv, der SPI Blocker muss dazu per enable deaktiviert sein.

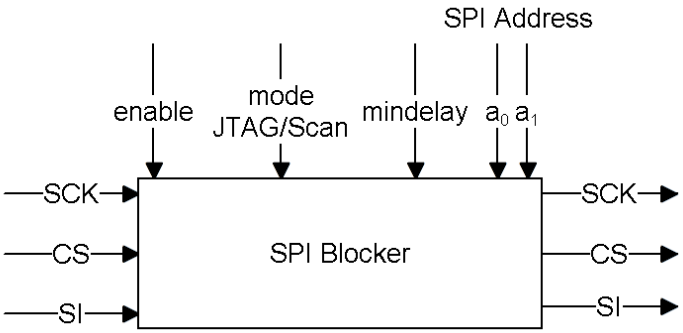


Abbildung 3.2: Blockschaltbild SPI Blocker

Der im SPI Blocker umgesetzte Zustandsautomat wird in Bild 3.3 dargestellt. Es handelt sich hierbei um einen deterministischen, endlichen Automaten. Die Übergänge zwischen den Zuständen passieren vor einer steigenden Flanke auf dem SCLK-Signal. Die Nummerierung im Bild deutet an, welches Bit innerhalb des SPI-Datenframes hier vorbereitet wird. Vor einer steigenden Flanke mit deaktivierten Chipselect-Signal geht der Automat immer in den Startzustand NO-FRAME über. Vor einem Taktschlag mit aktivem Chipselect-Signal wird dieser Zustand verlassen. Sofern das zu setzende erste Adressbit des Datenframes mit der eingestellten Adresse übereinstimmt, wird der Zustand a_0 eingenommen, ansonsten der Zustand $\overline{a_0}$. Im zweiten Fall wird nach dem Taktschlag ein asynchroner Chipselect-Reset durchgeführt. Im nächsten JTAG-Takt folgt dann erneut ein erstes Adressbit mit den bereits bekannten Übergängen in die Zustände a_0 oder $\overline{a_0}$. Falls das erste Adressbit der eingestellten SPI-Adresse entspricht, befindet sich der Automat im Zustand a_0. Vor einem falschen zweiten Adressbit wird nun ebenfalls ein asynchroner SPI-Reset durchgeführt. Dieser sorgt beim nächsten Bit wieder für Übergänge nach a_0 und $\overline{a_0}$. Erst wenn das zweite Adressbit ebenfalls mit der geforderten SPI-Adresse übereinstimmt folgt aus dem Zustand a_0 der Übergang in den Zustand IN-FRAME. Hier kann der Automat - solange Chipselect nicht deaktiviert werden soll - verbleiben; seriell am gleichen Chipselect betriebene ASICs ignorieren den Rest des bereits adressierten Datenframes, welcher somit nicht mehr an Zeitvorgaben des SPI-Protokolls gebunden ist.

Wird der SPI Blocker im JTAG-Modus (mode=JTAG) betrieben, ist eine Optimierung in Form des JTAG Speedups möglich. Dabei wird ausgenutzt, dass dem mit SI überlagerten TDI-Signal bei der JTAG-Kommunikation - sofern der TAP sich nicht in den Zuständen SHIFT_DR oder SHIFT_IR befindet - keine Bedeutung zukommt. Die Adressbits können somit in den meisten Fällen injiziert werden, ohne die JTAG-Kommunikation zu stören. Dies sorgt dann beim ersten Adressbit für einen Übergang in den Zustand a_0 bzw. beim zweiten Bit für einen Übergang in den Zustand IN-FRAME. Bei Betrachtung des TAP-Zustandsautomats fällt auf, dass so die beiden o-Transitionen zwischen

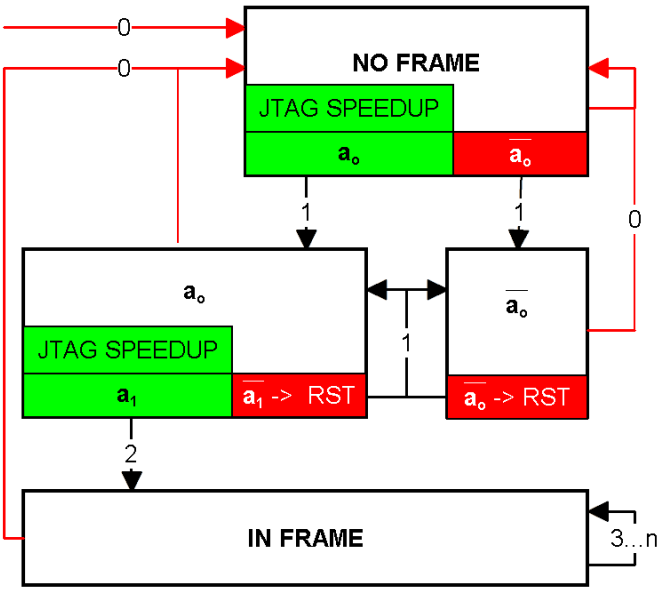


Abbildung 3.3: Zustandsautomat SPI Blocker

SELECT_DR und SHIFT_DR zur Adressierung eines SPI-Datenframes genutzt werden können, welcher den gesamten, folgenden Shiftvorgang (im Zustand SHIFT_DR) abdeckt. Gleiches gilt für das Instruktionsregister, wobei dieses meist jedoch nicht von langen Shiftoperationen betroffen ist (typische Instruktionslänge: 4-16 Bit). Bei JTAG-Kommunikation im JTAG-Modus ist mit erheblich weniger asynchronen Chipselect-Resets zu rechnen. Bei den typischen JTAG-Shiftoperationen sind gar keine Chipselect-Resets mehr nötig.

3.3.3 Typische Signalverläufe

Bild 3.4 verdeutlicht, wie eine typische JTAG-Kommunikation durch den SPI Blocker abgewandelt wird:

Signalverlauf 1

Der oberste Verlauf zeigt eine direkte JTAG-Kommunikation ohne die Nutzung von SPI-Bus-Signalen. Das Instruktionsregister wird mit der Sequenz 01001 beschrieben. Dazu wird der TAP zunächst einmal durch die TMS-Sequenz 01100 vom Zustand RESET in den Zustand SHIFT_IR versetzt, was zum Zeitpunkt t=5 geschehen ist. Es folgen 5 weitere Taktschläge im Zustand SHIFT_IR, wobei die gewünschte Instruktion 01001 auf TDI eingespielt wird, während auf dem TDO-Ausgang der vorherige, im CAPTURE-Zustand gesetzte Inhalt des Instruktionsregisters sichtbar wird (hier: 10000).

Der Shiftvorgang wird durch die TMS-Sequenz 110 abgeschlossen, welche den TAP vom SHIFT_IR-Zustand in den IDLE-Zustand bringt.

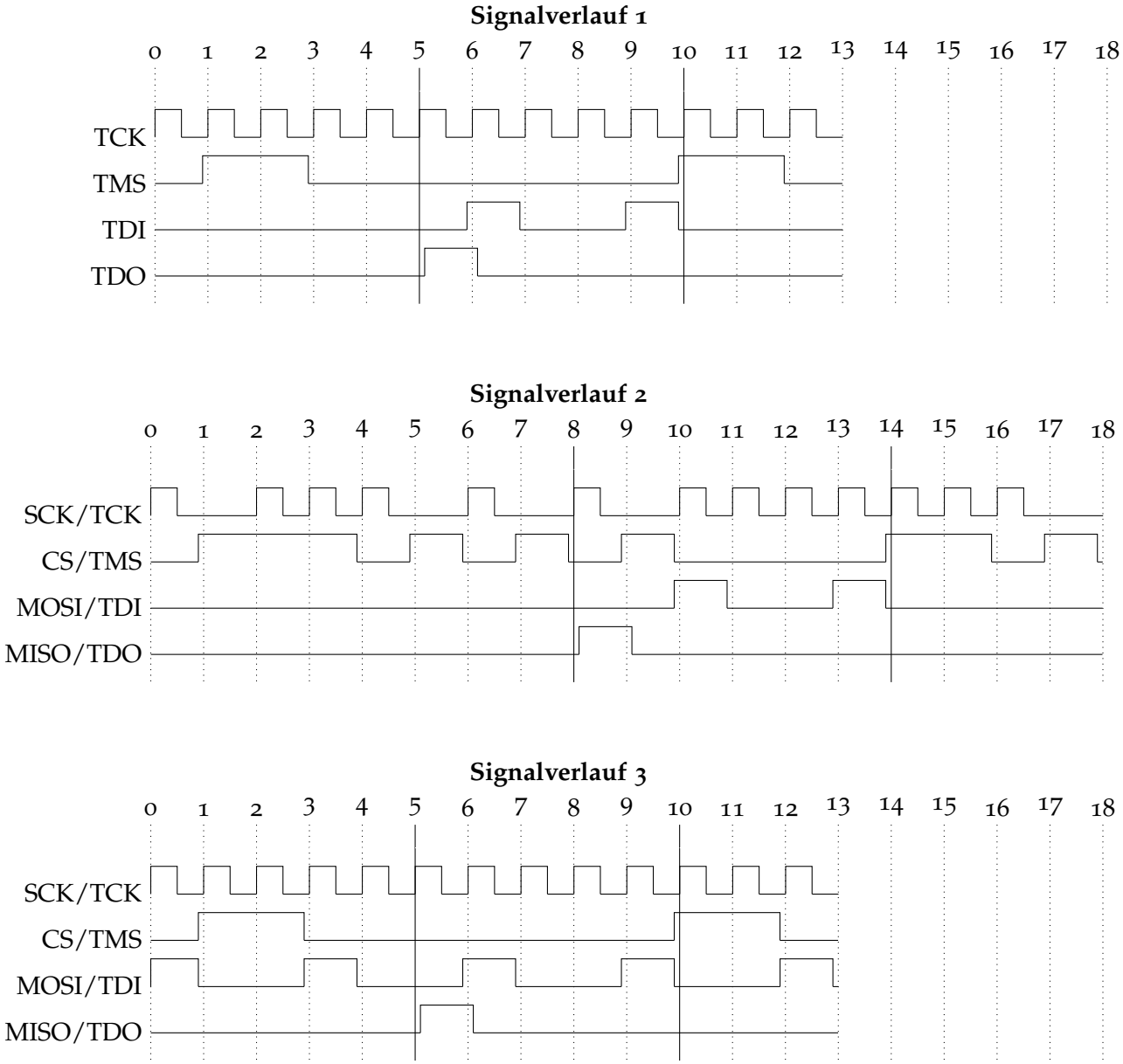


Abbildung 3.4: JTAG@SPI: Signalverläufe 1-3 beim Ausführung der JTAG-Instruktion 01001

Signalverlauf 2

Beim zweiten Verlauf geschieht die gleiche JTAG-Kommunikation, diesmal jedoch per JTAG@SPI im normalen Modus. Die SPI-Adresse des Ziel-ASICs ist ist hierbei $a_0 = 1$, $a_1 = 0$. Man erkennt schon am Taktsignal, dass im gesamten Verlauf 5 asynchrone Chipselect-Resets durchgeführt werden. Dies geschieht, sobald das Chipselect-Signal mindestens einen Takt lang aktiv war (TMS=0) und das SI-Signal bei der steigenden Flanke nicht mit dem ersten Adressbit (a0) übereinstimmte. Die Überführung des TAPs vom Zustand RESET in den Zustand SHIFT_IR benötigt so wieder 5 Taktschläge, zusätzlich aber noch 3 Chipselect-Resets. Erst zum Zeitpunkt t=8 ist der Zustand SHIFT_IR erreicht. Es folgt der Shiftvorgang, bei dem das Instruktionsregister mit 01001 befüllt wird. Das erste geshiftete Bit erzeugt zunächst wieder einen darauffolgenden Chipselect-Reset, da es nicht mit dem ersten Adressbit übereinstimmt. Das zweite und dritte Bit der Instruktion entsprechen der ASIC-SPI-Adresse 10, so dass der SPI-Blocker nun in den IN-FRAME-Zustand wechselt und dort bis zur nächsten Deaktivierung des Chipselect-Signals zum Zeitpunkt t=14 verweilt. Bei der folgenden TMS-Sequenz 110 zum Erreichen des TAP-Zustands IDLE wird nach dem letzten Taktschlag zum Zeitpunkt t=17 erneut ein CS-Reset ausgelöst, da das SI-Signal zum Zeitpunkt t=16 (0) nicht dem ersten Adressbit entspricht.

