Design for Testability (DFT) Strukturen für ASIC-Design und ihre Emulation auf FPGA

Benjamin Dusch, B.Eng.
Hochschule Offenburg, Badstraße 24
0781 / 205 267, benjamin.dusch@fh-offenburg.de

Mit dem Übergang zu immer komplexeren Designs an der Hochschule Offenburg werden DFT-Strukturen wie „Boundary Scan“ und „Scan“ in ASIC-Designs notwendig. Die DFT-Struktur Scan wird hierbei zukünftig bei Implementierung eines speziellen Scan Chain der Core Logic des ASIC-Designs verwendet und danach in der Boundary Scan Architektur integriert.
Zunächst werden die Strukturen im recht einfachen ASIC-Design „Rolling Dice“, entwickelt am IAF der Hochschule Offenburg, implementiert.
Nach Verifizierung der Funktionalität der Strukturen durch Emulation erfolgt die Einführung in komplexere ASIC-Design wie Front-End ASIC DQPSK sowie Prozessor-ASIC PDA V.2 (beide ebenfalls entwickelt am IAF der Hochschule Offenburg).
Eine Verifizierung der mit DFT-Strukturen ausgestatteten komplexeren ASIC-Design erfolgt im Rahmen dieser Ausarbeitung nicht, Bezug genommen wird hauptsächlich auf die Einführung der DFT-Strukturen in das ASIC-Design des „Rolling Dice“.
Ein Vergleich von Aufwand gegenüber Nutzen bei Implementierung von DFT-Strukturen in „kleine“ gegenüber „große“ ASIC-Design bildet ein wichtiges Fazit.

1. Einleitung

1.1 Motivation

In einem Zeitalter, in welchem Integrierte Schaltungen immer mehr an Komplexität zulegen, mikroelektronische Schaltungen aus immer mehr Bausteinen bestehen, ist es, um der Komplexität gerecht zu werden, zwingend nötig, eine mikroelektronische Schaltung eines ASICs (Application Specific Integrated Circuit) um zwei im Folgenden erläuterte Funktionen zu erweitern.
Die erste Funktionserweiterung ist die automatische Testbarkeit des ASICs nach der Herstellung. Die bisher am IAF nur ausgeführten manuellen funktionalen Tests sollen durch „Automatische Tests“ ersetzen werden. Das heißt, dass nach Herstellung des ASICs man diesen über eine bestimmte Testlogik, welche der funktionalen ursprünglichen mikroelektronischen Schaltung hinzugefügt wurde, ohne großen Aufwand testen können muss, um so die Verifikation der Funktion des ASICs nach der Herstellung zu vereinfachen.
Die zweite Funktionserweiterung ist die Debugging-Funktion eines ASICs, welche die Entwicklung eines ASICs (Application Specific Integrated Circuit) vereinfachen und damit die Entwicklungszeit verkürzen soll. Sinnvoll ist es von extern sämtliche Registerinhalte (Zustände der Flip Flops) während des Betriebs im Einzeltaktbetrieb überwachen und frei manipulieren zu können.
Im Rahmen dieser Arbeit werden beide Funktionserweiterungen eines ASIC-Designs erzielt.

Abbildung 1: Verdeutlichung des Einzeltaktbetriebs zwecks Debugging eines ASICs

1.2. Design for Testability Strukturen

Bei DFT handelt es sich um Designtechniken in der Mikroelektronik, bei deren Anwendung mikroelektronische Schaltungen mit weiterer Testlogik erweitert werden, mit dem Ziel adäquate Testabdeckung bei gleichzeitiger Vereinfachung von Testmechanismen zu erzielen.
Verwendet werden in dieser Ausarbeitung als Designtechniken die DFT-Struktur Scan zur Implementierung eines Scan Chain in die Core Logic des ASIC-Designs (als Core Logic wird die gesamte funktionale Logik eines ASIC-Designs bezeichnet) sowie übergeordnet die DFT-Struktur Boundary Scan.
Die genannte Testlogik soll eine Boundary Scan Architektur sein, wobei es sich bei der Boundary Scan Architektur eher um eine Teststeuerungslogik handelt, durch welche andere integrierte Testlogiken kontrol-

liert werden. Es sollen durch die Boundary Scan Teststeuerungslogik innerhalb der Boundary Scan Architektur integrierte Scan Chain implementiert und gesteuert werden.

Für die Core Logic des ASIC-Designs wird zunächst ein Scan Chain implementiert.
Dadurch werden alle Flip Flops der Schaltung des ASICs auslesbar, steuerbar und manipulierbar gemacht.
Es werden hierfür bei der „Scan Insertion“ alle Flip Flops der Core Logic durch spezielle Scan Flip Flops mit integrierter Testlogik ersetzt, welche in einen Scan-Testmodus versetzt werden können.
Um diesen Scan-Testmodus nutzen zu können wird die Boundary Scan Architektur implementiert.

Im Normalmodus des ASICs nehmen die Scan Flip Flops ihre ursprüngliche Funktion innerhalb des Designs ein und die Scan-Funktionalität hat keinen Einfluss auf deren normale Funktion.
Der Testmodus hingegen kann aktiviert werden mittels Aktivieren einer speziellen Boundary Scan Instruction für den Scan Chain. Wird dieser Testmodus aktiviert, werden alle Scan Flip Flops seriell zu einem Schieberegister verschaltet, bilden einen Scan Chain von TDI nach TDO, welcher über Schiebevorgänge ausgelesen oder beliebig auf Werte aus 0 oder 1 gesetzt werden kann.

