Douglas Cunningham, Petra Hofstedt, Klaus Meer, Ingo Schmitt (Hrsg.): INFORMATIK 2015
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015

Test eingebetteter Prozessoren im Zielsystem mit hoher diagnostischer Auflösung

Christian Gleichner, Heinrich T. Vierhaus, Brandenburgische Technische Universität Cottbus-Senftenberg

Abstract: Strukturorientierte Tests können eingesetzt werden, um die Funktionstüchtigkeit hochintegrierter Schaltungen zu überprüfen. Der Produktionstest wird von einer speziellen, schaltungsinternen Testlogik unterstützt, die über dedizierte I/O-Kanäle angesteuert wird. Im Zielsystem ist diese Schnittstelle aber nicht verfügbar. Um einen erweiterten Systemtest mit diagnostischen Fähigkeiten im Feld zu realisieren, bietet es sich an, den Testzugang über vorhandene Standardschnittstellen zu ermöglichen. Die in dieser Arbeit vorgestellte Testschnittstelle erlaubt es, Testroutinen für den im Zielsystem eingebetteten Prozessor durchzuführen, welche über den stets unvollständigen funktionalen Test hinaus auch strukturorientierte Tests mit hoher Fehlerüberdeckung umfassen.

Keywords: Scan-test, Fehlerdiagnose, eingebettetes System, EDT, USB, FlexRay.

1 Einleitung

In vielen technologischen Bereichen nimmt die Komplexität eingebetteter Systeme mit steigenden Ansprüchen bezüglich Qualität und Sicherheit, aber auch aus ökologischen und ökonomischen Aspekten stetig zu. Zum einen wird dies erreicht durch die Anzahl der Prozessoren, Controller und Sensoren und deren Vernetzung, zum anderen durch den rasanten technologischen Fortschritt in der Halbleiterindustrie. Durch die hohe Integrationsdichte und die dadurch erhöhte Sensibilisierung gegenüber potentieller Fehlerquellen während des Produktionsprozesses lassen sich bei der Halbleiterfertigung defekte Chips oder solche mit erhöhter Fehleranfälligkeit nicht vermeiden. Somit werden Ausbeute und Lebensdauer hochintegrierter Schaltungen (IC) reduziert. Da diese vermehrt auch Einzug in sicherheitskritische Anwendungen halten, ist es wichtig, einen hohen Qualitätsstandard und eine hohe Fehler- und Ausfallresistenz zu gewährleisten.

Sei hier das Automobil exemplarisch für ein komplexes elektronisches System betrachtet. Es wäre wünschenswert einen Fehler möglichst frühzeitig zu erkennen, bevor er zu Störung essentieller Funktionen führt. Kommt es zum Teil- oder gar Systemausfall, so ist es darüber hinaus von großer Bedeutung, einen einmal festgestellten Fehler im Nachhinein in der Werkstatt oder als Rückläufer beim Hersteller schnell und eindeutig reproduzieren und diagnostizieren zu können. Die Ursache eines gemeldeten Fehlers in der Fahrzeugelektronik ist aber häufig nicht einwandfrei feststellbar [VS14]. So besteht im Fehlerfall nur die Möglichkeit, Systemkomponenten anhand der Fehlerbeschreibung auf Verdacht auszutauschen. Eine nachträgliche Fehleranalyse durch den Halbleiter-

hersteller erfordert hohen Aufwand, da der IC unter anderem erst von der Platine gelöst werden muss.

Im Produktionstest beim Halbleiterhersteller werden hochauflösende strukturorientierte Verfahren angewandt, um fehlerhafte Chips zu identifizieren und auszusortieren. Hierzu werden in den IC eingebrachte Teststrukturen mit separaten Zugangskanälen genutzt, um eine hohe Fehlerüberdeckung in kurzer Testzeit zu garantieren. Der Testzugang zu dieser Produktionstestlogik steht nach dem Packaging und somit nach dem Aufbringen auf die Platine nicht mehr zu Verfügung.

Das entworfene Konzept realisiert einen für die Diagnose eingebetteter Prozessoren erforderlichen strukturorientierten Test unter Verwendung der schaltungsinternen Produktionstestlogik und serieller Standardschnittstellen. Somit wird ein Test eines ICs mit hoher diagnostischer Auflösung über eine vorhandene Steuergeräteschnittstelle im Zielsystem (Kraftfahrzeug) ohne Demontage des Steuergerätes und der ICs verfügbar gemacht. Dem Steuergerätehersteller kann dadurch ein erweiterter Produktionstest bereitgestellt werden, der weit über den Leiterplattentest hinaus eine nachweisbare hohe Prüfschärfe bietet.

Kann die Störung einer Systemfunktionalität durch die Diagnosemöglichkeit bereits im Feld, d.h. während eines Werkstattaufenthalts, eindeutig auf einen fehlerhaften IC zurückgeführt werden, lassen sich wiederholte Fehlersuchen und teure Reparaturen vermeiden. Der Halbleiterhersteller kann zudem die aus der Diagnose eines defekten ICs gewonnenen Informationen nutzen, um während des Fertigungsprozesses und des Fertigungstests entsprechende Maßnahmen zu ergreifen, mit denen die Chipqualität verbessert werden kann [Ab14].

Um integrierte Schaltungen auch nach dem Fertigungstest noch testen zu können, steht heutzutage bereits die etablierte JTAG-Schnittstelle zur Verfügung [JT01]. Diese ist hauptsächlich für den Boundary-Scan zum Test der Verbindungen zwischen IC und Leiterplatte vorgesehen. Darüber hinaus ist es mittels JTAG möglich, Testvektoren an die primären Eingänge der internen Schaltungslogik anzulegen, um diese erschöpfend, pseudoerschöpfend oder funktional zu testen. Ist der interne Schaltungsaufbau bekannt, kann auch ein strukturorientierter Test, mittels eines SBST-Verfahrens (*Software-based self test*), realisiert werden.