Signalverlauf 3

Der letzte Signalverlauf demonstriert erneut die gleiche Operation, diesmal befindet sich der SPI Blocker jedoch im JTAG-Modus. In dieser Einstellung kommt es nur selten zu einem Chipselect-Reset, da in 14 von 16 TAP-Übergängen mit TMS=0 das TDI-Signal frei gesetzt werden kann. Der SPI-Blocker injiziert an diesen Stellen das jeweils gewünschte Adressbit. Der Zeitpunkt t=0 markiert dabei den Übergang des TAPs vom RESET-Zustand in den IDLE-Zustand. Hierzu muss TMS für einen Takt auf 0 gesetzt werden, das Chipselect-Signal ist demnach aktiv. Da das TDI-Signal an diesem Übergang keine Bedeutung hat, setzt es der SPI Blocker auf das erste Adressbit ($a_0 = 1$). Ein anschließender Chipselect-Reset erübrigt sich. Zum Zeitpunkt t=3 wird das Chipselect-Signal erneut aktiviert. Der TAP wechselt hier vom SELECT_IR-Zustand in den CAPTURE_IR-Zustand. Bis zum Ende des Shiftvorgangs bleibt das TMS-Signal unverändert. Dies bedeutet, dass das Chipselect-Signal schon 2 Takte vor dem eigentlichen Shiftvorgang bis zu dessen Ende aktiviert bleibt. Im Shiftvorgang kann das TDI-Signal nicht frei gesetzt werden, bei den zwei vorherigen TAP-Übergängen von SELECT_IR nach CAPTURE_IR bzw. von CAPTURE_IR nach SHIFT_IR dagegen schon. An diesen Stellen (t=3, t=4) injiziert der SPI Blocker die gewünschte SPI-Adresse auf dem SI-Signal. Der komplette Shiftvorgang von t=5 bis t=10 wird somit von anderen, im SPI-Modus am gleichen Chipselect-Signal betriebenen ASICs ignoriert. Es folgt ab t=10 erneut die TMS-Sequenz 110, um den TAP vom SHIFT_IR-Zustand in den IDLE-Zustand zu bringen. Im letzten Takt (t=12) wird das Chipselect-Signal erneut aktiviert; der SPI Blocker injiziert vorsichtshalber wieder das erste Adressbit.

3.4 Softwarebasierter Scan-Test

3.4.1 Entwurf

Sofern ein ASIC nicht aufgrund seines komplexen Aufbaus mit einem Logik-BIST ausgestattet ist, wird am Ende der ASIC-Produktion ein von außen kontrollierter Scan-Pattern-Test durchgeführt (vgl. Bild 3.5). Hierzu wird der ASIC zunächst über bestehende Interfaces (SPI, JTAG) in einen durch mehrere Schlüssel abgesicherten Testmodus umgeschaltet. Wird hier ein falscher Schlüssel übertragen, so wird der Testmodus bis zum nächsten Hardware-Reset unwiderruflich deaktiviert. Im Testmodus dienen umfunktionierte IC-Pins als Eingänge und Ausgänge der internen Scan-Ketten. Die Generierung der Testdaten als auch die Kompaktierung der Antwortsignale findet extern durch ATE (Automated Test Equipment) statt. Die Testvektoren werden automatisch durch ein ATPG-Tool (Automatic Test Pattern Generator) erstellt. Im sogenannten Test-Per-Clock-Verfahren wird bei jedem Taktschlag ein Testvektor an den Eingängen angelegt und der resultierende Antwortvektor überprüft. Ein derartiger Scan-Test soll nun im Rahmen des Feldtests softwarebasiert vom steuergerät-internen Hauptcontroller aus durchgeführt werden.

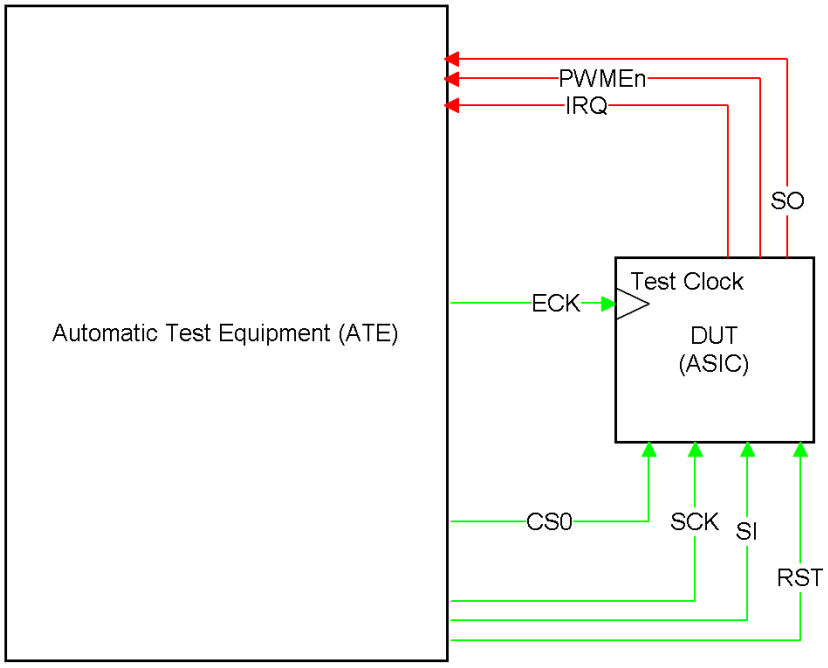


Abbildung 3.5: Scan-Test in der Produktion

3.4.2 Anwendung

In Bild 3.6 sind die für den implementierten Test relevanten Steuergerät-Komponenten dargestellt. Die zu testende Komponente (Device Under Test, DUT) muss zunächst über das SPI-Interface in den JTAG@SPI-Modus geschaltet werden. Mit JTAG-Befehlen ist nun der eigentliche Scanmodus zu aktivieren. Die Signale {ECK, CS0, SI, SCK, RST} bilden die Testeingänge; {PWMEn, IRQ, SO} sind die vom Hauptcontroller zu prüfenden Antwortsignale. Dabei dient das ECK-Signal als Testtakt. Im ursprünglichen Scan-Test bei der ASIC-Produktion wird ein zusätzlicher Pullup-Transistor am IRQ-Signal betrieben. Für die Ansteuerung des Gate-Signals ist ein Eintrag in den Testvektoren vorgesehen. In dieser Implementierung wird der zusätzliche Transistor durch den controller-internen Pullup am entsprechenden Eingang ersetzt. Dieser weist allerdings einen anderen Widerstand zur Versorgungsspannung auf.

Das Testprogramm auf dem Hauptcontroller ist aus mehreren Gründen auf den im vorherigen Kapitel implementierten SPI Blocker angewiesen. Zunächst muss der DUT per JTAG@SPI in den entsprechenden Testmodus gebracht werden. Da Signale vom SPI-Bus als Testeingänge fungieren, muss er ebenfalls während des Tests eingeschaltet sein. Ein Problem dieser Implementierung stellt der nicht direkt ansteuerbare Testeingang RST dar. Zum Setzen des RST-Eingangs ist es notwendig, den Scantest anzuhalten und SPI-Nachrichten an einen weiteren ASIC (Stabilisator) zu senden. Währenddessen ist das serielle Interface des Hauptcontrollers in Benutzung, der SPI Blocker wird abgeschaltet. Im hier untersuchten Fall wird der Stabilisator über ein anderes Chipselect-Signal (CS1) aktiviert, ein Aufbau mit gleichem Chipselect für den zu testenden ASIC und den Stabilisator ist jedoch auch denkbar.

Der eigentliche Testverlauf beginnt, nachdem der zu testende ASIC in den Scanmodus versetzt wurde. Die einzelnen Testvektoren werden bei eingeschaltetem SPI Blocker (enable=true, mode=normal) an den Testeingängen gesetzt, die Antwortsignale werden nach einem Takt auf dem ECK-Signal vom Hauptcontroller überprüft. Eine interne Variable speichert, ob bereits ein falsches Antwortsignal beobachtet wurde. Der Testeingang RST hält im laufenden Betrieb den Zustand 1. Wenn nun ein Testvektor mit RST=0 angewandt werden soll, muss der normale Testablauf (getaktet durch das ECK-Signal) zunächst angehalten werden. Nun wird der SPI-Blocker deaktiviert (enable=false), so dass eine SPI-Nachricht an den Stabilisator gesendet werden kann. Als Folge wird das RST-Signal für 67 μs in den Zustand 0 versetzt. In dieser Zeit aktiviert der Hauptcontroller wieder den SPI Blocker und führt einen Takt auf dem ECK-Signal durch. Eine zusätzliche Wartezeit garantiert, dass das RST-Signal für den nächsten Testvektor wieder den Wert 1 annimmt. Bei mehreren Testvektoren mit RST=0 wird dieses Vorgehen jedesmal wiederholt.

Diese Anwendung ist im Hinblick auf die Fehlerabdeckung durch das Layout des Steuergeräts und den nicht direkt ansteuerbaren Testeingang (RST), begrenzt. Im Produktionstest dient der Scan-Test nicht nur der Erkennung von Stuck-At-Faults, sondern auch von Transition-Delay-Faults. Gerade die zweite Fehlerklasse ist aber auf schnelles Testequipment und vor allem konstante Testfrequenz angewiesen, welche hier nicht gegeben ist. Diese Implementierung ist somit auf das statische Stuck-At-Fehlermodell begrenzt. Dabei sind die Abweichungen von einer konstanten Testgeschwindigkeit nicht nur durch den RST-Testeingang begründet. Ein weiteres Bottleneck ist die Übertragung der Testvektoren vom Laptop zum Hauptcontroller. Zum einen ist das hier genutzte

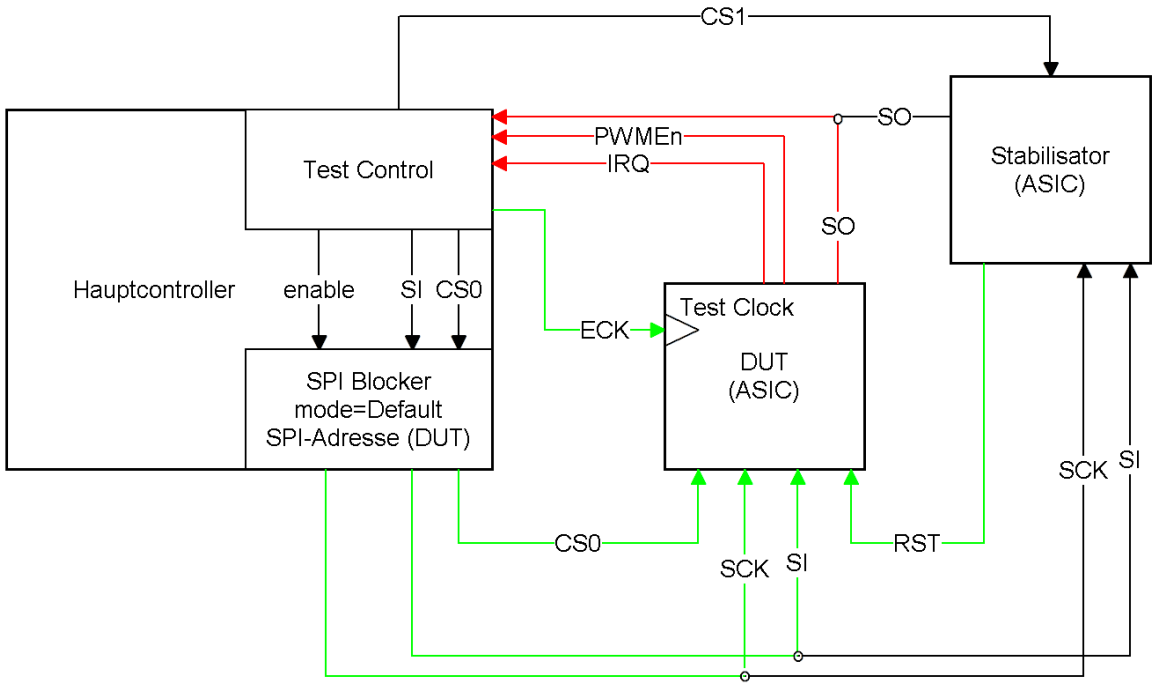


Abbildung 3.6: softwarebasierter Scan-Pattern-Test

CF-Inferface paketorientiert (255-Byte pro Datenpaket), zum anderen ist das Zwischenspeichern im Hauptcontroller auf 28Kb begrenzt, sofern 4Kb für das Programm reserviert bleiben. Um diesen Test für Transition-Delay-Faults zu erweitern würde ein sehr schnelles Interface zur Übertragung der Testvektoren, direkt ansteuerbare Testeingänge, eine hardwareunterstützte Ausgabe auf den entsprechenden Pins sowie Ersatz für den softwarebasierten SPI Blocker benötigt.