Dieses serielle Schieberegister wird als Scan Chain der Core Logic (siehe Abbildung unten) bezeichnet und ist also in einer die Steuerung des Scan Chain übernehmenden Teststeuerungslogik, der Boundary Scan Architektur, als Test Data Register (TDR) eingebunden.

Anstatt aufwendiger manueller Tests eines ASICs können so mittels der DFT-Struktur Boundary Scan und integrierter DFT-Struktur Scan „Automatische Tests“ auf einem ASIC-Design ausgeführt und desweiteren der ASIC im Debugging-Modus betrieben werden.

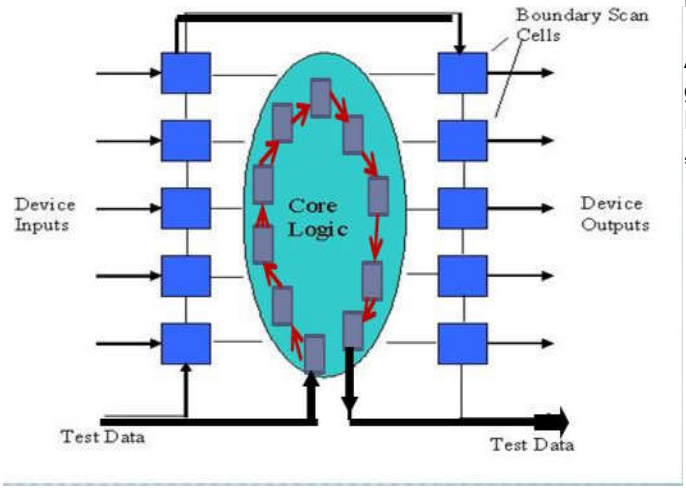


Abbildung 2: ASIC mit Boundary Scan und Scan (-Chain)

2. Allgemeine Herangehensweise

Der Einfachheit wegen werden die Funktionserweiterungen Debugging und Automatische Tests zunächst in einem einfachen ASIC-Design eingeführt, im vom IAF der Hochschule Offenburg entwickelten Design des „Rolling Dice“. Es handelt sich hierbei um ein ASIC Design, bei welchem bei Aktivierung des ASICs eine Zahl „erwürfelt“ wird. Dies geschieht, indem der Anwender eine frei gewählte Zeit lang eine Taste betätigt, ein Signal auf den ASIC gibt, damit die Logik aktiviert, welche eine Zahl von 1 bis 6 errechnet und über LED-Ausgabe anzeigt.
An diesem Design kann leicht nach Einführen der zusätzlichen DFT-Strukturen die korrekte Funktion des funktionalen Designs mittels manueller funktionaler Tests sowie auch der Testlogik von Boundary Scan und Scan nachgewiesen werden.
Der Nachweis der Funktionalität von Boundary Scan und Scan stellt zugleich den Nachweis der Funktionserweiterung „Automatische Tests“ eines ASIC-Designs dar. Automatische Tests werden im Umfang dieser Arbeit nur auf Basis funktionaler Boundary Scan Design (BSD)-Test-Pattern durchgeführt, wobei das Maß der Testabdeckung ermittelt werden wird.
Eine größere Testabdeckung ergibt sich bei Erzeugung und Verwendung scan-basierender ATPG (Automatic Test Pattern Generation)-Test-Pattern, die für die Automatischen Tests in Kombinationen mit den funktionalen BSD-Test-Pattern verwendet werden. Diese Art der Test-Pattern werden im Umfang dieser Arbeit nicht erzeugt.
Nach Bestätigung der Funktion der Automatischen Tests des ASIC-Designs über die Boundary Scan Architektur wird am ASIC-Design des „Rolling Dice“ eine Lösung gesucht, um hardware- wie softwareseitig mit dem fertigen emulierten ASIC-Design kommunizieren und das Design debuggen, im Einzeltaktbetrieb betreiben zu können.
Können die gewünschten Funktionserweiterungen des ASIC-Designs des „Rolling Dice“ bestätigt werden, erfolgt die Einführung der DFT-Strukturen in das komplexe ASIC-Design des „PDA V.2“.

An dieser Stelle kann dann ein kurzer Vergleich vorgenommen werden, welcher Aufwand gegenüber Nutzen bei Implementierung von DFT-Strukturen in „kleine“ gegenüber „große“ ASIC-Design thematisiert.

3. Scan/Boundary Scan Insertion in ASIC-Design

3.1 Einleitung

Die Einführung von Scan in die Core Logic von ASIC-Design und Boundary Scan in ASIC-Design erfolgt mittels des professionellen Programmes Design Vision von Synopsys durch die Skriptsprache Tickle (TCL), über welche Befehle an das verwendete Programm gegeben werden. Jeweils zusammengehörende Befehle werden zusammengefasst zu einem auszuführenden Skript, welches per Aufruf über die Kommandozeile ausgeführt wird.

Es werden folgende Skripte ausgeführt:

3.2 Scan Insertion

Skript 1: Definition der zu verwendenden Bibliotheken
Zunächst werden die bei der Synthese zu verwendenden Technologien als Bibliotheken in einem Skript definiert.