Die neue IJTAG-Schnittstelle soll einen standardisierten und kostengünstigen Zugang zu Testmodulen integrierter Schaltungen auch noch nach dem Produktionstest ermöglichen. Diese internen Testmodule können etwa MBIST-Einheiten zum Test der Speicherbausteine oder LBIST-Einheiten zur Ansteuerung der integrierten Scan-Strukturen sein. Somit macht es IJATG ebenso möglich, die in den IC integrierte Produktionstestlogik für einen diagnostischen Test zu nutzen. Das in dieser Arbeit vorgestellte Konzept stellt eine Alternative zur IJTAG-Lösung dar.

IJTAG nutzt den JTAG-TAP als Zugang zum Gateway-Pfad und zu den daran angeschlossenen Testmodulen [IJ13]. Da der JTAG-TAP und der Scan-Pfad üblicher-

weise mit 5 bis 30 MHz betrieben werden, ist auch die Datenrate für den IJATG-Test entsprechend beschränkt. Da die Testeingaben direkt mit jedem Takt seriell hineingeschoben werden, bedeutet eine JTAG-Clock von 30 MHz eine Datenrate von 30 MBit/s. Der Testzugang über standardisierte Anwendungsschnittstellen kann eine kürzere Testzeit ermöglichen, wenn ein Bussystem mit hoher Datenrate die Grundlage bildet.

Neben dem Testzugang umfasst das entworfene Konzept auch einen integrierten Selbsttest, der strukturelle Fehler des ICs bereits vor einer möglichen Auswirkung auf die Funktionalität des Systems identifizieren kann und somit die Systemzuverlässigkeit erhöht. Um den universellen Einsatz in verschiedenen Anwendungsszenarien zu unterstreichen, wird das entworfene Design im Folgenden als *Universal Scan-Test Interface* (USIF) bezeichnet

2 Universal Scan-Test Interface

Die USIF-Architektur realisiert den Testzugang zum eingebetteten System über HighSpeed-Standardschnittstellen. Über die als Testzugang spezifizierte Schnittstelle werden Testdaten übertragen, in ein definiertes einheitliches Format, welches der zugrundeliegenden Testtechnologie angepasst ist, zwischengespeichert und an den integrierten Test-Controller weitergeleitet. Diese Testdaten können komplette Testmuster für einen direkten Scan-Test, alternativ aber auch hochgradig komprimierte Konfigurationsdaten für einen Selbsttest sein.

Die Testkonfigurationen, wie z.B. der Testtyp, die Testanalyse oder das Ausgabeformat, werden über die im Testsatz enthaltenen Metadaten bestimmt. Testdaten werden entsprechend dem spezifizierten Verfahren in die ICs geladen, um diese strukturell testen zu können. Mittels zusätzlich übertragener Referenzdaten kann eine schaltungsinterne Analyse der Testergebnisse erfolgen. Zudem können die Testantworten auch direkt oder kompaktiert zu Signaturen und ggf. zusätzlich gefiltert über die Standardschnittstelle für anschließende diagnostische Auswertungen ausgelesen werden. Das Konzept wurde bereits in [Gl12, DI13] vorgestellt.

2.1 Aufbau der Testschnittstelle

Die Anbindung an die Scan-Testarchitektur geschieht hauptsächlich über eine Steuerungseinheit und einen oder mehrere Zwischenspeicher für die zu übermittelnden Testdaten. In Abb. 1 ist der konzeptionelle Aufbau der Testschnittstelle dargestellt. Als Zwischenspeicher werden hier separate FIFO-Module verwendet. Der Speicher für die Testeingaben ist das *TestIn*-FIFO. Für die Referenzdaten, die zur Auswertung der Testergebnisse benötigt werden, ist hier das *TestRef*-FIFO vorgesehen. In dem Ausgangsspeicher *TestOut*-FIFO werden die Testantworten abgelegt. Die Testantworten des Scan-Tests können entweder direkt der Scan-Struktur entnommen oder kompaktiert

zu Signaturen von der Testarchitektur abgegriffen werden. Die Analysekomponente setzt die optionale Auswertung on-chip um und kann, im Falle einer anschließenden diagnostischen Auswertung, für die Filterung einer umfangreichen Ausgabe auf fehlerhafte Testantworten sorgen. Um den Testzugang zu steuern, muss eine zusätzliche Controller-Einheit implementiert werden. Diese wertet die empfangenen Metadaten aus, koordiniert somit Schreib- und Lesezugriffe und konfiguriert den Scan-Test.

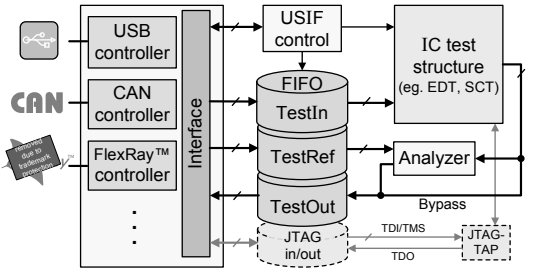

Abb. 1: Aufbau der Testschnittstelle[1]

In der Abb. 1 ist zudem ein optionales JTAG-FIFO zu sehen, das die über eine Standardschnittstelle empfangenen JTAG-Eingangssignale TDI und TMS paketweise zwischenspeichert und für den JTAG-TAP bereitstellen kann. Die Ausgangsdaten aus dem TDO-Signal werden wieder zwischengespeichert, um diese paketweise auslesen zu können. Die Test-Clock TCK und das Reset-Signal TRST werden dabei on-chip erzeugt. Es kann somit auch ein alternativer JTAG-Zugang über eine Standardschnittstelle realisiert werden.