3.4.3 Konvertierung der Testdaten

Die Testdaten für den Scan Test können als ASCII-kodierte Tabelle aus TetraMax exportiert werden. Jede Zeile enthält dabei eine Timing-Einstellung (Periodendauer in ns) sowie einen Testvektor, der in einem Takt angewendet wird (vgl. Scan-Per-Clock). Da sich diese speicherlastige Darstellungsform nicht zur Übertragung zwischen Laptop und Hauptcontroller eignet, müssen die Daten entsprechend umkodiert werden. Bild 3.7 verdeutlicht die verschiedenen Darstellungsformen. Die zwei Konvertierungschritte dazwischen sind in den Perl-Skripten PAT2BIN.PL und BIN2CFI.PL realisiert.

Zunächst werden die einzelnen Testvektoren binär kodiert. Im hier untersuchten Fall bestehen die Testdaten pro Taktschlag aus 6 Eingangssignalen (Zustand 0, 1) und 3 zu überprüfenden Ausgangssignalen (Zustand High, Low, Don't Care). Es sind somit $6 \cdot 1 + 3 \cdot 2 = 12$ Bit zur Kodierung eines Testvektors nötig. Dies wird zur vereinfachten (und schnelleren) Anwendung im Hauptcontroller

auf 16 Bit (2 Byte) aufgerundet. Das PAT2BIN.PL-Skript liest die ASCII-Tabelle und konvertiert jede Zeile. Dabei stehen die ersten 6 Bit (Masken 0x0000 - 0x0020) für die Eingangssignale, weitere 6 Bits (Masken 0x0040-0x0200) umschreiben für die 3 Ausgangssignale, ob sie defininiert sind ($\simeq$ keine Don't Cares sind) und welchen Wert sie dann aufweisen. Obwohl diese Implementierung bei der Wahl der Testgeschwindigkeit stark eingeschränkt ist und diese nicht konstant garantieren kann, werden die Testvektoren in Blöcken mit je einer minimalen Timing-Einstellung gruppiert.

Das BIN2CFI.PL-Skript liest im Anschluss diese Binärdarstellung und bettet sie in eine CFI-Skript-Vorlage ein. Diese Vorlage schaltet den ASIC zunächst per JTAG@SPI-Befehlen in den Testmodus. Im Anschluss werden pro Steuerungsnachricht je 125 Testvektoren übertragen. Dies ergibt sich aus dem maximalen Größe einer Nachricht (255) sowie 4 vorgestellten Bytes zur Auswahl der Firmwarefunktion. Das Testprogramm im Hauptcontroller beantwortet jede dieser Nachrichten mit einem Zustandsbit, je nachdem ob bereits ein Fehlverhalten in Form einer falschen Testantwort aufgetreten ist. Vor jedem Block an Testvektoren wird die gemeinsame Timing-Einstellung als separater Steuerungsbefehl umgesetzt. Das gefüllte CFI-Skript kann nun in Kombination mit der Test-Firmware für den Hauptcontroller am Laptop des Testaufbaus ausgeführt werden. Eine Prüfung der letzten Antwortnachricht am Ende des Skripts gibt Aufschluss über das Testergebnis.

Die angegebenen Dateigrößen ergeben sich durch einfache Abschätzung: Im ASCII-Ausgangsformat benötigt ein kompletter Testvektor samt eingefügten Kommentaren eine Zeile mit $\sim$ 130 Bytes. Bei 750.000 Testvektoren ergeben sich so ca. $\sim$ 100Mb. In der Binärdarstellung werden 2 Bytes pro Testvektor gespeichert, Header- und Timing-Angaben sind zu vernachlässigen. Sie ist daher mit ca. $\sim 1,5$ Mb die kleinste Darstellung. Im CFI-Skript ergibt sich für 125 Testvektoren eine ASCII-kodierte Zeile mit hexadezimaler Darstellung der Testvektoren ($2 \cdot 250$ Byte) sowie ein Overhead von ca. $\sim$70 Byte. Daraus folgt eine geschätzte Speichergröße von $\sim 3,5$ Mb.

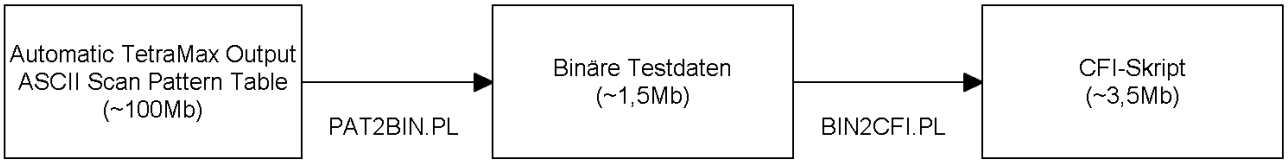


Abbildung 3.7: Testdaten-Kodierungen

3.5 Built-In Self-Test (BIST) für Taktsignale

Eine steuergerät-interne Taktquelle kann nur mithilfe einer zusätzlichen Testschaltung oder durch einen Abgleich mit anderen, bereits getesteten Taktquellen überprüft werden. Da derartige Referenzquellen fehlen, soll hier zunächst der Systemtakt des Hauptcontrollers durch eine hardwarebasierte Lösung getestet werden. Dazu dient ein per PWM (Pulsweitenmodulation) ange-

steuerter Digital-Analog-Wandler (DAC, Digital-Analog-Converter) mit einem nachgeschalteten Analog-Digital-Wandler (ADC, Analog-Digital-Converter). Eine spätere, softwarebasierte Implementierung (vgl. Softwarebasierter Scan-Test, Kapitel 3.6) verfügt somit über eine Referenz für den softwarebasierten Abgleich weiterer, steuergerät-interner Taktquellen.

3.5.1 Entwurf

Die Genauigkeit von Taktsignalen ist besonders bei mehreren Taktquellen eine sehr wichtige Systemeigenschaft. Sie bestimmt, wie häufig es bei Synchronizationen jeglicher Art zu Übertragungsfehlern oder Messungenauigkeiten kommt. Bei der Prüfung von Taktsignalen sind folgende drei Signaleigenschaften zu betrachten:

Frequenzabweichung : Alle Kristalle weisen eine kleine, aber charakteristische Abweichung ihrer Frequenz auf. Die zu beobachtenden Effekte geschehen dabei meist „schleichend“, d.h. es kommt erst nach vielen Systemtakten zu einer merkbaren, aufsummierten Zeitdifferenz.

Jitter : Bei genauerer Betrachtung von Taktsignalen fällt ein „Zittern“ der Flanken auf. Das Auftreten der einzelnen Transitionen ist normalverteilt um die gewollten Zeitpunkte. Die Standardvarianz dieser Verteilung kann als Maßeinheit für diesen Jitter dienen.

Flankensteilheit : Das Signal steigt und sinkt nicht in unendlich kleiner Zeit, allein schon wegen Leitungskapazitäten. Es weist somit zwei Zeitspannen auf, die es benötigt, um die beiden binären Transitionen zu vollziehen.

Die in [CH96] vorgeschlagene Schaltung zur Generierung analoger Spannungen mithilfe eines durch PWM angesteuerten Tiefpasses ist in Bild 3.8 dargestellt. Die PWM-Frequenz steht hierbei in direktem Zusammenhang zur Taktfrequenz des Hauptcontrollers. Sie wird mithilfe des GPTA-Moduls (General Purpose Timer Array) des Hauptcontrollers erzeugt, wobei der Systemtakt des Controllers durch einen Vorteiler dividiert wird, bevor dieses Signal dann einen Zähler antreibt. Der hier eingesetzte Hauptcontroller arbeitet mit 80 MHz und erlaubt Teiler von 2,5,9 und 14. Als Randbedingung für die korrekte Ansteuerung des Tiefpasses muss der PWM-Ausgang über symmetrische Ausgangsimpedanz verfügen. Die vom Tiefpass generierte, analoge Ausgangsspannung ist abhängig von der erzeugenden PWM-Frequenz und kann mit einem controller-internen ADC zurückgemessen werden. Der zwischengeschaltete Impedanzwandler verringert die Ausgangsimpedanz des Signals, um genügend Strom für eine ADC-Messung zu liefern. Der ADC digitalisiert seine analoge Eingangsspannung in 10 Bit bei einer Referenzspannung von 3,3V (ca. 0,8mV/LSB). Es stellt sich nun die Frage, mit welcher Genauigkeit aus diesem Ergebnis Rückschlüsse auf die ursprüngliche Taktquelle möglich sind.

Das Übertragungsverhalten des Tiefpasses kann als Verstärkung in Abhängigkeit von der Eingangsfrequenz wie in Bild 3.9 dargestellt werden und berechnet sich zu:

$$H(f) = \frac{1}{\sqrt{1 + (2 \cdot \pi \cdot f \cdot R \cdot C)^2}}$$

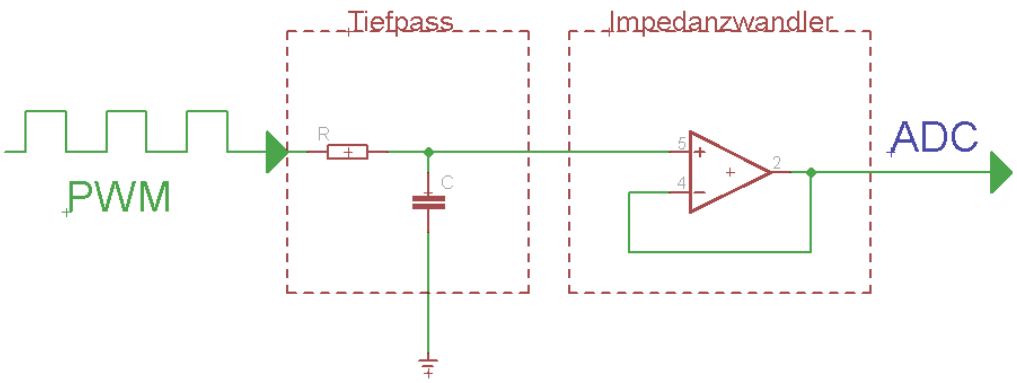


Abbildung 3.8: BIST für Taktsignale

Aufgrund der realen Anwendung wird hier auf logarithmische Achsen und normierte Frequenzen verzichtet. Der aufgezeigte Tiefpass besteht aus einem Widerstand R= 100Ohm und einem Kondensator C=100 nF. Die Bauteilvarianz dieser Einzelkomponenten wird hier zunächst mit ±5% angenommen und sorgt für eine obere und untere Schranke bei der Übertragungsfunktion. Als Beispiel soll eine Messfrequenz von 50KHz gelten. Hierfür wird ein Vorteiler von 5 sowie ein Zähler von 320 eingestellt ($\frac{80\ MHz}{5 \cdot 320} = 50KHz$). Durch positive und negative Bauteilvarianz ergeben sich so eine minimale und maximale Verstärkung:

$$H(50KHz)_{min} = \frac{1}{\sqrt{1 + (2 \cdot \pi \cdot 5000Hz \cdot (100Ohm \cdot 1,05) \cdot (100nF \cdot 1,05))^2}} = 0,27739$$

$$H(50KHz)_{max} = \frac{1}{\sqrt{1 + (2 \cdot \pi \cdot 5000Hz \cdot (100Ohm \cdot 0,95) \cdot (100nF \cdot 0,95))^2}} = 0,33262$$

Der PWM-Ausgang des Controllers schaltet zwischen 0V und 3,3V, die berechnete Verstärkung entspricht demnach einer am Ausgang des Tiefpasses messbaren Analogspannung von v_{min} = 0,915387V bzw. v_{max} = 1,097646V. Beim Rückschluss auf die generierende PWM-Frequenz müssen die Bauteilvarianzen erneut betrachtet werden. Da sie im realen Fall nicht bekannt sind, muss vom schlechtesten Fall ausgegangen werden. Demnach könnte die minimale Verstärkung auch durch die betragsmäßig größere Übertragungsfunktion (mit Varianz-Faktor 0,95) entstanden sein. Genauso könnte die maximale Verstärkung durch die kleinere Übertragungsfunktion (mit Varianz-Faktor 1,05) erzeugt worden sein. Dies ergibt eine minimale und maximale Grenze für die ursprüngliche Frequenz zu:

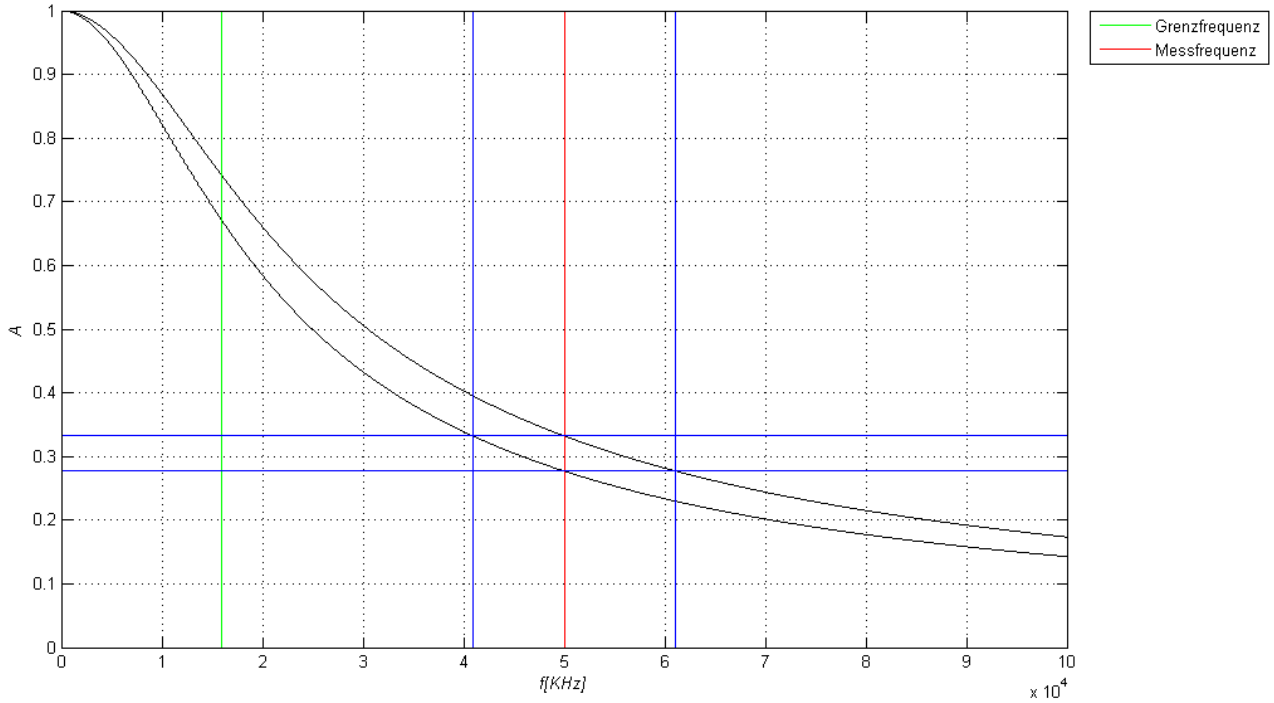


Abbildung 3.9: Übertragungsverhalten eines Tiefpass mit Bauteilvarianz (5%)

$$f(H = 0,33262)_{min} = \frac{\sqrt{\left(\frac{1}{H}\right)^2 - 1}}{2 \cdot \pi \cdot (100Ohm \cdot 1,05) \cdot (100nF \cdot 1,05)} = 40930Hz$$

$$f(H = 0,27739)_{max} = \frac{\sqrt{\left(\frac{1}{H}\right)^2 - 1}}{2 \cdot \pi \cdot (100Ohm \cdot 0,95) \cdot (100nF \cdot 0,95)} = 61080Hz$$

Diese Beispielrechnung zeigt, dass eine Bauteilvarianz von $\pm$5% bereits zu einer Messungenauigkeit bei der Frequenzbestimmung von -18,14% bzw. +22,16% führt. Die Abweichung nach oben ist durch die Form der Übertragungsfunktionen bedingt immer größer. Für alle Bauteilvarianzen bis $\pm$18% zeigt Bild 3.10 die größere Messungenauigkeit. Demnach führt eine erreichbare Bauteilvarianz von $\pm$1% zu einer akzeptablen Messungenauigkeit kleiner $\pm$5%.

Im Hinblick auf die zu betrachtenden Signaleigenschaften einer Takts kann die hier vorgestellte BIST-Methode nur Frequenzabweichungen feststellen. Ein normalverteilter Jitter oder zu geringe Flankensteilheit verändern die vom Tiefpass erzeugte Analogspannung nicht.

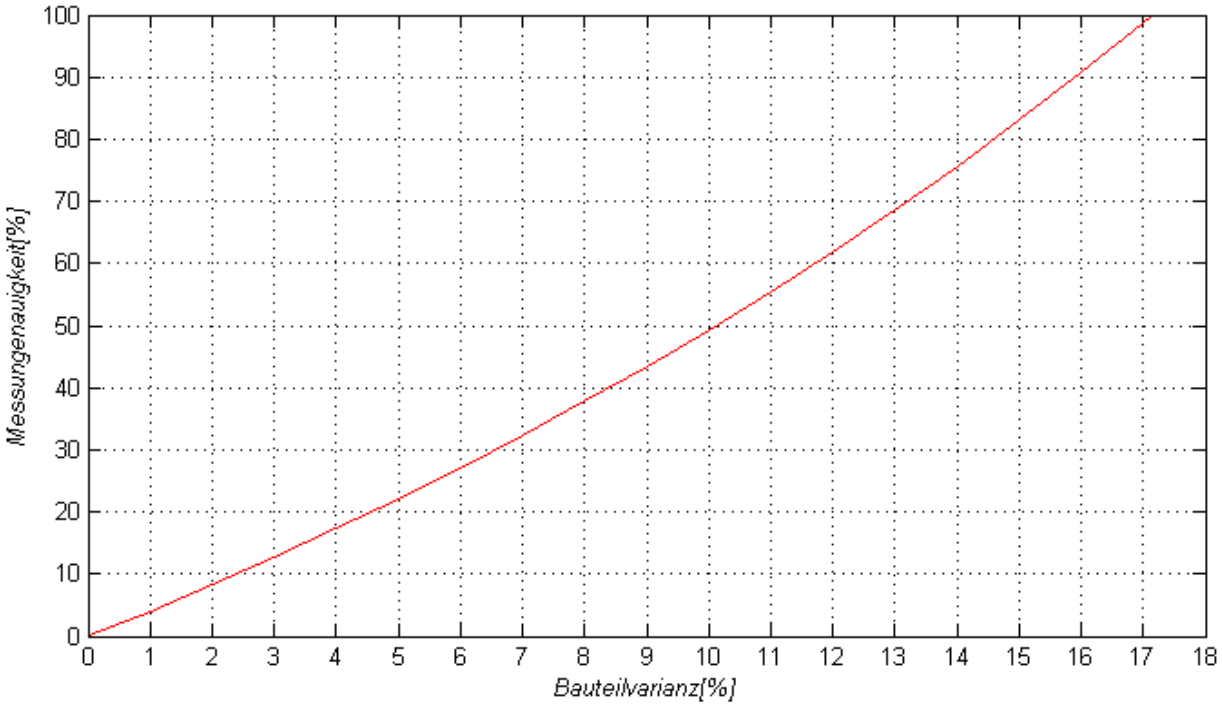


Abbildung 3.10: Messungenauigkeit bei der Frequenzbestimmung in Abhängigkeit der Bauteilvarianz

3.5.2 Auslegung der Bauelemente

Die Grenzfrequenz eines Tiefpasses liegt bei einer Verstärkung von $\frac{1}{\sqrt{2}}$ (ca. 70,71 %) vor und berechnet sich zu:

$$f_c = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

Diese sollte möglichst klein gewählt sein. Die Übertragungsfunktion nähert sich für große Frequenzen asymptotisch der 0, eine kleine Grenzfrequenz führt also zu einem möglichst großen Spektrum an nutzbaren Verstärkungen. Dies bedeutet, dass die Werte für R und C groß zu wählen sind, was weitere Vorteile mit sich bringt: ein großer Widerstand verringert den zurückfließenden Strom, wenn das PWM-Signal auf 0 liegt. Ein großer Kondensator kann mehr Energie speichern und an den Impedanzwandler abgeben.

Zum Test des Haupttakts auf Frequenzabweichung werden nun drei verschiedene Messfrequenzen angesetzt, möglichst gut verteilt über den Wertebereich der Verstärkungen. Bild 3.11 verdeutlicht die

angesetzten Messungen bei 1 KHz, 50KHz und 90KHz. Die Bauteile weisen nun eine Varianz von ±1% auf. Die ebenfalls im Bild markierten Abweichungen auf der Frequenzachse liegen somit unter ±5%.

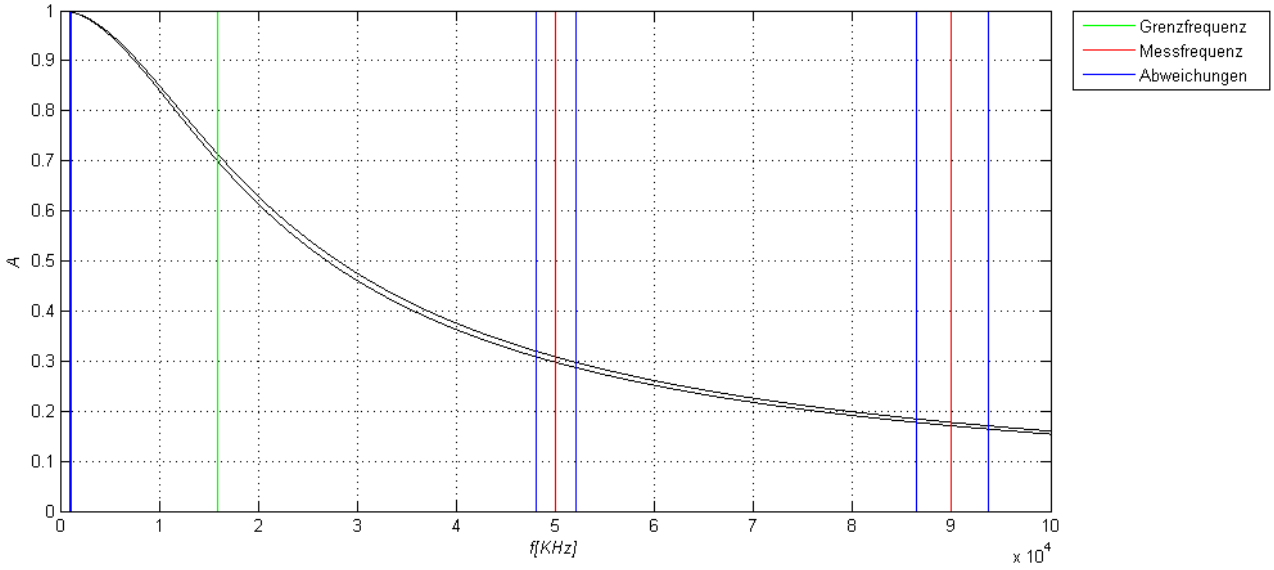


Abbildung 3.11: Messung

3.6 Softwarebasierter Abgleich weiterer Taktsignale

Nachdem das Taktsignal des Hauptcontrollers mit der vorherigen BIST-Methode geprüft wurde, kann es als Referenz für einen softwarebasierten Abgleich mit ASIC-internen PLL-Schleifen (Phase Locked Loop) dienen. Die Systemtakte der einzelnen ASICs (z.B. 25 MHz) werden durch einen vom Hauptcontroller generierten Haupttakt (typ. 1 MHz) sowie interner PLL-Schaltungen zur Teilung der Periodendauer gewonnen. Eine zu große Abweichung dieser ASIC-Systemtakte führt zu gleichem Fehlverhalten wie beim Takt des Hauptcontrollers. Da die ASICs im Gegensatz zum Controller meist mehr analoge Schaltanteile haben und häufiger zeitkritische Messungen durchführen, können die Folgen hier sogar noch schwerer wiegen. Ausserdem existieren pro Steuergerät bis zu 10 ASICs mit jeweils einer PLL-Schleife, die Auftrittswahrscheinlichkeit eines PLL-Defekts in einem Steuergerät ist daher umso größer.

3.6.1 Entwurf

Mithilfe des JTAG@SPI-Protokolls können ASIC-interne Taktsignale - unter anderem der Systemtakt - auf bestimmten Pins herausgeschaltet werden. Dieses Signal kann vom Hauptcontroller mit einem

Digitaleingang diskret abgetastet werden. Da die ASICs im Vergleich zum Hauptcontroller nur ungleich langsamer getaktet werden, bietet sich hier eine Unterabtastung an. Bei einzelnen, zeitlich verzögerten Messungen wird nicht nur der aktuelle Zustand des abzutastenden Signals, sondern auch der Zeitpunkt der Messung aufgezeichnet. Anhand dieser Zeitangabe kann ermittelt werden, an welchem Punkt in der zu testenden Taktperiode die Messung stattgefunden hat. Zunächst ist folgende Unterscheidung notwendig:

Ein Messpunkt (abgekürzt mp) stellt eine konkrete Einzelerfassung dar. Sie besteht aus dem Auftrittszeitpunkt sowie dem in diesem Moment am Testtakt anliegenden Signalzustand (z = { 0, 1 }). Der Zeitpunkt der Messung wird mithilfe des controller-internen Taktzählers bestimmt und kann durch Kenntnis der Controller-Taktfrequenz auch in eine Zeitangabe (in Nanosekunden) umgerechnet werden.