Skript 2: Definition von nicht zu verwendenden Standardzellen
Es werden anschließend in einem weiteren Skript die Standardzellen definiert, welche in der Synthese nicht verwendet werden sollen. Hier werden nun im Gegensatz zur normalen Synthese eines Designs Standard Flip Flops verboten, d.h. nur Scan Flip Flops zugelassen.

Skript 3: Einlesen, Verlinken und Prüfen des Designs
In diesem Skript werden die Quellcodes des ASIC-Designs eingelesen und das Design verlinkt sowie auf Korrektheit/Konsistenz geprüft.

Skript 4: Design Rule Checking des Designs
Bevor die Synthese erfolgen kann, wird mittels RTL Test Design Rule Checking (DRC) das Design auf mögliche Verletzungen von Design Rules gecheckt.

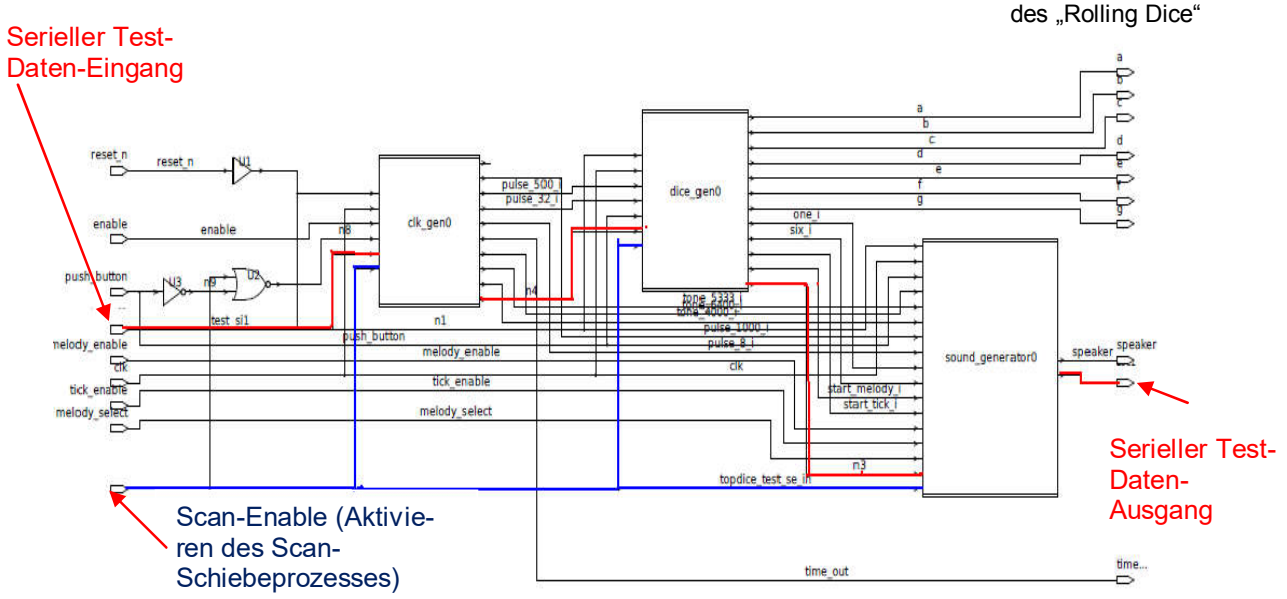
Skript 5: Synthese: test-ready-compile des Designs
Das Skript führt die eigentliche Synthese auf Gatter-Ebene durch und optimiert das Design hinsichtlich der zu erfüllenden constraints, also Auflagen, die das Design erfüllen muss und prüft letztlich nach dem Test-Ready Compilation nochmals die Einhaltung der Design Rules.

Skript 6: Definition und Implementierung des Scan Chains
Nun können die Definitionen für das Implementieren des Scan Chain erfolgen und der Scan Chain eingeführt werden.

Skript 7: Design Rule Checking des Scan-Designs
Nach Einführen des Scan Chain muss erneut das Design auf Verletzung Design Rules geprüft werden.

Das Ergebnis der Scan Insertion (der serielle Pfad durch die gesamte Core Logic) ist folgend am Design des „Rolling Dice“ dargestellt:

Abbildung 3: Scan Chain der Core Logic des „Rolling Dice“



3.3 Boundary Scan Insertion

Implementieren des Test Access Ports (TAP)
Bevor Boundary Scan per Skripte implementiert wird, erfolgt vorab die grafische Erweiterung des Designs um den TAP im professionellen Programm HDL Designer von Mentor.

Skript 1: Definition und Implementierung des Boundary Scan
Das Skript führt sämtliche Befehle aus, die zum Einführen der Boundary Scan Architektur samt TAP nötig sind.

Skript 2: Verifizieren des Boundary Scan Designs gemäß IEEE 1149.1
Die Einführung von Boundary Scan muss anschließend verifiziert werden und hinsichtlich der Einhaltung des Standards IEEE 1149.1 geprüft werden. Dafür muss das Design vorbereitet, ein spezieller Flow durchlaufen werden.