2.2 Testdatenspeicher

Die über die Kommunikationsschnittstelle empfangenen Testeingabedaten werden im *TestIn*-FIFO zwischengespeichert. Das FIFO befindet sich in der USIF-Architektur zwischen dem Empfangspuffer des jeweiligen als Testschnittstelle genutzten Kommunikations-Controllers und der integrierten Testarchitektur (SCT oder EDT). Wie in Abb. 2 dargestellt, wird es mit den aus dem Empfangspuffer entnommenen Daten über den Schreibport der Datenbreite d_{USIF} beschrieben. Diese Datenbreite wird durch die Auslegung der USIF-Architektur bestimmt und wurde für die Umsetzung auf 32 Bit festgelegt. Die Administration der Datenpuffer bei einem Zugriff über die Testschnittstelle wird durch den USIF-Controller übernommen, daher auch durch diesen mit dem Takt der USIF-Clock clk_{USIF} betrieben. Der Lesezugriff erfolgt durch den Test-Controller. Hierbei wird mit Scan-Takt clk_{SC} jeweils ein Datenwort entsprechend der Datenbreite des Eingangsportes i_{SC} der Testarchitektur ausgelesen. In der Abbildung ist zudem das FIFO-Signal *TestIn_locked* als Teil des USIF-Statusregisters zu sehen.

[1] Die Produktionstestlogik wurde anhand des Embedded Deterministic Test (EDT) sowie des an der BTU entworfenen Scan-Controller-based Test (SCT) veranschaulicht.

Hierdurch soll rechtzeitig signalisiert werden, dass das FIFO für den Schreibzugriff über die Testschnittstelle gesperrt ist, um so das System vor einen Deadlock-Zustand zu bewahren. Die weiteren FIFO-Speicher entsprechen in Aufbau und Funktion dem *TestIn*-FIFO, nur das diese sich zwischen den entsprechenden kommunizierenden Komponenten befinden. Das *TestOut*-FIFO, wie in Abb. 2 zu sehen, reicht die Testausgabedaten aus der Test-Architektur, ggf. gefiltert durch den Analyzer, an die Standardschnittstelle weiter.

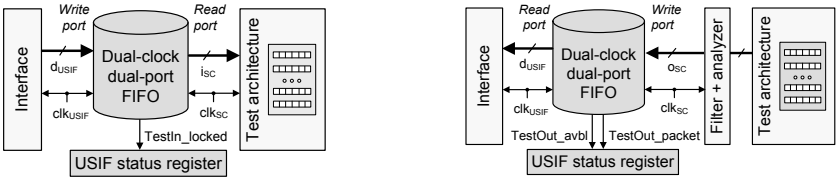

Abb. 2: FIFOs der Testeingabe- (links) und Testausgabedaten (rechts)

2.3 Zentrale Teststeuerung

Der Testzugang und somit der Testablauf wird mittels eines definierten Instruktionssatzes kontrolliert. Die Steuerung des Testzugangs kann durch das Zustandsdiagramm in Abb. 3 nachvollzogen werden. Für den Testzugang durch das USIF wird von einem im speziellen Testmodus befindlichen System ausgegangen. Das heißt, sämtliche von der Standardschnittstelle empfangenen Daten werden als Testdaten, also USIF-Instruktionen bzw. Testmuster für die Testarchitektur, interpretiert.

Vom Ausgangszustand *Idle* wird ein Test durch entsprechende Anweisung gestartet. Hierzu wird der Empfangspuffer des Kommunikations-Controllers ausgelesen, sobald dieser verfügbare Daten enthält. Soll ein interner Selbsttest ausgeführt werden, so wird dieser über den Zustand *RunBIST* initiiert. Im Falle des Zugangs zur internen Testlogik ist zunächst eine Authentifizierung erforderlich. Das in der Abbildung eingerahmte Teildiagramm soll den vor unbefugtem Zugriff geschützten Bereich darstellen.

Im zentralen Zustand *TestControl* findet die Dekodierung der empfangenen USIF-Instruktionswörter statt, um den entsprechenden Zugriff zu initiieren. Um eine Konfiguration, also ein Schreibzugriff auf ein adressiertes Register vorzunehmen, wird der Zustand *WriteConfig* genutzt. Ein Lesezugriff auf ein Konfigurationsregister kann durch den Zustand *ReadConfig* geschehen. Hierbei wird der Registerinhalt angefragt und im darauffolgenden Takt, nachdem der Wert geladen ist, im Zustand *ForwardConfig* in den Sendepuffer des Kommunikations-Controllers geschrieben. Die gestrichelten Pfeile in der Darstellung des Zustandsdiagramms sollen den Zugriff bzw. den Einfluss eines Zustandes auf die entsprechenden Komponenten verdeutlichen.

Ein Lesezugriff auf den Testausgabespeicher wird durch die Zustände *ReadData* und *ForwardData* realisiert. Es wird jeweils ein Datenwort aus dem adressierten

Speicherblock entnommen und an den Sendepuffer des Kommunikations-Controllers weitergereicht. Bei einem Schreibzugriff wird zunächst in den Zustand *WaitData* gewechselt, um auf den Empfang eines Testdatenpaketes mit einem im Instruktionswort spezifizierten Umfang zu warten. Meldet der Kommunikations-Controller verfügbare Daten, wird eine Leseanfrage gestellt, um das erste Datenwort im Zustand *WriteData* in den adressierten Testspeicher zu schreiben. Hier wird in jedem weiteren Takt ein Datenwort gelesen und an den Testspeicher weitergeleitet. Dies wird wiederholt, bis das komplette Datenpaket in den Testspeicher übertragen ist.