Ein Testpunkt (abgekürzt tp) ist ein Zeitpunkt innerhalb der zu beobachtenden Taktperiode. Zu jedem Zeitpunkt einer Messung kann wie folgt der entsprechende Testpunkt ermittelt werden:

$$t_{tp} = (t_{mp})\ Modulo\ t_{tp}$$

Desweiteren wird zwischen der Periode des zu testenden Takts als Testperiode und der Periode des Controller-Takts als Tastperiode unterschieden. Bild 3.12 zeigt ein Beispiel für eine Unterabtastung. Der zu testende Takt (3 MHz) wird mithilfe eines Controllertakts (8 MHz) abgetastet. Dabei werden mehrere Messpunkte aufgezeichnet. Die hier gewählten Periodenzeiten (ptakt = $\frac{1}{3000000}$, ptast = $\frac{1}{8000000}$) besitzen ein kleinstes gemeinsames Vielfaches bei 1 μs, d.h. die Sequenz der abgetasteten Testpunkte wiederholt sich nach genau dieser Zeitspanne. Die Anzahl der möglichen, unterschiedlichen Testpunkte ergibt sich — je nachdem wie teilerfremd Takt- und Tastfrequenz sind — zu:

$$n_{tp} = \frac{kgV\,(f_{takt},\ f_{tast})}{f_{takt}}$$

Die Bestimmung des kleinsten gemeinsamen Vielfachen (kgV) ist hierbei auf ganzzahlige Parameter angewiesen. Durch Einführung eines Skalierungswerts (scale) kann die Anzahl der Testpunkte ebenfalls aus den Periodendauern der Signale berechnet werden, wobei $scale = f_{tast} \cdot f_{takt}$ immer zu ganzzahligen kgV-Parametern führt:

$$n_{tp} = \frac{kgV\,(p_{takt} \cdot scale, p_{tast} \cdot scale))}{p_{tast} \cdot scale}$$

In dem in Bild 3.12 dargestellten Beispiel ergibt sich $n_{tp} = \frac{kgV(f_{takt}, f_{tast})}{f_{takt}} = \frac{kgV(3\ MHz,\ 8\ MHz)}{3MHz} = 8$.

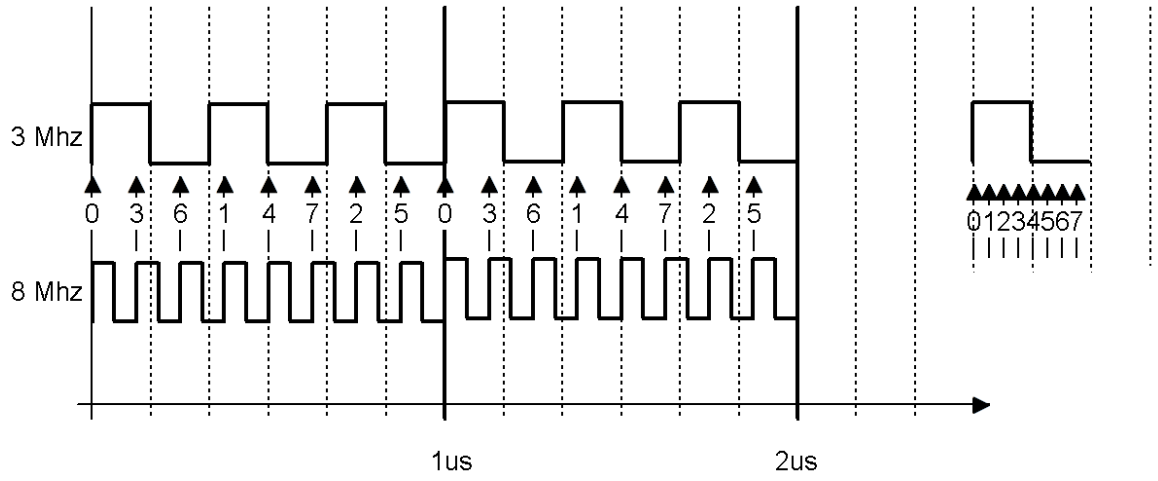


Abbildung 3.12: Taktsignal-Abgleich

3.6.2 Anwendung

Während einer Gesamtmessung werden mehrere Messpunkte (t_{mess}, z_{mess}) erfasst. Diesen wird ein jeweiliger Testpunkt zugeordnet:

$$index_{tp} = \frac{(t_{mess}\ Modulo\ t_{Taktperiode}) \cdot n_{tp}}{t_{Taktperiode}}$$

An jedem Testpunkt wird gezählt, wie häufig die Signalzustände {0, 1} aufgetreten sind. Hierzu dient ein Array von Zählern für jeden Testpunkt und beide Signalzustände. Bei einem vorliegenden Messpunkt geschieht dies durch Inkrementieren des Zählers $d[index_{tp}][z]$.

Zwischen den Einzelmessungen ist eine zufällige Verzögerung einprogrammiert, um eine Gleichverteilung der Messpunkte auf den Testpunkten zu erreichen. Dabei wird zunächst eine Zufallszahl i ($0 <= i <= i_{max}$) generiert. Eine For-Schleife führt nun i-mal eine NOP-Instruktion (No Operation) aus, welche genau einen Controller-Takt benötigt. i_{max} wird vor der Messung festgelegt und optimalerweise so gewählt, dass die zeitliche Verzögerung bei $i = i_{max}$ das kleinste gemeinsame Vielfache der Periodendauern ergibt.

Am Ende der Messung können die unterschiedlichen Zähler für die Zustände {0, 1} auch zur Auftrittswahrscheinlichkeit des Signalzustands 1 zusammengefasst werden:

$$p(z = 1)_{tp} = \frac{d[index_{tp}][1]}{d[index_{tp}][0] + d[index_{tp}][1]}$$

Bild 3.13 zeigt das Ergebnis einer Gesamtmessung am hier betrachteten System. Die vorliegenden Taktfrequenzen (Controller: 80 MHz, ASIC: 25 MHz) führen zu 16 verschiedenen Testpunkten. Man erkennt deutlich die steigende und fallende Flanke innerhalb der Testperiode. Dieser beobachtbare Gesamteffekt ergibt sich durch alle drei Eigenschaften des Taktsignals (Frequenzabweichung, Jitter, Flankensteilheit). Eine weitere Differenzierung ist aufgrund der erhobenen Testdaten nicht möglich.

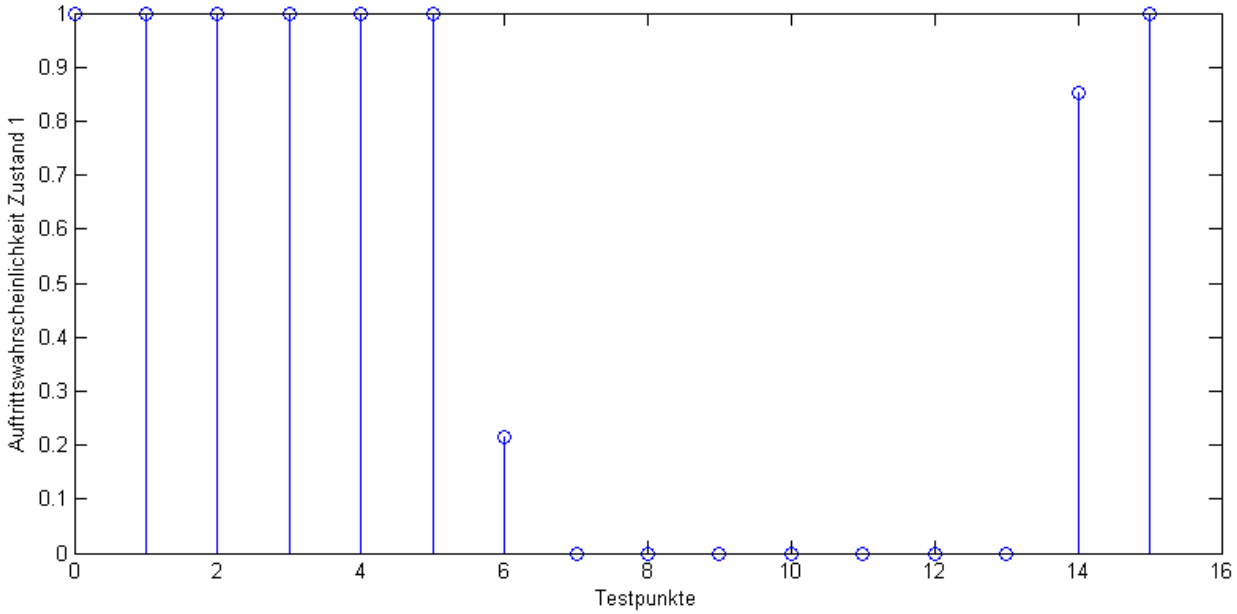


Abbildung 3.13: Taktsignal-Abgleich

3.7 Befundungstest

Der hier vorgestellte Befundungstest bezieht sich auf einen einzigen ASIC. Sofern in Zukunft weitere ASICs die SPI-Befehle zum Umschalten in den JTAG@SPI-Testmodus enthalten, kann der Test problemlos um die jeweils unterstützten BIST-Methoden erweitert werden. Falls diese BIST-Methoden vom 1149.1-Standard her verbotene Taktsynchronizationen zwischen Test- und Systemtakt benötigen, muss dies eventuell in der Firmware für den Hauptcontroller berücksichtigt bzw. nachgebessert werden. Ein JTAG-basierter Boundary Scan auf Steuergerät-Ebene ist - falls mehrere ASICs externe Boundary-Scan-Ketten aufweisen - ebenfalls denkbar.

3.7.1 Funktionaler Vortest

Der funktionale Vortest erleichtert den Testaufbau zum Befundungstest. Hier wird schrittweise die Verbindung zwischen Laptop, Hauptcontroller und ASIC geprüft. Es folgt ein einfacher Test zur Erkennung von Fehlern in den JTAG-Teststrukturen, speziell den JTAG-Registern. Folgende Testmethoden sind verfügbar:

CRC-Prüfung der Firmware : Die entwickelte Hauptcontroller-Firmware wird vom Compiler mit ihrer Speicheradresse und Größe ausgestattet. Eine einprogrammierte Checksummenfunktion berechnet die vom Ethernet-Protokoll her bekannte CRC32M-Prüfsumme. Diese kann über eine Firmwarefunktion abgerufen werden. So erkennt der Tester, ob die richtige Firmwareversion im Einsatz ist. RAM-Defekte im Hauptcontroller, welche das Testprogramm korrumpieren, werden ebenfalls erkannt.

Controller Ping : Diese Firmwarefunktion gibt alle als Parameter im Aufruf angehängten Bytes zurück. Kommunikationsprobleme zwischen Laptop und Hauptcontroller sind somit leicht ersichtlich. Der Test sendet pseudo-zufällige Datenblöcke an den Controller, welcher diese beantwortet. Bei falschem Rückgabewert wird der Test mit dem Ergebnis einer fehlerhaften Kommunikationsschnittstelle abgebrochen.

TAP Ping : Die variablen Parameter dieser Funktion werden durch das festgelegte JTAG-Register geshiftet und anschließend vom Hauptcontroller zum Laptop zurückgeschickt. Dies ist sowohl mit dem Instruktionsregister als auch mit allen Datenregistern möglich. Der vorherige Inhalt der Register wird im Anschluss wieder hergestellt. Dieser Test validiert die komplette Verbindung vom Laptop bis hin zu den Teststrukturen (JTAG-Registern) des entsprechenden ASICs. Dabei können die JTAG-Register mit bekannten Mustern {0000...,0101..., 1010...,1111...} auch strukturell geprüft werden.

3.7.2 Testplan

Bild 3.14 zeigt das Blockschaltbild des hier zu befundenden ASICs. Er besteht aus integriertem Speicher (ROM & RAM), einem Digital- und Analogteil. Es folgt eine Liste von Testmethoden sowie der gestestete Komponenten :

Funktionaler Vortest : Hierdurch werden die Teststrukturen, insbesondere der Testaufbau sowie die JTAG-Register (unter anderem iBS, internal Boundary-Scan-Chains) geprüft.

Memory BIST : Der standardmäßig auch bei jedem Start des ASICs ausgeführte Memory-BIST prüft 3 interne RAM- und 2 interne EEPROM-Blöcke durch einen March-C-Test. Er deckt somit den kompletten Speicherbereich im Bezug auf Stuck-At Faults, Transition Faults, Adress-Decoder Faults und Coupling Faults ab. Die Komplexität des Testalgorithmus liegt im Bezug auf die Größe des Speichers in $O(10n)$ [MLBc].