Skript 3: Erstellen von Test-Pattern sowie der BSDL-Datei
Zum Testen der Boundary Scan Logik werden in einer Verilog Test-Bench integrierte funktionale Boundary Scan Test Pattern herausgeschrieben, die später auf Basis der Netzliste des fertigen Designs simuliert werden. Auch wird ein BSDL-File (Boundary Scan Description Language File) erstellt, welches in IEEE 1149.1 definiert ist und der Beschreibung der Testfähigkeiten des Designs dient.

Skript 4: Herausschreiben einer Netzliste für Emulation
Um das Design auf dem FPGA-Emulations-TestBoard der Hochschule Offenburg nachbilden und testen zu können, muss vom fertigen Scan-Design eine spezielle Netzliste geschrieben werden.

Das fertige Boundary Scan Design des „PDA V.2“ ist in der unteren Abbildung dargestellt.
Das implementierte Boundary Scan Register, der endliche Zustands-Automat „TAP-Controller“ mit Boundary Scan Architektur neben der Core Logic mit Peripherie und PLL des ASIC-Designs, sind zu sehen.

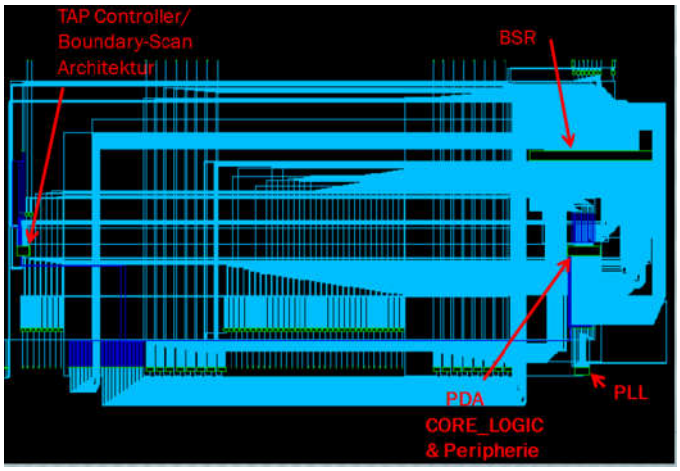


Abbildung 4: Boundary Scan Design des „PDA V.2“

4. Automatische Tests/Automatische Verifikation der Funktion des ASIC-Designs

Zum Erreichen der Fähigkeit Automatischer Tests wird die implementierte Boundary Scan Architektur mit integriertem Scan Chain der Core Logic verwendet.
Durch automatische Steuerung der Testlogik am Test Mode Select (TMS) und Einfüttern von Bitfolgen (Test-Pattern) am Test Data Input (TDI) bei gleichzeitiger spezieller Taktung des Systems über die Test Clock (TCK) und Überwachen der am Test Data Output (TDO) ankommenden Bitfolgen, kann das ASIC-Design durch Verwendung von Test-Pattern verifiziert und automatisch getestet werden.

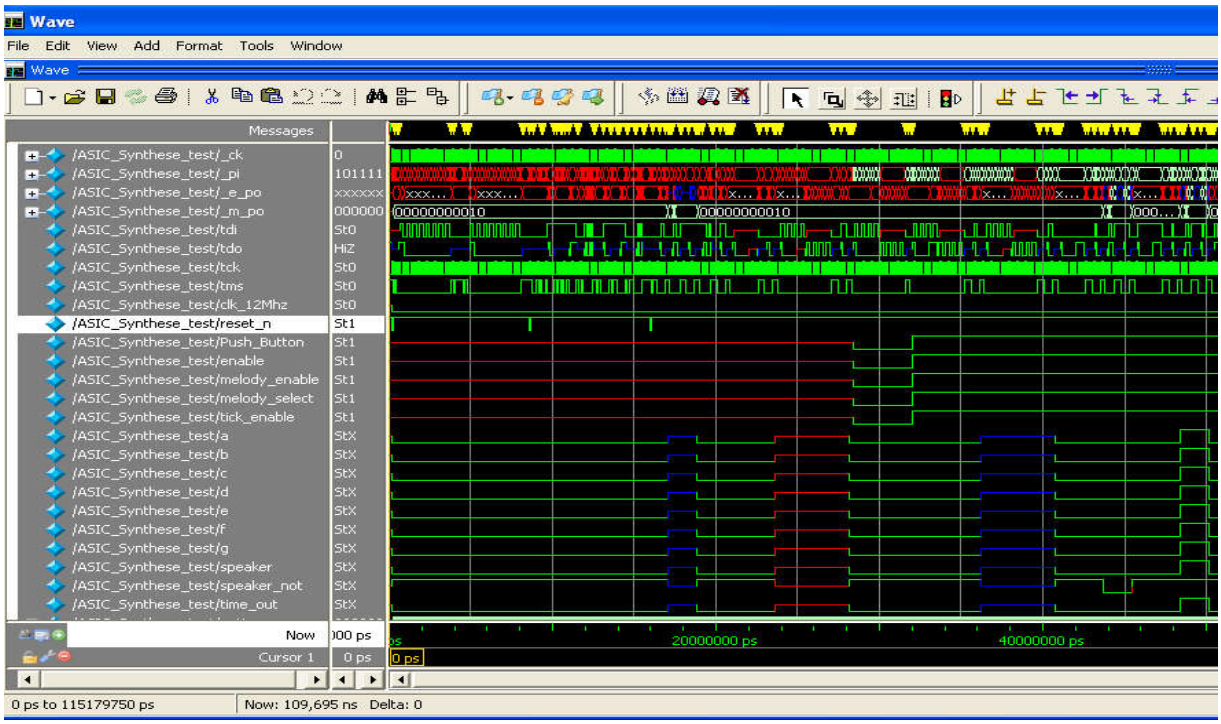
Automatische Tests, basierend auf den herausgeschriebenen funktionalen Test-Pattern, geben über zwei Aspekte Auskunft.
Werden diese Test-Pattern in einer Simulation am fertigen ASIC-Design verwendet, so sagt uns das Testergebnis, ob die implementierte Boundary Scan Architektur funktioniert und korrekt implementiert wurde.
Verwendet man diese Test-Pattern hingegen am gefertigten ASIC-Design, die Test-Pattern werden also in den ASIC tatsächlich eingefüttert und die Testergebnisse überwacht und mit Erwartungswerten abgeglichen, dann gibt uns dies Auskunft, ob Boundary Scan am gefertigten ASIC funktioniert und ob das ASIC-Design mit einer bestimmten Wahrscheinlichkeit korrekt gefertigt wurde, seiner ursprünglichen Funktion entspricht. Diese Wahrscheinlichkeit wird