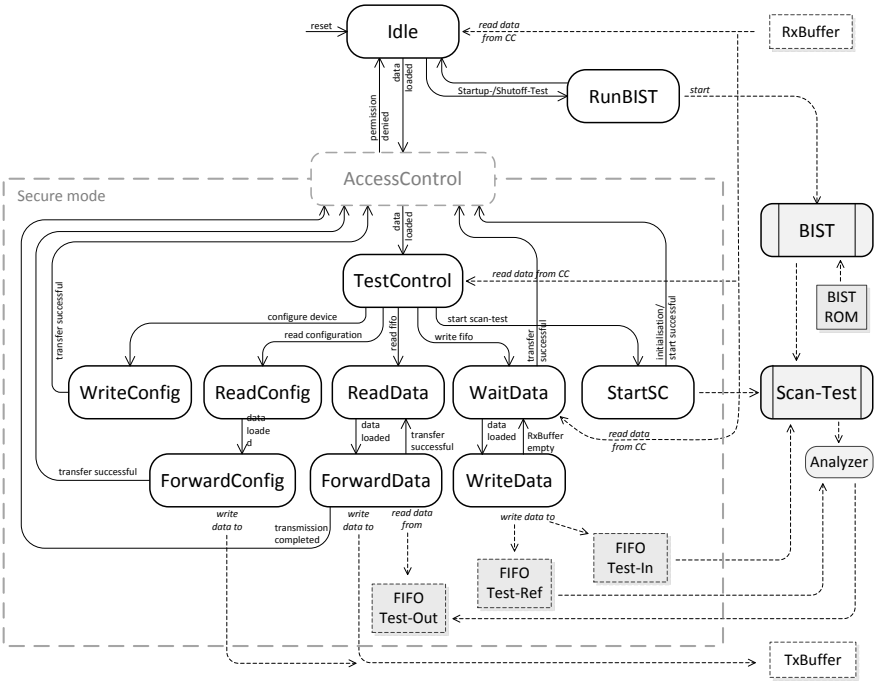

Abb. 3: Vereinfachtes Zustandsdiagramm des USIF-Controllers

Durch den Zustand *StartSC* wird ein Scan-Test initiiert. Hierfür werden die Testkomponenten initialisiert und der Test-Controller gestartet. Nachdem der Scan-Test aktiviert ist, wird direkt wieder in den Zustand *TestControl* gewechselt, um auf den Empfang von Datenpaketen als Testeingaben für den Scan-Test zu warten. Der Test-Controller entnimmt die Testeingaben dem Testspeicher *TestIn*-FIFO.

Die Übermittlung der Testdaten erfolgt paketweise, um den Scan-Test in Zyklen, je nach verfügbaren Testdaten in den Testspeichern, auszuführen. Das heißt, der Testmustersatz wird auf Anwenderseite in Pakete spezifizierter Größe aufgeteilt. Für die eigentliche Übertragung auf dem Buskanal werden die Pakete in Nachrichten dem zugrunde-

liegenden Busprotokoll entsprechend unterteilt. Dieses Schema ist in Abb. 4 veranschaulicht.

Die Übertragung der Testantworten erfolgt auf gleiche Weise. Enthält der Testausgabespeicher ein Datenpaket in entsprechenden Umfang, so wird dieses ausgelesen, um Speicherplatz für weitere Testausgaben freizugeben. Das Datenpaket wird wieder im entsprechenden Nachrichtenformat über den Bus übertragen. Auf Anwenderseite können die Datenpakete entweder direkt ausgewertet oder zu einem kompletten Testantwortsatz zusammengefügt werden.

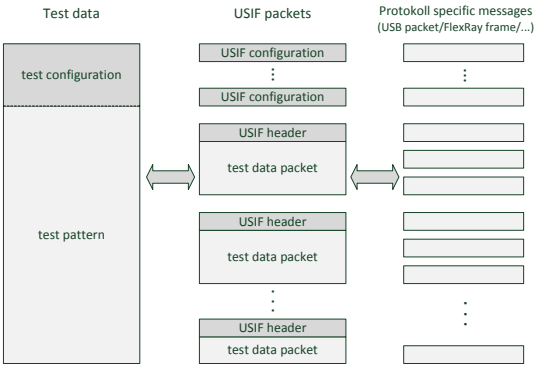

Abb. 4: Paketweiser Datenaustausch

2.4 Test-Architektur

Die industriell genutzten Testtechnologien sind in der Regel Eigentum spezieller EDA-Hersteller, wie z.B. Mentor Graphics, Synopsys oder Cadence, und damit nicht frei verfügbar. Eine der meistgenutzten Methoden ist der *Embedded Deterministic Test* (EDT) von Mentor Graphics [Ra04]. Die Entwicklung der USIF-Architektur erfordert aber eine Lösung, welche frei und ohne Restriktionen verwendet werden kann. Hierzu wird der an der BTU Cottbus entwickelte Scan-Controller als interne Testumgebung in die Architektur eingebunden und somit der sogenannte *Scan-Controller-based Test* (SCT) umgesetzt. Die Basisarchitektur des Scan-Controllers ist in Kooperation mit Infineon Technologies AG im BMBF-Verbundprojekt AZTEKE entstanden [AZ05]. Die Grundlage des Scan-Controllers bildet eine STUMPS-basierte Architektur und beherrscht den strukturorientierten Test sowohl auf statische als auch auf dynamische Fehler. Die Erzeugung der Testmuster beruht dabei auf Pseudozufallssequenzen eines *Linear Feedback Shift Registers* (LFSR) mit nachgeschalteter Musteroptimierung. Die Rückkopplung des LFSR kann innerhalb der Testinitialisierungsphase, aber auch während des Tests, über einen Feedback-Parameter konfiguriert werden. Es handelt sich um ein sogenanntes *Multiple-Polynomial LFSR* (MP-LFSR). Die Komprimierung der Testantworten findet über ein *Multiple Input Signature Register* (MISR) statt. Zusätzlich zu der dadurch umgesetzten *Time Compaction* kann optional eine *Space Compaction*

durch einen XOR-Tree konfiguriert werden. Detaillierte Ausführungen zum Aufbau und Funktionsweise des Scan-Controllers sind in den Vorarbeiten [Ga04, Fr07, Ko08, Ko10] zu finden. In der prototypischen Umsetzung wurde sowohl der SCT als auch der EDT realisiert.

2.5 Testszenarien

Das Ziel dieser Entwicklung ist es, mit derselben Testumgebung optional die folgenden Anwendungsfälle für den Offline-Test zu unterstützen:

- den Produktionstest
  - nach Packaging beim IC-Hersteller,
  - nach Leiterplattenbestückung beim ECU-Hersteller,
- den on-chip Selbsttest im Zielsystem,
- die Fehleranalyse im Zielsystem und
- das Monitoring während des Normalbetriebs.

Diese Szenarien wurden in [Gl12] vorgestellt.