Softwarebasierter Scan-Test : Im Vergleich zu einen Logik-BIST kann dieser Test nur Stuck-At-Faults erkennen. Dafür bezieht er sich jedoch auf den gesamten Digitalteil des ASICs. Je nach Größe des Digitalteils und Umsetzung von Design-For-Test-Maßnahmen kann in annehmbarer Testzeit eine maximale Abdeckung aller Stuck-At-Faults erreicht werden. Die jeweiligen Scandaten können mithilfe eines ATPG-Tools automatisch generiert werden.

ADC Testsignal : Interne Testmethoden, welche über das JTAG@SPI-Protokoll ansprechbar sind, können das Eingangssignal des ADC Dezimators mit einem Testsignal belegen. Dieses kann im Anschluss vom Delta Sigma-Wandler (Clock Fab DS) zurückgelesen werden. Auf diese Art ist der Digitalteil des Delta-Sigma-Wandlers testbar.

PLL-Taktabgleich : Zur Ansteuerung einer Zusatzplatine mit RC-Glied und Impedanzwandler werden Steuergerät-interne Signale herausgeführt. Dies ist nach einer späteren Integration in den Hauptcontroller nicht mehr nötig.

Bei mehreren ASICs mit dem hier untersuchten JTAG@SPI-Testmodus kann zudem ein Boundary Scan Test durchgeführt werden, um die Zwischenverbindungen (Interconnects) der ASICs strukturell zu testen. Dies benötigt jedoch eine externe Boundary-Scan-Kette im ASIC-Design. Dies erlaubt, den Befundungstest auf Board-Ebene auszuweiten.

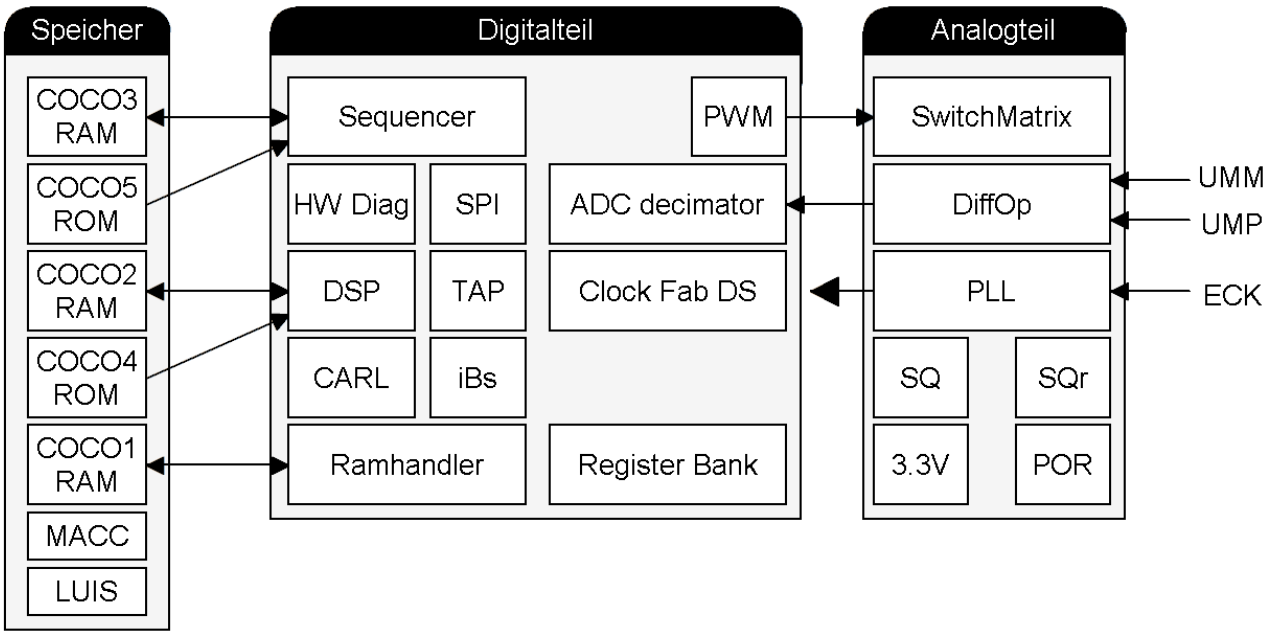


Abbildung 3.14: Blockschaltbild des hier untersuchten ASICs

4 Ergebnisse

4.1 Analyse des JTAG@SPI-Protokolls

Die erarbeitete JTAG@SPI-Protokollansteuerung erlaubt die korrekte Durchführung von JTAG-Kommunikation mit einem ASIC. Das Steuergerät muss hierzu nicht geöffnet werden, was den Einsatz der ASIC-internen, strukturellen Testmethoden in allen Testbereichen ermöglicht. Die erreichte Übertragungsgeschwindigkeit ist wegen der softwarebasierten Ansteuerung nicht mit der Testgeschwindigkeit eines ATE vergleichbar. Dieser arbeitet beim Produkttest des vorliegenden ASICs mit bis zu 25MHz. Neben der Realisierung in Software (vgl. SPI-Blocker) existiert in dieser Implementierung ein weiteres Bottleneck: das eingesetzte Übertragungsinterface zwischen Laptop und Hauptcontroller, sowie die Größe des Zwischenspeichers im Hauptcontroller (28Kb). Bild 4.1 zeigt die vorgenommenen Zeitmessungen bei der Ausführung von JTAG-Instruktionen mit unterschiedlicher Shiftlänge des Datenregisters. Die genutzten Shiftdaten sind zufällig gewählt. Die Delay-Einstellung des SPI-Blockers wurde zur Darstellung der maximalen Geschwindigkeit auf ons gesetzt. Die blauen Markierungen stellen jeweils den Zeitbedarf zur Übertragung der Instruktion dar (IR Shift, 5 Takte), während die grünen Markierungen dem Shift des Datenregisters (DR Shift) mit variierender Shiftlänge entsprechen. Beim IR Shift wird das 5-Bit-breite Instruktionsregister befüllt; hierzu sind nach dem JTAG-Protokoll mindestens 12 Testtakte bei der TAP-Ansteuerung nötig. Die Anzahl der Testtakte beim DR SHIFT berechnet sich zu $Shiftlnge + 6$. Hieraus ergibt sich bei Betrachtung der Shiftlänge 32 folgende, durchschnittliche Übertragungsgeschwindigkeit:

$$r_{JTAG@SPI} = \frac{(5Bit\ IRSHIFT + 32Bit\ DRSHIFT)}{Gesamtzeit} = \frac{12\ Bits + (32+6)\ Bits}{0.000446584375s} = 111.960,93\frac{Bit}{Sekunde}$$

Die kleinen zeitlichen Abweichungen der Messergebnisse sind durch Ausführungen mit und ohne JTAG-Speedup begründet. Die durch den Speedup erzielte Zeiteinsparung hängt von der SPI-Adresse des ASICs, den (hier zufällig gewählten) Shiftdaten und der jeweiligen JTAG-Operation ab. Zur theoretischen Auswertung des JTAG-Speedups wird der Zustandsautomat des SPI Blockers nun als stochastischer Prozess, speziell als diskrete, homogene Markov-Kette einfacher Ordnung aufgefasst. Bild **??** führt die Übergangswahrscheinlichkeiten im Normalmodus und im JTAG-Modus auf, wie sie bei langen Shiftketten oder Pausezuständen auftreten, bei denen das TMS-Signal für viele Takte den Wert o innehält, der Chipselect des seriellen Busses also aktiv ist.

Ohne die Optimierung liegt am Anfang eines SPI-Datenframes zu 50% das richtige erste Adressbit an. Ein Übergang in die Zustände ao und $\overline{a_0}$ ist somit gleichwahrscheinlich. Im Zustand $\overline{a_0}$ wird nach

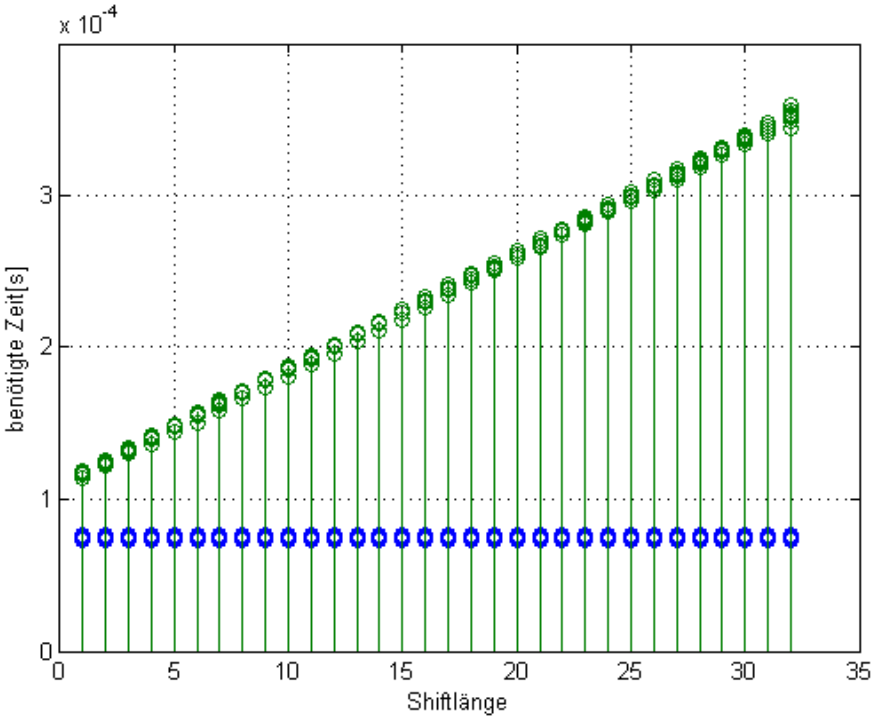


Abbildung 4.1: JTAG@SPI-Timing: Übertragungsdauer bei unterschiedlichen Shiftlängen

dem ersten Takt ein asynchroner CS-Reset durchgeführt, dies bricht den SPI-Datenframe ab. Das folgende, erste Adressbit ist wieder mit gleicher Wahrscheinlichkeit richtig oder falsch. Im Zustand a_0 wird das zweite Adressbit mit einer Wahrscheinlichkeit von ebenfalls 50% richtig gesetzt, was zu einem Übergang in den Zustand IN-FRAME führt. Bei einem CS-Reset aufgrund eines falschen zweiten Adressbits ist das im nächsten Takt gesetzte erste Adressbit wieder in der Hälfte der Fälle richtig und führt so zu a_0 oder $\overline{a_0}$.

Im JTAG-Modus verändern sich die Übergangswahrscheinlichkeiten aufgrund der Struktur des TAP-Zustandautomats (vgl Bild 2.8 in Kapitel 2.5.1). Der NO-FRAME-Zustand wird bei TMS/CS=0 verlassen. In 14 der 16 0-Transitionen im TAP-Automaten - nämlich in allen außer SHIFT_DR -> SHIFT_DR und SHIFT_IR->SHIFT_IR - kann das richtige Adressbit injiziert werden. Bei den 2 kritischen Übergängen ergibt sich in der Hälfte der Fälle zufällig ein richtiges oder falsches Adressbit. Die Übergangswahrscheinlichkeiten von NO-FRAME nach a_0 und $\overline{a_0}$ ergeben sich so zu $\frac{15}{16}$ und $\frac{1}{16}$. Eine gleiche Übergangsverteilung liegt nach dem CS-Reset im Zustand $\overline{a_0}$ vor. Im Zustand a_0 wird das erste Adressbit richtig gesetzt, es ist also bereits eine 0-Transition im TAP passiert. Der nächste Übergang muss ebenfalls TMS=0 aufweisen, da ein vorzeitiger, erzwungener Frame-Abbruch hier nicht betrachtet wird. Diese TMS-Sequenz $\{0,0\}$ kann im TAP an 11 Stellen geschehen. An 4 dieser Stellen ist der zweite TAP-Zustand SHIFT_DR oder SHIFT_IR, d.h. in 7 von 11 Fällen ist das

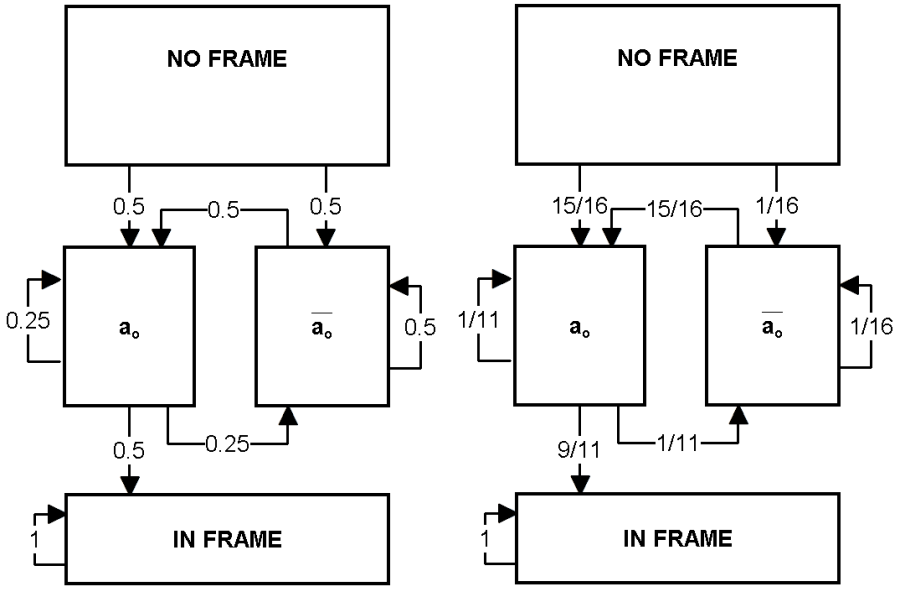


Abbildung 4.2: SPI Blocker, aufgefasst als einfache Markov-Kette

Injizieren des Adressbits zulässig. In den übrigen 4 Fällen kann das Adressbit nicht injiziert werden, es ist somit per Zufal in 2 Fällen richtig und in 2 Fällen falsch.