bestimmt durch das Maß der Testabdeckung, die sich durch Verwendung der Test-Pattern ergibt. Test-Pattern können bei Verwendung nie eine Testabdeckung von 100 % erzielen, da es nicht möglich ist alle möglichen Fertigungsfehler festzustellen. Dem Maß der Testabdeckung entsprechend, verhält sich auch der Wert der Wahrscheinlichkeit, dass der ASIC korrekt gefertigt wurde. Die Vorgehensweise um das Maß der Testabdeckung zu erfahren, wird im darauffolgenden Kapitel behandelt.

An dieser Stelle wird nur der erste Typ von Automatischen Tests in einer Simulation ausgeführt.
In ModelSIM wird das Boundary Scan Design des ASICs mit entsprechenden Bibliotheken, die erstellte Verilog Test-Bench, welche die Test-Pattern beinhaltet, geladen, das Projekt kompiliert, ein Makro (Stimuli-Do-File) ausgeführt, um im Wesentlichen das System samt Boundary Scan vor der Simulation zu reseten sowie um die Simulationszeit festzulegen, und dann die Simulation der Test-Pattern gestartet.

Nach Beendigung der Simulation erscheint eine Ansicht aller Signalverläufe.
Das Wave-Fenster (siehe nächste Abbildung) zeigt Signale der Eingangs-/Ausgangsports des Designs bei der Simulation als Simulationsergebnis an. Wurden alle Test-Pattern erfolgreich simuliert und ergaben keine Fehler, so endet die Simulation mit der Bildschirmausgabe „TEST COMPLETED WITH NO ERRORS“.
Ist dieser Stand erreicht, ist sichergestellt, dass die implementierte Boundary Scan Architektur korrekt implementiert wurde und funktioniert.

Abbildung 5:
Wave-Fenster nach erfolgter Simulation der funktionalen Boundary Scan Design Test-Pattern in ModelSIM



5. Ermitteln der Testabdeckung durch die funktionalen BSD Test-Pattern

Die Testabdeckung gegeben durch die funktionalen BSD Test-Pattern, kann durch Simulation der Test-Pattern auf Basis eines Fehlermodells mittels des Programmes Synopsys TetraMAX ATPG errechnet werden.

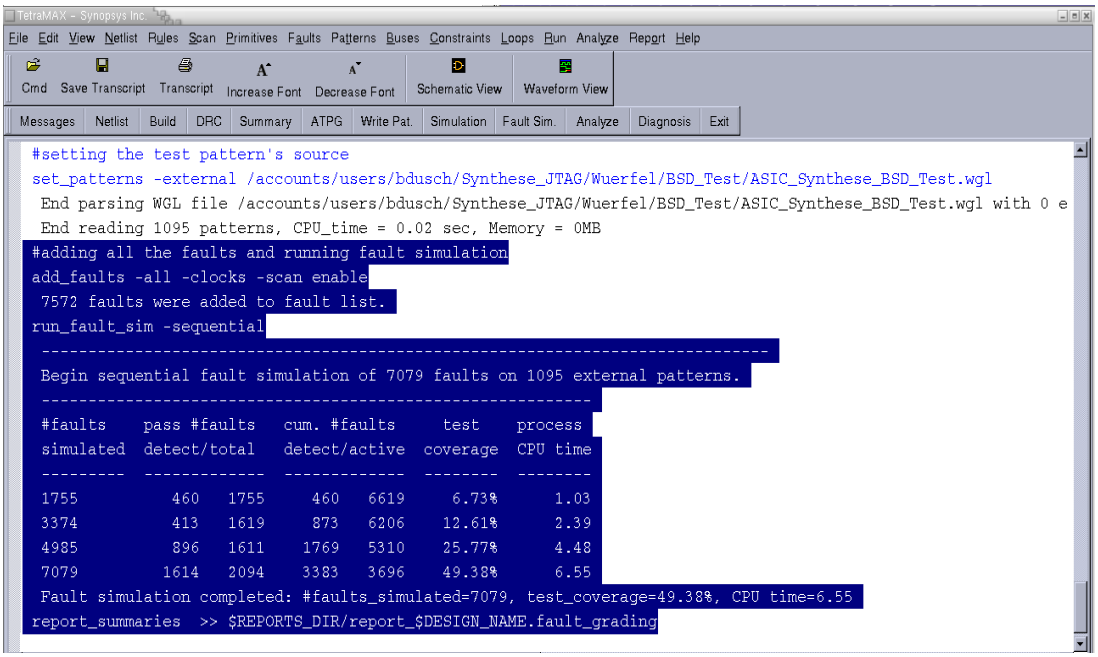
Diese Fehlersimulation wird wiederum skriptgesteuert durchgeführt.
Es werden Hardware-Fehler, die bei Fertigung des ASICs entstehen können, systematisch vom Programm automatisch zu einer Fehlerliste, zu einem