Um die verschiedenen Auswertungsmöglichkeiten umzusetzen, wurden die Analysekomponente sowie ein zusätzlicher BIST-Controller, der den Selbsttest realisiert, entworfen. Dadurch werden folgende Modi der Testanalyse unterstützt:

- *LogAll*, die Ausgabe aller ausgelesenen Testergebnisse,
- *LogFaults*, die Ausgabe der fehlerhaften Testergebnisse inklusive zugehöriger Zeitstempel,
- *PassFail*, die on-chip Auswertung mittels off-chip Referenzdaten, und
- *BIST*, die on-chip Auswertung mittels on-chip Referenzdaten.

3 Applikation

Die Steuerung des USIF-Controllers und somit des diagnostischen Tests soll von einem Anwendersystem aus, das die Kommunikation mit dem peripheren USIF-Gerät beherrscht, möglich sein. Hierzu wurde das Anwendungsprogramm *Universal Scan-Test Interface Applikation*[2] implementiert. Dieses Programm stellt die Schnittstelle der Testumgebung zum Anwender dar. Für die Verbindung zum zu testenden Gerät wird die USB-Schnittstelle[3] genutzt.

[2] Java-Applikation inkl. Benutzeroberfläche, die Eingabemasken zur Konfiguration der Testumgebung und zur Eingabe der Testdaten bereitstellt.
[3] Hierzu wurde als USB-API mit zugehörigem Treiber das Open-Source-Projekt *libusb-win32* eingebunden.

Durch den Anwender angegebene Testdaten[4] können eingelesen, entsprechend umgewandelt und über die zum USIF-Gerät hergestellte Verbindung übertragen werden. Mittels eines Scan-Prozesses ist der Testablauf, die paketweise Übertragung der Testeingaben und -ausgaben über die USB-Schnittstelle und die Testauswertung, automatisiert. Hierzu wird zunächst der Testsatz aus der eingelesenen Datei je nach Format extrahiert und in einem Bytearray für die Testeingaben *arrayIn* und ggf. in einem separaten Array für die Referenzwerte *arrayRef* abgespeichert. Der weitere Ablauf ist in Abb. 5 dargestellt.

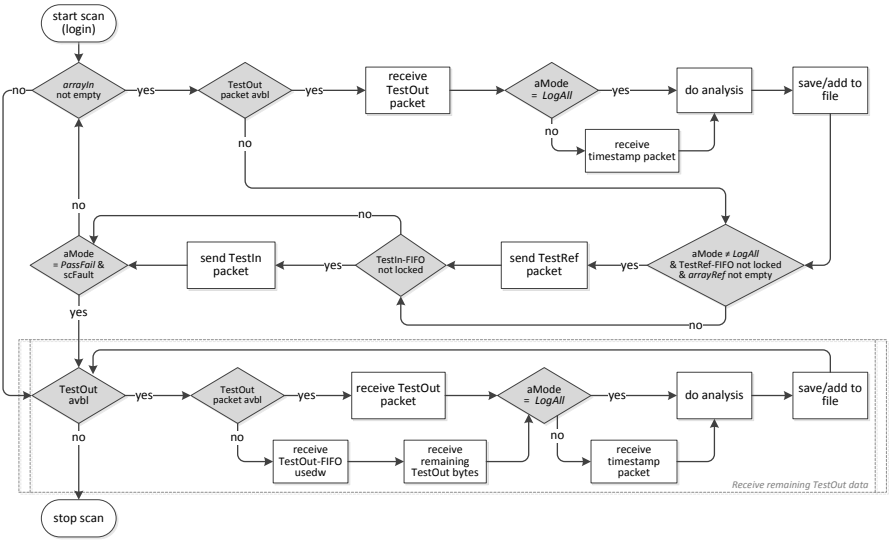

Abb. 5: Datenfluss des Scan-Prozesses

3.1 Extraktion der Testmuster

Das Testprogramm für den Scan-Controller wird durch einen zugehörigen ATPG-Prozess erstellt und als Bitstrom aufbereitet in einer Pattern-Datei abgespeichert. Somit kann die Applikation die erstellte Pattern-Datei einlesen, direkt als Bitstrom interpretieren und an das USIF übertragen.

Die durch kommerzielle ATPG-Programme erzeugten Testmuster liegen in einem standardisierten Format einer *Pattern Description Language* (PDL) vor. Gebräuchliche Pattern-Formate sind *Standard Test Interface Language* (STIL1450, STIL2005), *Core Test Language* (CTL), *Waveform Generation Language* (WGL), *Texas Instruments Test Description Language* (TDL 91). Diese werden üblicherweise durch ATEs interpretiert, um die BIST-Architektur im Fertigungstest anzusteuern.

[4] Pattern-Dateien in SCT-, STIL-, CTL-, TDL-Formaten.

Die Applikation setzt eine Pattern-Extraktion aus einer solchen Dateischnittstelle um. Somit kann ein durch das USIF angesteuerter EDT mittels vorliegender EDT-Pattern betrieben werden. Die USIF-Applikation unterstützt hierzu die Pattern-Formate STIL, CTL und TDL91.

4 Auswertung

4.1 Testeigenschaften

Um Messungen der Testzeiten für verschiedene Schaltungen und Testkonfigurationen vorzunehmen, wurde jeweils die zu testende Schaltung in entsprechender Auslegung der Scan-Struktur und die USIF-Architektur, inklusive der Testarchitektur und der Kommunikations-Controller, synthetisiert und auf FPGA-Basis implementiert. Der Testablauf wird komplett durch die Applikation auf dem PC gesteuert.

Den anschließenden Betrachtungen lag der folgende Versuchsaufbau zugrunde:

- Altera Cyclone II FPGA (EP2C70F896C6N),
- Aufsteckplatine für Schnittstellen:
  - USB Transceiver Texas Instruments TUSB1106, USB 2.0-Buchse Typ Mini-B,
  - FlexRay Transceiver NXP Semiconductors TJA1080A , D-Sub DE9-Buchse,
- Applikation betrieben auf PC Intel Core i7-2600 CPU 3,40 GHz (Arbeitsspeicher 8 GB), OS Windows 7 (64 Bit),
- Testzugang: USB FullSpeed, Bulk-Transfer.