Mit den nun festgelegten Übergangswahrscheinlichkeiten kann die Wahrscheinlichkeit berechnet werden, mit der sich der SPI Blocker nach n Takten mit TMS=0 in einem bestimmten Zustand befindet. Dabei werden die Wahrscheinlichkeiten, dass sich der SPI Blocker in einem Zustand befindet in einem Vektor z = {p(IN-FRAME), p(a_0), p($\overline{a_0}$), p(IN-FRAME) } zusammengefasst.

$$z_n = A^n * z_0$$

mit

$$A_{normal} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{pmatrix} \quad und \quad A_{Speedup} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{15}{16} & \frac{1}{11} & \frac{15}{16} & 0 \\ \frac{1}{16} & \frac{1}{11} & \frac{1}{16} & 0 \\ 0 & \frac{9}{11} & 0 & 1 \end{pmatrix} \quad und \quad z_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Bild 4.3 zeigt die Wahrscheinlichkeit, dass sich der SPI Blocker nach n Schritten im IN-FRAME-Zustand befindet. Es ist zu sehen, dass der In-Frame Zustand im JTAG-Modus nach 8 Schritten in 90% der Fälle erreicht ist. Im Gegensatz dazu wird der In-Frame-Zustand im Normalmodus erst nach 11 Schritten zu 90% erreicht.

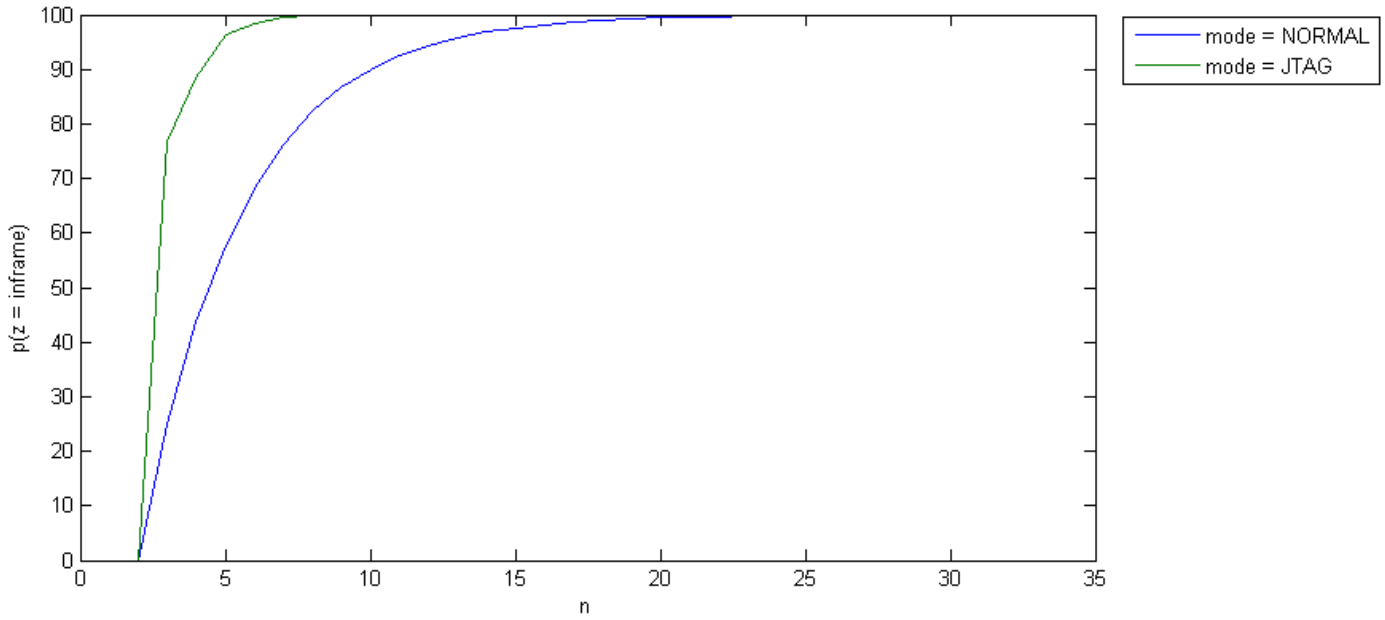


Abbildung 4.3: JTAG@SPI-Timing: Übertragungsdauer bei unterschiedlichen Shiftlängen

4.2 Übertragungsverhalten des Taktsignal-BIST

Der vorgeschlagene Taktsignal-BIST bestehend aus Tiefpass und Impedanzwandler wurde auf einer Testplatine nachgebildet. Es gilt R= 100Ohm und C=100 nF, die Bauteilvarianz liegt bei 1%. Bild 4.4 zeigt gemessene Analogspannungen in Abhängigkeit der zuvor eingestellten PWM-Frequenz. Man erkennt deutlich das gewünschte Übertragungsverhalten des Tiefpasses.

4.3 Messdaten des softwarebasierten Taktabgleichs

Bild 4.5 visualisiert 768 Gesamtmessungen, die über einen Zeitraum von 10 Sekunden durchgeführt wurden. Jede Gesamtmessung dauerte 4,4 ms und besteht aus 4096 Messpunkten, die zu einer Testperiode zusammengesetzt eine Linie des Diagramms ausmachen. Zwischen den Gesamtmessungen benötigt die Übertragung der erhobenen Testdaten vom Hauptcontroller zum Laptop 9,2 ms. Dabei werden lediglich der aufbereiteten Werte des Zählerarrays, nicht etwa die Rohdaten transferiert.

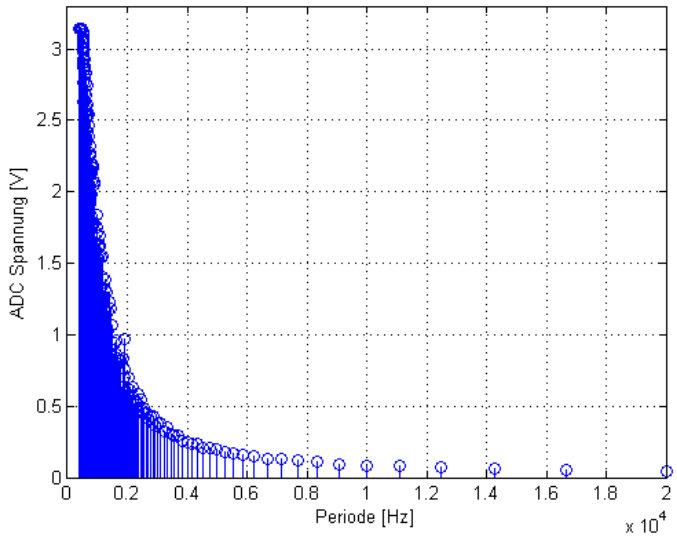


Abbildung 4.4: Übertragungsverhalten Tiefpass

4.4 Analyse des Befundungstests

Bild 4.6 stellt das Blockschaltbild des hier befundeten ASICs samt erreichter Testabdeckung und benötigter Testzeit dar. Mithilfe des Memory BIST kann der Speicherbereich im Bezug auf Stuck-At Faults, Transition Faults, Adress-Decoder Faults und Coupling Faults in 15 ms komplett geprüft werden. Im Digitalteil erreicht der softwarebasierte Scan-Test in 132 Sekunden eine Abdeckung von 97% aller Stuck-At Faults. Weitere Fehlerklassen werden hier — aufgrund der beschriebenen Problematiken — nicht abgedeckt. Weitere Komponenten des Digitalteils können darüberhinaus funktional getestet werden: Durch die PingTAP-Funktion des Vortests wird der TAP samt JTAG-Registern untersucht. Das Testsignal für den ADC erlaubt zusätzlich die funktionale Prüfung des Delta-Sigma-Wandlers. Im Analogteil des ASICs ist zunächst nur der PLL-Schwingkreis mithilfe des softwarebasierten Taktsignal-Abgleichs testbar. Bei 16 Testpunkten entspricht dies einer Genauigkeit von $\frac{15}{16}$ (93,75%). Der Zeitaufwand ist abhängig von der Anzahl der erhobenen Messpunkte, bei 128 Messpunkten pro Testpunkt ergeben sich im Gesamten $128 \cdot 16 = 2048$ Messpunkte. Dies benötigt ca. 11,2 ms. Die Schaltmatrix (SwitchMatrix) kann per JTAG@SPI angesteuert und mit einem Multimeter nachgemessen werden. Die Stromquellen SQ, SQr sowie der Spannungsregler (VReg) können ebenfalls über die Schaltmatrix auf einen Pin herausgeschaltet werden, so dass eine funktionale Prüfung mithilfe von Spannungs- und Stromessungen möglich ist.

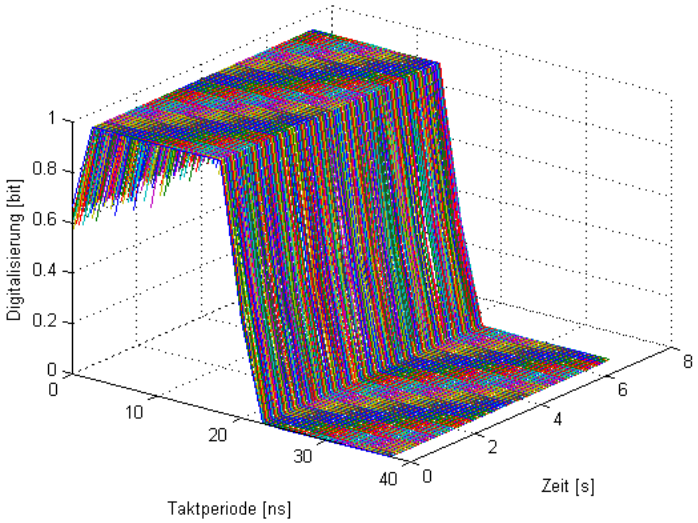


Abbildung 4.5: Messdaten des softwarebasierten Taktabgleichs

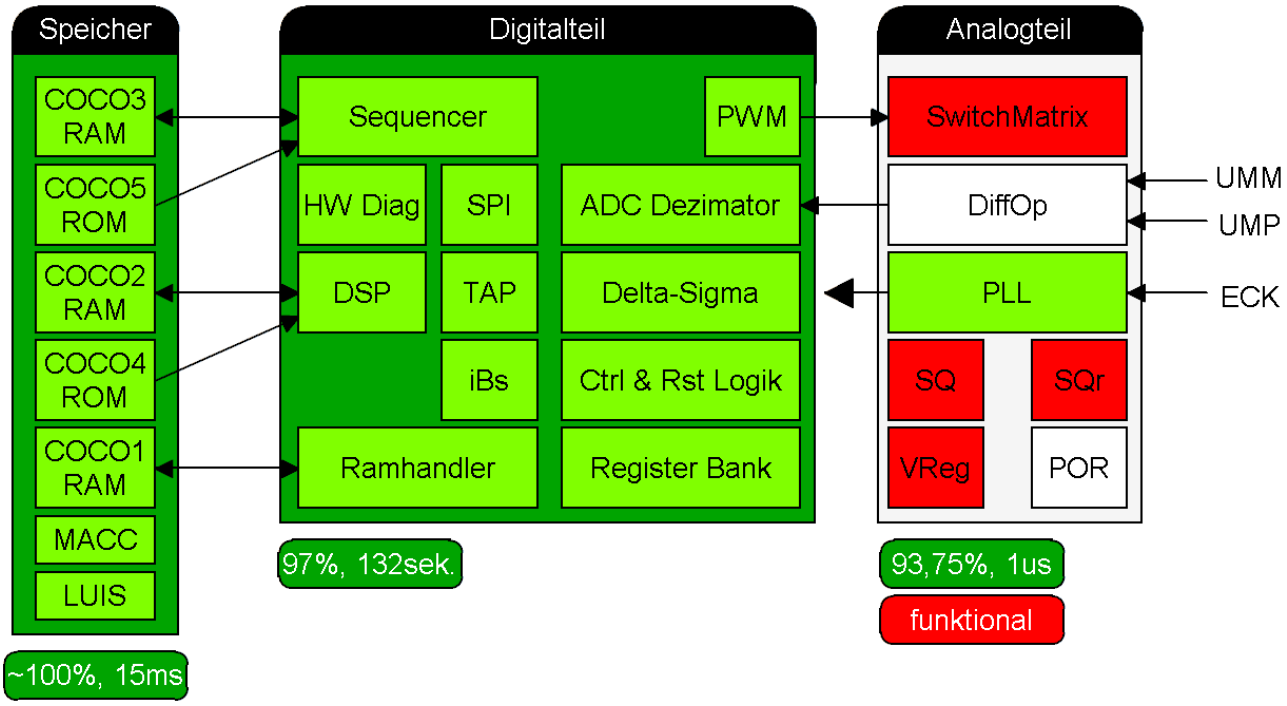


Abbildung 4.6: Testabdeckung & Testzeiten des Befundungstests

5 Fazit

Der SPI Blocker erlaubt — trotz Adress-Multiplexing — jegliche Testkommunikation mit einzelnen ASICs. Dies ist nicht zwingend an das JTAG-Protokoll gebunden. Per JTAG@SPI können im geschlossenen Steuergerät ASIC-interne, strukturelle Testmethoden ausgeführt werden. Von der Übertragungsgeschwindigkeit her ist diese Implementierung nicht mit den Testtakt eines ATE vergleichbar. Dies stellt jedoch keine Einschränkung dar, da das JTAG-Interface nicht zur Übertragung großer Datenmengen eingesetzt wird.