Fehlermodell, zusammengefügt, und durch Simulation von Test-Pattern ermittelt, welche der möglichen Fertigungsfehler gefunden werden können bzw. abgedeckt werden.

Die nächste Abbildung zeigt das Ergebnis der Testabdeckung. Die Testabdeckung von 49,38 % ist nicht ausreichend groß, da nur 3383 von 7079 Fehlern abgedeckt werden. In Anbetracht der Tatsache aber, dass nur funktionale BSD Test-Pattern verwendet wurden, ist das Ergebnis signifikant.

Dieses Programm kann, wie bereits erwähnt, im weiteren Sinne für das Herausschreiben von ATPG-Test-Pattern für ein Design verwendet werden.
Eine Optimierung der Testabdeckung wird im Rahmen der Erzeugung scan-basierender ATPG-Test-Pattern im Laufe meiner weiteren Arbeit am IAF erzielt werden.

Abbildung 6:
Ergebnis der Fehler-Simulation der funktionalen Boundary Scan Design Test-Pattern in TetraMAX



6. Emulation des ASIC-Designs zwecks Test der Debugging-Funktion

Die Funktionserweiterungen werden durch Emulation des fertigen Boundary Scan Designs auf einem FPGA-Emulations-Testboard nachgewiesen.

Auf dem Board befindet sich ein FPGA, Programmier-Schnittstellen sowie den konfigurierbaren I/Os des FPGAs zugeordnete Steckerleisten zur Kontaktierung der FPGA-I/Os. Über die Steckerleisten ist es somit möglich, I/Os des FPGAs, welche im Rahmen der Emulation Ports des ASICs nachbilden sollen, über die Steckkontakte zu erreichen, mit Signalen zu beaufschlagen oder per Messung zu überwachen.

Das FPGA-Emulations-Testboard verfügt desweiteren über I/Os mit angeschlossenen LEDs.

Zum Emulieren ist nun die spezielle Netzliste des Designs, herausgeschrieben durch Skript 4 der Boundary Scan Insertion, zu verwenden. Das ASIC-Design wird im FPGA nachgebildet, wobei Ports des FPGAs Ports des ASICs darstellen, nachbilden sollen. Im Programm Altera Quartus II werden hierfür vorher Pins des FPGAs den Ports des ASIC-Designs zugewiesen.
Jedem Eingangs-Port und jedem Ausgangs-Port des ASICs werden Pins des FPGAs zugewiesen. Die I/Os des FPGAs, welche über angeschlossene LEDs verfügen, werden als Ausgänge des ASICs verwendet, die LED zeigen damit Würfel-Ergebnisse an, wenn der ASIC arbeitet. Dies ist ideal für die Demonstrationszwecke der Debugging-Funktion am ASIC-Design.