Für die Untersuchung von Testzeiten wurden ISCAS89-, ITC99-Schaltungen und VLIW-Prozessoren[5] in verschiedenen Auslegungen der Scan-Struktur herangezogen. Als Testarchitektur wurde der Scan-Controller eingebunden und dementsprechend die komprimierten Testsätze durch den zugehörigen ATPG-Prozess ermittelt.

Der Ermittlung der Testzeit lag folgende Konfiguration zugrunde:

- Fehlermodell: Stuck-at,
- Scan-Clock: f_{SC} = 20 MHz,
- Scan-Controller Eingangsport: i_{SC} = 16 Bit, Ausgangsport: o_{SC} = 16 Bit,
- Scan-Modus: *Decompress* (LFSR-basierte Testmustererzeugung),
- Kompaktierung: *Mixed* (MISR + XOR-Tree),
- Analysemodus: *LogAll*, Analyse-Rate: (a_{Rate}, a_{Base}) = (m_{Chain}, m_{Chain}-*1*)[6],
- Paketgröße: g_{Packet} = 512 Byte.

[5] Superskalare Architektur mit statischer Programmablaufplanung. Die generische Hardware-Beschreibung ermöglicht es, den VLIW-Prozessor in diversen Konfigurationen zu synthetisieren [Sc06].
[6] Es wird hier jeweils eine Signatur pro Testmuster (bestehend aus m_{Chain} Testvektoren) erzeugt.

In Tab. 1 ist eine Auswahl der untersuchten Schaltungen zusammengefasst. Zu jeder Schaltung der Spalte 1 wurde ein Scan-Design mit unterschiedlicher Anzahl an Scan-Ketten erstellt. In der zweiten Spalte ist die Scan-Struktur dargestellt. Da die Scan-Flipflops zu möglichst gleichlangen Ketten aufgeteilt werden, ergibt sich die Scan-Tiefe m_{chain} aus der spezifizierten Anzahl an Scan-Ketten n_{chain}. Die Spalte 3 enthält die Anzahl der durch den ATPG-Prozess erhaltenen Testmuster. Die durch diese Testmuster erreichbare Fehlerüberdeckung ist in der darauffolgenden Spalte zu finden. Der Datenumfang des komprimierten Testsätze ist in Spalte 5 zu sehen. Als komprimierter Testsatz ist das ermittelte Scan-Controller-Programm zu verstehen [Ga04, Ko08, Gl12]. In der Spalte 6 ist die nötige Testzeit für einen Selbsttest, dem der komprimierte Testsatz für einen deterministischen Test on-chip zur Verfügung steht, dargestellt. In den weiteren Spalten sind die mittels des beschriebenen Versuchsaufbaus eruierten mittleren Testzeiten zu sehen.

	Testschaltung	Scan-Ketten n_{chain}/m_{chain}	Testmuster n_{Pat}	Fehlerüber-deckung (stuck-at) %	Testsatz [kByte]	Zeitaufwand Selbsttest [ms]	Testzeit USB [s]	Testzeit FlexRay[7] [s]
ISCAS89	s13207	128/7	262	94,10	10,03	0,256	0,104	-
		256/4			8,97	0,230	0,089	0,396
	s15850	64/11	155	94,30	9,14	0,234	0,093	-
		128/6			8,07	0,207	0,084	0,310
ITC99	b18	256/11	530	95,38	88,44	2,264	0,906	3,592
		512/6			97,82	2,504	1,029	-
	b19	256/22	538	95,32	176,75	4,525	1,774	8,543
		512/11			193,27	4,958	2,059	-
VLIW16S4[8]		256/12	454	96,21	60,39	1,546	0,541	2,509
		512/6			63,48	1,625	0,620	-
VLIW32S8		256/18	790	98,56	261,62	6,698	2,724	11,067
		512/9			279,63	7,159	2,949	-

Tab. 1: Testaufwand, Fehlerüberdeckung und Testzeit

Für den Selbsttest kann in jedem Scan-Takt ein Datum aus dem komprimierten Testsatz entnommen und dieses an den Eingangsport des Scan-Controllers angelegt werden. Somit ist die Testzeit nur von der Anzahl der Eingaben und der Taktfrequenz des Scan-Controllers f_{SC} abhängig. Für ein Testsatzumfang von g_{Test} Byte und einem Eingangsport der Datenbreite i_{SC} gilt demnach:

[7] Für den Testzugang über FlexRay wurden nur die Testzeiten zu dem jeweils kompakteren Datensatz ermittelt.
[8] Dem Bezeichner ist hier die Anzahl der parallelen Datenpfade (Slots) und die Datenbreite eines Datenpfades zu entnehmen. Der VLIW16S4 bspw. wurde als 16-Bit-Architektur mit 4 Slots synthetisiert.

$$t_{BIST} = \frac{g_{Test}}{i_{SC}/8} * \frac{1}{f_{SC}} \Rightarrow t_{BIST} = \frac{n_{in}}{f_{SC}},$$

wobei $n_{in} = 8g_{Test} / i_{SC}$ die Anzahl der Eingangsdatenworte des Testsatzes ist.

Die Werte für die Testzeit eines über die Standardschnittstellen durchgeführten Scan-Tests wurden durch die Applikation, über die der kompletten Testlauf organisiert wird, ermittelt. Ein Testlauf umfasst hier das Einlesen der Testdaten aus einer spezifizierten Pattern-Datei, der in Abb. 5 beschriebene Scan-Prozess und das Aufbereiten und Speichern der Testergebnisse. Die Testzeit lässt sich folgendermaßen ausdrücken:

$$t_{Test} \approx \frac{g_{Test} + g_{Res}}{r_{Eff}} + t_{Init} + t_{Req} n_{Packet} + t_{Res}.$$

Sämtliche Testdaten, deren Umfang sich sowohl aus dem der Testeingaben g_{Test} als auch aus dem der Testantworten g_{Res} ergibt, werden mit der effektiven Datenrate r_{Eff} übertragen. Hinzu kommen die Zeiten, die für die Initialisierung des Tests t_{Init} und für die Auswertung der Testantworten t_{Res} benötigt werden. Außerdem fällt eine bestimmte Zeit für die Statusanfragen t_{Req} pro Datenpaket an.