Der softwarebasierte Scan-Test ist als Ersatz für einen Logik-BIST nur eingeschränkt nutzbar. Durch mehrere Faktoren ist diese Implementierung bei weitem nicht schnell genug, die Fehlerabdeckung des Tests bezieht sich daher nur auf das statische Stuck-At-Modell. Die Erkennung von Transition Delay Faults ist ausgeschlossen. Eine Erweiterung um diese Fehlerklasse könnte durch folgende Maßnahmen erreicht werden:

- schnelles Interface zur Übertragung der Testvektoren zwischen Laptop und Hauptcontroller

- Hardwareunterstützung zum Setzen von Testeingängen und Prüfen von Testausgängen

- Ersatz für den softwarebasierten SPI Blocker

Die BIST-Methode für Taktsignale ermöglicht erstmalig eine Prüfung des Hauptcontroller-Takts auf Frequenzabweichungen. Durch den softwarebasierten Abgleich ASIC-interner PLL-Schwingkreise können die Systemtakte einzelner ASICs auf Frequenzabweichung, Jitter und Flankensteilheit hin überprüft werden. Auch wenn der Test keine Einzelbetrachtung dieser Eigenschaften erlaubt, so können sie im Gesamten mit der Auflösung eines Testpunkts beschränkt werden.

Eine komplette Konformität zum 1149.1-Standard ist auch bei zukünftigen ASICs ausgeschlossen, da der Einsatz dedizierter Test-Pins für ein Testinterface zu kostspielig ist. Die bereits umgesetzte, konstengünstige Lösung — die JTAG Signale durch SPI-Befehle auf dem seriellen Interface erreichbar zu machen — scheint hier als milde Nicht-Konformität zum Standard ein annehmbarer Kompromiss zu sein. Dennoch sollten weitere Abweichungen vermieden werden, da das JTAG-Testinterface bereits im Datenblatt des hier untersuchten ASICs beschrieben wird und eventuell in Zukunft als Produktfeature aufgeführt werden soll. Insbesondere der bisher rein funktionale Produkttest von Steuergeräten kann durch dieses Feature erheblich erleichtert und im Hinblick auf Testzeit und Testkosten optimiert werden.

6 Ausblick

Durch den Einsatz des erweiterten IEEE 1149.7-Standards kann zunächst der Verdrahtungsaufwand im Steuergerät (vgl. Star-2-Topologie) minimiert werden. In diesem Zusammenhang ist eine JTAG@SPI-Lösung ohne Ansteuerung des Chipselect-Signals denkbar. Der neue Standard würde so eine adressierte Testkommunikation mit ASICs erlauben, während das Chipselect-Signal über den gesamten Testzeitraum deaktiviert bleibt. Dies führt nicht nur aufgrund des auf Durchsatz optimierten 1149.7-Standards sondern auch durch die Einsparung des softwarebasierten SPI Blockers zu einer erheblichen Durchsatzsteigerung bei der Testkommunikation. Damit ist auch die erste Maßnahme zur Erweiterung des softwarebasierten Scan-Tests gegeben. Zum hardwareunterstützten Setzen der Testeingänge könnte in Zukunft DMA (Direct Memory Access) eingesetzt werden. Sofern der SPI-Bus in Zukunft durch den BOSCH-internen MSC-Bus ersetzt wird ist auch eine schnellere Kommunikation zwischen Laptop und ASIC gegeben. Dies ermöglicht eventuell, den softwarebasierten Scan-Test für Transition Delay Faults zu erweitern.

Sofern zukünftige ASICs mit einer externen Boundary-Scan-Kette ausgestattet sind, kann der bisher rein funktionale Produktionstest von Steuergeräten durch einen Boundary Scan im Hinblick auf Testabdeckung und Testzeit erheblich optimiert werden.

Ein hierarchischer Testplan basierend auf BIST-Methoden und strukturellen, softwarebasierten Tests kann in Zukunft allen Testbereichen ohne die Notwendigkeit externer Tester genügen. Hierbei sind die Concurrent-Online-BIST-Methoden interessant, da so nicht über eine eventuelle, gefährdende Aktivierung von Test-Modi im Fahrbetrieb nachgedacht werden muss.

Literaturverzeichnis

[CH96] C. Halper, G. B. M. Heiss H. M. Heiss: Digital-to-analog conversion by pulse-count modulation methods. In: *IEEE Transactions on Instrumentation and Measurement* (1996), S. 805 –814. http://dx.doi.org/10.1109/19.517000. – DOI 10.1109/19.517000. – ISSN 0018–9456

[Cha06] Cha, J. H.: An extended model for optimal burn-in procedures. In: *IEEE Transactions on Reliability* (2006), S. 189 – 198. http://dx.doi.org/10.1109/TR.2006.874921. – DOI 10.1109/TR.2006.874921. – ISSN 0018–9529

[CHPW05] C. H.-P. Wen, K.-T. Cheng-K. Yang W.-T. Liu J.-J. C. L.-C. Wang W. L.-C. Wang: On a software-based self-test methodology and its application. In: *23rd IEEE VLSI Test Symposium, 2005 Proceedings*, 2005. – ISSN 1093–0167, S. 107 – 113

[DK98] D.B. Kececioglu, Wendai W.: Parameter estimation for mixed-Weibull distribution. In: *Annual Reliability and Maintainability Symposium, 1998 Proceedings*, 1998, S. 247 –252

[Hil07] Hiller, H.: "There Is More Than Moore In Automotive ....". In: *44th ACM/IEEE Design Automation Conference, DAC 2007*, 2007. – ISSN 0738–100X, S. 376

[HJW96] H.-J. Wunderlich, G. K.: Bit-flipping BIST. In: *IEEE/ACM International Conference on Computer-Aided Design, ICCAD 1996*, 1996, S. 337 –343

[Hol00] Holland, R.: The introduction of networking into the automotive industry. In: *IEEE Seminar on New Product Introduction in Electronics (Ref. No. 2000/047)*, 2000, S. 3/1 –316

[IEE10] IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture. In: *IEEE Std 1149.7-2009* (2010), S. 1–985. http://dx.doi.org/10.1109/IEEESTD.2010.5412866. – DOI 10.1109/IEEESTD.2010.5412866

[IP08] I. Pomeranz, S. M. R.: Unspecified Transition Faults: A Transition Fault Model for At-Speed Fault Simulation and Test Generation. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2008), S. 137 –146. http://dx.doi.org/10.1109/TCAD.2007.907000. – DOI 10.1109/TCAD.2007.907000. – ISSN 0278–0070

[JFL01] J.-F. Li, C.-T. Huang C.-W. W. K.-L. Cheng C. K.-L. Cheng: March-based RAM diagnosis algorithms for stuck-at and coupling faults. In: *International Test Conference, 2001 Proceedings*, 2001, S. 758 –767

[JLB07] J. Lagos-Benites, P. Bernardi-M. Grosso D. Ravotto E. Sanchez-M. S. R. D. Appello A. D. Appello: An Effective Approach for the Diagnosis of Transition-Delay Faults in SoCs, based on SBST and Scan Chains. In: *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, DFT 2007*, 2007. – ISSN 1550–5774, S. 291 –302

[JR00] J. Rajski, J. T. N. Tamarapalli T. N. Tamarapalli: Automated synthesis of phase shifters for built-in self-test applications. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2000), S. 1175 –1188. http://dx.doi.org/10.1109/43.875312. – DOI 10.1109/43.875312. – ISSN 0278–0070

[JTA90] IEEE Standard Test Access Port and Boundary - Scan Architecture. In: *IEEE Std 1149.1-1990* (1990). http://dx.doi.org/10.1109/IEEESTD.1990.114395. – DOI 10.1109/IEEESTD.1990.114395

[Lee09] Leens, F.: An introduction to I2C and SPI protocols. In: *IEEE Instrumentation Measurement Magazine* (2009), S. 8 –13. http://dx.doi.org/10.1109/MIM.2009.4762946. – DOI 10.1109/MIM.2009.4762946. – ISSN 1094–6969

[Ley09] Ley, A. W.: Doing more with less - An IEEE 1149.7 embedded tutorial : Standard for reduced-pin and enhanced-functionality test access port and boundary-scan architecture. In: *International Test Conference, ITC 2009*, 2009, S. 1 –10

[MJH81] M. J. Howes, D. V. M.: *Reliability and Degradation - Semiconductor Devices and Circuits*. 1981

[MLBa] M. L. Bushnell, V. D. A.: *Essentials of Electronic Testing for Digital Memory & Mixed-Signal VLSI Circuits*. – 57–58 S.

[MLBb] M. L. Bushnell, V. D. A.: *Essentials of Electronic Testing for Digital Memory & Mixed-Signal VLSI Circuits*. – 489–548 S.

[MLBc] M. L. Bushnell, V. D. A.: *Essentials of Electronic Testing for Digital Memory & Mixed-Signal VLSI Circuits*. – 285 S.

[Moh] Mohamed, A. R.: Built-In Self-Test (BIST), S. 6

[NAT01] N. A. Touba, E. J. M.: Bit-fixing in pseudorandom sequences for scan BIST. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2001), S. 545 –555. http://dx.doi.org/10.1109/43.918212. – DOI 10.1109/43.918212. – ISSN 0278–0070

[RD98] R. Dorsch, H.-J. W.: Accumulator based deterministic BIST. In: *International Test Conference, 1998 Proceedings*, 1998. – ISSN 1089–3539, S. 412 –421

[Rob94] Robinson, G. D.: Why 1149.1 (JTAG) really works. In: *Electro/94 International Conference Proceedings, Combined Volumes*, 1994, S. 749 –754

[SC00] S. Chiusano, H.-J. W. P. Prijetto P. P. Prijetto: Non-intrusive BIST for systems-on-a-chip. In: *International Test Conference, 2000 Proceedings*, 2000, S. 644 – 651

[SM09] S. Menon, V. A. A. D. Singh S. A. D. Singh: Output Hazard-Free Transition Delay Fault Test Generation. In: *27th IEEE VLSI Test Symposium, VTS 2009*, 2009. – ISSN 1093–0167, S. 97 –102

[SN06] S. Neophytou, S. T. M. K. Michael M. M. K. Michael: Efficient deterministic test generation for BIST schemes with LFSR reseeding. In: *12th IEEE International On-Line Testing Symposium, IOLTS 2006*, 2006, S. 6 pp.

[Tou07] Touba, N. A.: X-canceling MISR; An X-tolerant methodology for compacting output responses with unknowns using a MISR. In: *2007. IEEE International Test Conference, ITC 2007*, 2007. – ISSN 1089–3539, S. 1 –10

[XS11] X. Shubing, T. R.: A selective X-masking for test responses in the presence of unknown values. In: *2011 8th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2011*, 2011, S. 159 –162

[ZVE10] ZVEI: Mikroelektroniktrendanalyse bis 2013. (2010). – http://www.zvei.de/

Erklärung

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

_____

(Dominik Ull)