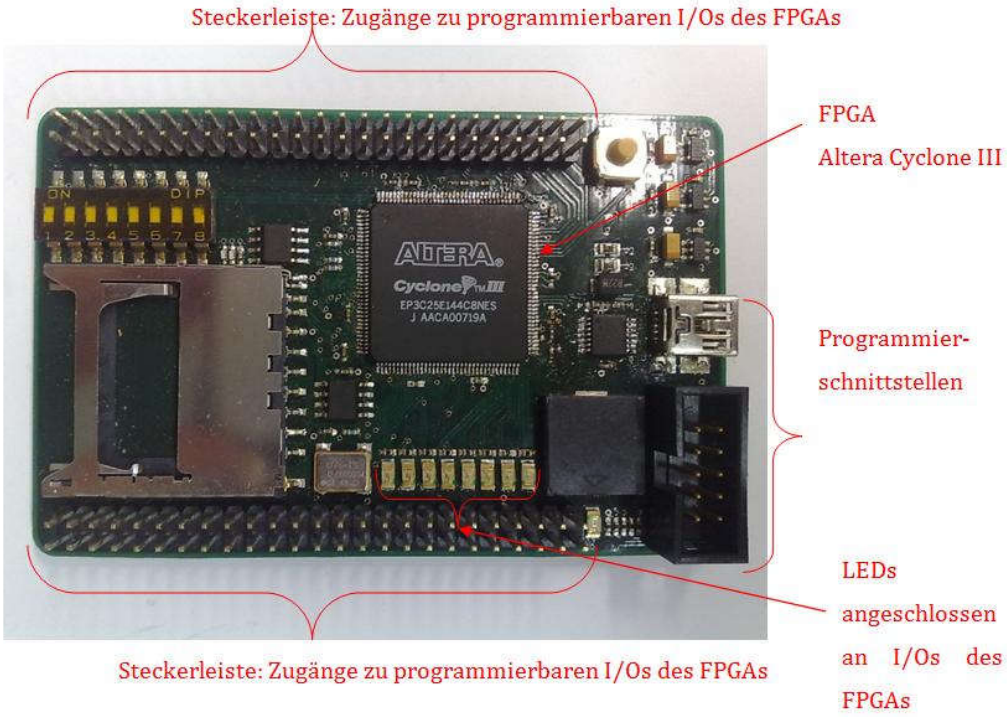


Abbildung 7: FPGA-Emulations-Testboard der Hochschule Offenburg

7. Debugging des ASIC-Designs: Einzeltaktbetrieb

7.1 Verwendete Hardware/Software

Im Rahmen von Recherchen hinsichtlich möglicher zu verwendender Software und Hardware, um ein Boundary Scan ASIC-Design anzusprechen, steuern und debuggen zu können, fiel die Wahl auf das XJTAG Development System.
Dieses beinhaltet diverse Programme zur Verwendung bei Boundary Scan Design sowie eine USB zu JTAG Schnittstelle.

Zur Demonstration der Debugging Funktion wird das Programm XJDeveloper verwendet und das auf einem FPGA-Emulations-Testboard emulierte ASIC-Design des „Rolling Dice“ über das Interface mit dem PC bzw. der Software verbunden.

7.2 Konfiguration des Programmes XJDeveloper / Einbindung des emulierten ASICs / Inbetriebnahme des ASICs

Für die Erkennung des ASIC-Designs ist eine Konfiguration des Programmes auszuführen, die spezifisch auf das ASIC-Design ausgerichtet ist.

Der genaue Ablauf der Konfigurierung soll in dieser Dokumentation nicht behandelt werden, es erfolgt lediglich ein grobes Erklären wichtiger Aspekte.

7.2.1 Laden der Board-Netzliste und der Boundary Scan Description Language (BSDL)-Datei

Das Programm XJDeveloper ist so aufgebaut, dass es einen ASIC nur dann hardware-mäßig erkennt, wenn dieser auf einem Board platziert bzw. aufgelötet ist. Deshalb wird eine Board-Netzliste für den ASIC geschrieben und der ASIC weiterhin bei Verwendung des Programmes XJDeveloper als Bestandteil eines Boards betrachtet.
Es ist eine spezielle Board-Netzliste im RINF-Format für das ASIC-Design und das im Kapitel der Boundary Scan Insertion generierte BSDL-File zu laden, damit das Programm XJDeveloper den ASIC grundsätzlich als „Board“ erkennt und Definitionen über dessen Testfähigkeiten erhält. Die manuell geschriebene Board-Netzliste und automatisch generierte BSDL-Datei werden per Programm-Menü eingelesen.

7.2.2 Konfigurieren der JTAG Chain

Eine JTAG Chain ist zu definieren, d.h. der Scan Chain der Core Logic vom Test-Daten-Eingang TDI zum Test-Daten-Ausgang TDO des ASICs zu beschreiben.

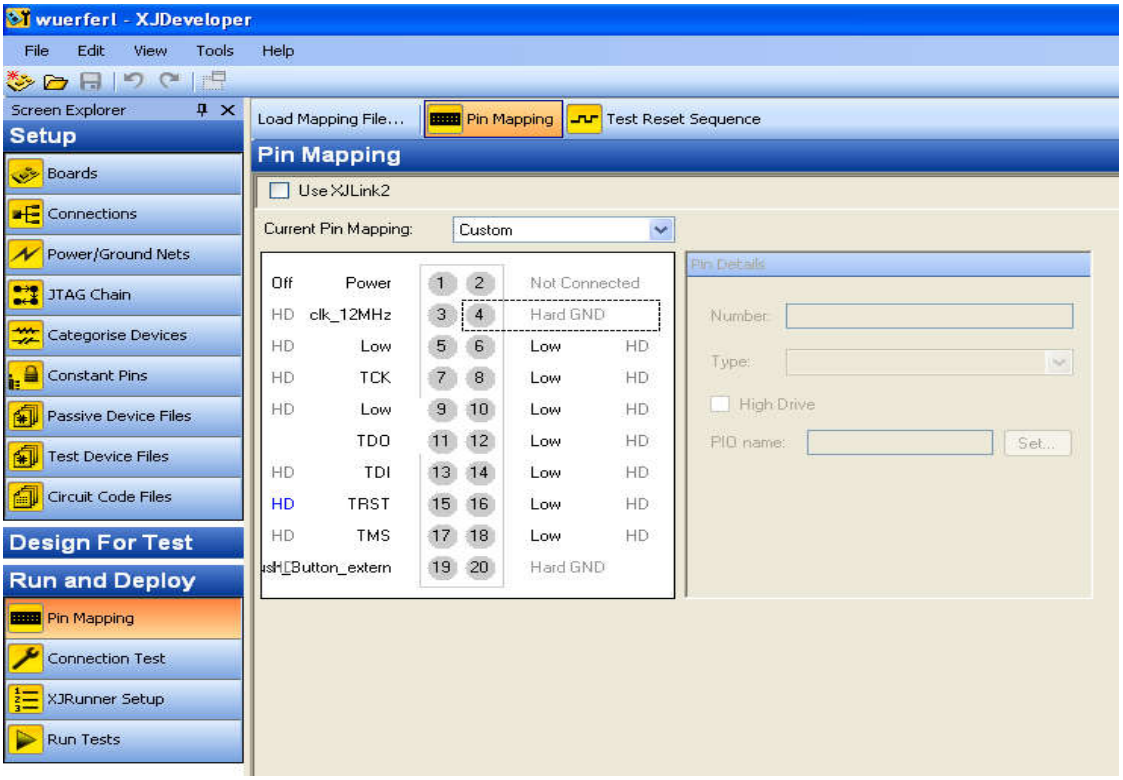
7.2.3 Pin Mapping der JTAG zu USB-Schnittstelle