Die Initialisierungsphase der Applikation besteht neben der Konfiguration des USIF-Gerätes hauptsächlich aus der Extraktion der Testdaten aus der einzulesenden Pattern-Datei. Die hierfür aufzuwendende Zeit ist also vor allem von dem Datenumfang des Testsatzes abhängig, fällt aber gegenüber der Sendezeit eines umfangreichen Testsatzes relativ gering aus. Da der Analysemodus *LogAll* gewählt und kein Referenzdatensatz spezifiziert wird, fällt auch keine umfangreiche Auswertung der Testergebnisse durch die Applikation an. Somit kann sowohl die Initialisierungszeit als auch die Analysezeit hier vernachlässigt werden. Die Gesamttestzeit wird also auf die Übertragung der Testdaten und der dazu nötigen Kommunikation (Status- u. Request-Botschaften) beschränkt.

Sei hier der Zugang exemplarisch über USB betrachtet, mit den Testzeiten aus Spalte 7. Es ist zu erkennen, dass die Testzeit lediglich von der Größe des Testsatzes abhängt. Der eingestellte Scan-Takt und die Scan-Tiefe, die die Zeit für die Shift-Phase des Scan-Tests bestimmt, wirkt sich nicht auf die Gesamtzeit aus, da zum einen der Test parallel zur Übertragung weiterer Testdaten geschieht. Zum anderen ist die on-chip Verarbeitungsgeschwindigkeit um ein Vielfaches höher als die Übertragungs-geschwindigkeit, sodass die Applikation nicht auf das Hineinschieben der Testmuster in die Teststrukturen warten muss. Da aber vor der Übertragung der Datenpakete sichergestellt werden soll, dass genügend freier Speicherplatz im Empfangspuffer des USB-Controllers bzw. auszulesender Speicher im *TestOut*-FIFO vorhanden ist, erfolgt eine regelmäßige Abfrage des USB-Statusendpunktes. Das bedeutet, es entstehen Wartezyklen pro Datenpaket, die sich negativ auf die Testzeit auswirken.

Da die Übertragung paketweise geschieht und für jedes Paket ein nicht unerheblicher administrativer Aufwand durch die Applikation hinzukommt, ist es hier günstig, die Testzeit über die Zeit pro Paket auszudrücken. Die Paketanzahl n_{Packet} ergibt sich aus dem Umfang des zu sendenden Testsatzes g_{Test}, des zu empfangenen Testantwortsatzes g_{Res} und der spezifizierten Paketgröße g_{Packet}. Da über die Applikation die Gesamtzeit eines Testlaufs ermittelt wurde, kann auf die Zeit pro Paket wie folgt geschlossen werden:

$$t_{Packet} = \frac{t_{Test}}{n_{Packet}} = \frac{t_{Test}}{\lceil g_{Test}/g_{Packet} \rceil + \lceil g_{Res}/g_{Packet} \rceil}.$$

Die Zeiten für die Initialisierung und die Auswertung des Scan-Tests fließen hier in die Paketzeit mit ein. Diese Darstellung der Paketzeit korreliert mit den gemessenen Werten (siehe Tab. 1). Für die unterschiedlichen Testschaltungen ergeben sich in etwa gleiche Zeiten für ein Datenpaket. Die Paketzeit beträgt hier im Mittel um die 5 ms. Daran ist zu sehen, dass ein erheblicher Zeitaufwand durch die Applikation verursacht wird. Über den Bus wird annähernd die volle Datenrate erreicht. Für die USB-Verbindung wurde eine Datenrate für die Nutzdaten von etwa 8 MBit/s ermittelt. Das heißt, ein Paket von 512 Byte kann in 0,512 ms übertragen werden. Der fast zehnfache Wert für ein Paket ist bedingt durch die zeitaufwendigen Statusabfragen.

Die Applikation wurde nicht auf optimale Laufzeit, sondern in erster Linie auf garantierten fehlerfreien Datenaustausch ausgelegt. So wird bspw. vor und nach jedem Zugriff, innerhalb der Sende- und Empfangsfunktion (siehe Abb. 5), auf eine aktualisierte Statusabfrage, die durch den USB-Host mittels Polling im Intervall von 1 ms erfolgt, gewartet. Es wird vorher sichergestellt, dass ein Zugriff erfolgen darf und danach geprüft, ob der Zugriff erfolgreich war.

4.2 Hardware-Aufwand

Um eine Aussage über den Hardware-Aufwand des entwickelten Designs zu machen, wurden durch Synthese, basierend auf der *Si2 NanGate FreePDK45 Generic Open Cell Library*, die Logikzellen und die dadurch eingenommene Chipfläche der einzelnen Komponenten ermittelt. In Tab. 2 sind die resultierenden Chipflächen und der relative Aufwand dargestellt. Hierzu wurde der Hardware-Aufwand exemplarisch einem VLIW-Prozessor, in einer Auslegung als 32-Bit-Architektur mit 8 Slots, gegenübergestellt. Dieser enthält eine Scan-Struktur mit 256 parallelen Scan-Ketten. Dementsprechend wurde der Scan-Controller für 256 Scan-Ketten ausgelegt. Dies betrifft den Eingangsport (für zwei 8-Bit-Adressen) und interne Komponenten wie ALU und MISR.

Die internen Speicherblöcke, wie die Testdatenspeicher des USIF-Controllers und der ROM des Selbsttests, sind hier nicht in die Betrachtung einbezogen worden. Diese Speicher sind in der implementierten Architektur generisch ausgelegt, um die Größe je nach Anwendungsfall anpassen zu können.

Komponenten	Chipfläche in µm²	Relativer Aufwand zur Beispiel-CUT
VLIW32S8	189.993	
Scan-Controller	6.487	3,41 %
USIF-Architektur	3.957	2,08 %
USIF-Controller	2.269	1,19 %
Config-Register	658	0,35 %
BIST-Controller	209	0,11 %
Analyzer	791	0,42 %

Tab. 2: Hardware-Aufwand der USIF-Architektur

Es ist zu erkennen, dass das Design für den Testzugang zur internen Produktionstestlogik gegenüber einem komplexen Prozessor sehr gering ausfällt. Da im eingebetteten System bereits vorhandene Standardschnittstellen und Sicherheitsmodule genutzt werden sollen, wird also nur verhältnismäßig wenig Zusatzlogik für die Verknüpfung der Anwendungsschnittstelle mit der Testarchitektur benötigt.

5 Zusammenfassung

Mittels des entworfenen Konzeptes ist es möglich, eine feingranulare Fehlerdiagnose von hochintegrierten Schaltungen auch in der Postproduktionsphase durchzuführen.

Für eingebettete Prozessoren bedeutet dies, dass im Falle einer Fehlfunktion des Systems eine vorhandene Anwendungsschnittstelle als Testzugang für einen detaillierten diagnostischen Test dient, somit also keine aufwendige Demontage für nähere Untersuchungen notwendig ist. Neben der gewonnenen Diagnosefähigkeit im Feld kann der Testzugang bereits den Produktionstest beim Komponentenhersteller unterstützen.

Eine weitere Testumgebung ist die feingranulare Fehleranalyse von Rückläufern. Der IC kann, ohne diesen von der Platine lösen zu müssen, beim Komponenten- oder Halbleiterhersteller detailliert untersucht werden, um mögliche Fehler in der Schaltungsstruktur zu diagnostizieren.

Ein wichtiger Punkt, der hier nochmals Erwähnung finden muss, ist der Schutz vor unbefugtem Zugriff. Der Zugang über standardisierte Schnittstellen zu internen Strukturen eines Prozessors, erfordert adäquate Maßnahmen gegen unautorisierte Kommunikation und missbräuchliche Anwendungen. Es gibt bewährte industriell genutzte Hardware-Lösungen, wie bspw. der von deutschen Automobilherstellern angewandte SHE-Standard, und darauf basierende oft herstellerspezifische Verfahren, die Authentifizierung und Verschlüsselung für Systemzugriffe gewährleisten [SH09].

Bei einer angemessenen Lösung für sicheren Zugriff bietet die präsentierte Testanbindung, die gegenüber einem komplexen IC eine Zusatzlogik mit relativ geringem Hardware-Aufwand darstellt, einen erheblichen Nutzen bezüglich Systemzuverlässigkeit und Diagnosefähigkeit.

Literaturverzeichnis

[Ab14] Abelein, U.; Cook, A.; Engelke, P.; Glaß, M.; Reimann, F.; Russ, T.; Teich, J.; Wunderlich, H.-J.: Non-Intrusive Integration of Advanced Diagnosis Features in Automotive E/E-Architectures. In Proc. IEEE Design, Automation & Test in Europe Conference & Exhibition, S. 1-6, 2014.

[AZ05] Applikationsspezifische Testmethodik für hochkomplexe Systeme der Kommunikations- und Kraftfahrzeugtechnik: BMBF-Verbundprojekt AZTEKE; Abschlussbericht. Technischer Bericht, ATMEL Germany GmbH, Infineon Technologies AG, Philips Semiconductors GmbH, 2005.

[DI13] Durchgängige Diagnosefähigkeit in Halbleiterbauelementen und übergeordneten Systemen zur Analyse von permanenten und sporadischen Elektronikausfällen im Gesamtsystem Automobil: BMBF-Verbundprojekt DIANA, Technischer Bericht, Audi AG, Conti Temic Microelectronic GmbH, Infineon Technologies AG, Zentrum Mikroelektronik Dresden AG, Ingolstadt, 2013.

[Fr07] R. Frost, D. Rudolph, C. Galke, R. Kothe & H.T. Vierhaus. A Configurable Modular Test Processor and Scan Controller Architecture. In Proc. IEEE International On-Line Testing Symposium, S. 277-284, 2007.

[Ga04] Gätzschmann, U.; Galke, C.; Vierhaus, H. T.: Ein flexibles Verfahren zur Testdaten-Kompaktierung und -Dekompaktierung für den Scan-Test. In Proc. GI/GMM/ITG Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen, 2004.

[Gl12] Gleichner, C.; Vierhaus, H. T.; Engelke, P.: Scan Based Tests via Standard Interfaces. In Proc. IEEE Euromicro Conference on Digital System Design, S. 844-851, 2012.

[IJ13] P1687/D1.62 - IEEE Draft Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device. IEEE Association, 2013.

[JT01] IEEE 1149.1-2001 Standard Test Access Port and Boundary-Scan Architecture. IEEE Association, 2001.

[Ko08] Kothe, R.; Vierhaus, H. T.: A Scan-Controller Concept for Low-Power Scan Tests. Journal of Low-Power Electronics, American Science Publishers, Vol. 4, S. 1-9, 2008.

[Ko10] Kothe, R.; Vierhaus, H. T.: Test Data and Power Reductions for Transition Delay Tests for Massive-Parallel Scan Structures. In Proc. IEEE Euromicro Symposium on Digital Systems Design, S. 283-290, 2010.

[Ra04] Rajski, J.; Tyszer, J.; Kassab, M.; Mukherjee, N.: Embedded Deterministic Test. IEEE Transactions on Computer-Aided Design of Integrated Circuits & Systems, Vol. 23, S. 776-792, 2004.

[Sc06] Schölzel, M.: Automatisierter Entwurf anwendungsspezifischer VLIW-Prozessoren. Dissertation, BTU Cottbus, 2006.

[SH09] Secure Hardware Extension - Functional Specification V1.1, Rev. 439. HIS-Herstellerinitiative Software, 2009.

[VS14] VehicleServicePros.com, D. Kolman. DIANA is Looking into Vehicle Electronic Controls. Juli 2010. http://www.vehicleservicepros.com/blog/10342815/diana-is-looking-into-vehicle-electronic-controls, [Stand 28.04.2015].