Es ist ein „Pin Mapping“ der JTAG zu USB-Schnittstelle von XJTAG zu definieren, d.h. hauptsächlich festzulegen, welchen der 16 möglichen frei konfigurierbaren Anschluss-Pins der Schnittstelle die einzelnen Ports TDI, TDO, TCK und TRST des Test Access Ports (TAP) zuzuweisen sind.

Das Programm ermöglicht im weiteren Sinne über Befehle *SET PIO.XXX := 0* (oder *1*); Eingänge XXX des ASICs anzusteuern, wobei per XJTAG-Schnittstelle der jeweilige ASIC-Eingang mit dem PC, dem Programm XJDeveloper, verbunden werden muss. Diese PIO-Verbindungen, „Programmable I/Os“, müssen an dieser Stelle auch konfiguriert werden.

In diesem Fall werden für die Demonstration der Debugging-Funktion des ASIC-Designs der Reset *TRST* zum Ausführen des System-Resets und der Eingang *Push_Button_extern* (Push_Button-Port) des ASICs zum Aktivieren der Funktion „Würfeln“ des ASICs „Rolling Dice“ als ansteuerbare Eingänge eingerichtet.

Abbildung 8:
Pin Mapping der JTAG zu USB-Schnittstelle



7.3. Schreiben eines Test-Programmes zur Demonstration der Debugging-Funktion am emulierten ASIC-Design

Das Programm ist nun grundsätzlich konfiguriert und das emulierte ASIC-Design angeschlossen.
Es kann nun im Programm XJDeveloper in einer XJTAG-eigenen Programmiersprache programmiert, das Programm kompiliert und ausgeführt werden.
Im Rahmen dieser Arbeit wurde ein umfangreiches Programm geschrieben, welches die Funktionen des Rolling Dice im Einzeltaktbetrieb bedient, gleichzeitig Registerinhalte aller Flip Flops des Scan Chains der Core Logic des ASIC–Designs des „Rolling Dice“ ausliest und anzeigt.

In einem Programmierfenster (Test Editor) im Programm XJDeveloper zum Schreiben, Kompilieren und Ausführen eines Debugging-Programmes über Boundary Scan / JTAG wird ein Testprogramm (Test

Device File) geschrieben, das Ausführen des Testprogrammes erfolgt per einfaches Anwählen der Funktion „Run Tests“ des Programmes XJDeveloper.

Visualisiert wird das ausgeführte selbst geschriebene Programm je nach Programmierung in einem Textausgabefenster, die Debugging-Funktion über Boundary Scan wird damit verdeutlicht.

7.4 Ausführen der Debugging-Funktion

Das Debugging-Testprogramm detailliert zu beschreiben würde hier den Rahmen sprengen, weshalb in Auszügen kurz auf Ergebnisse eingegangen wird.

Die folgende Abbildung zeigt eine ausgegebene Bestätigung eines durchgeführten Resets zu Beginn des Testprogramms als Textausgabe an, ebenso werden alle Registerwerte der Scan Flip Flop des TDR auf dem Bildschirm binär angezeigt.
Da Boundary Scan Register (BSR) und Scan Chain der Core Logic in einem TDR zusammengefasst sind (STT, „Scan Through TAP"), können sowohl Zustände von Ein-/Ausgängen des ASICs als auch von Flip Flops des ASIC ausgelesen und angezeigt werden.

Dank einer erstellten Beschreibung des Scan Chains der Core Logic am Ende der Scan Insertion Synthese, können nun alle Bits des TDR beim Debuggen eindeutig den jeweiligen Flip Flops des ASIC-Designs zugeordnet werden.

Die untere Abbildung zeigt alle Zustände der Ein-/Ausgänge des konfigurierten emulierten ASICs mit beschalteten Eingängen und alle Zustände der Flip Flops der Core Logic des ASICs „Rolling Dice“ nach Reset

Der Zähler „Counter“ des ASIC-Designs des „Rolling Dice“ ist nach dem Reset auf 0.
Wird der ASIC durch das Debugging-Testprogramm über Boundary Scan gesteuert und im Einzeltakt betrieben, der ASIC in seiner Funktion gestartet, so zählt der Counter ca. alle 1000 Takte (f = 1 kHz) um eins hoch. Nach Einzeltakt 1017 bei Takt 1018 zählt der Counter um eins hoch, so wie dies von der Logik erwartet wird (siehe darauffolgende Darstellung).

Die Zustände aller Flip Flops der Core Logic des ASICs sowie der Ein-/Ausgänge des ASICs konnten bis zu diesem Takt Nummer 1018 beobachtet und analysiert werden.

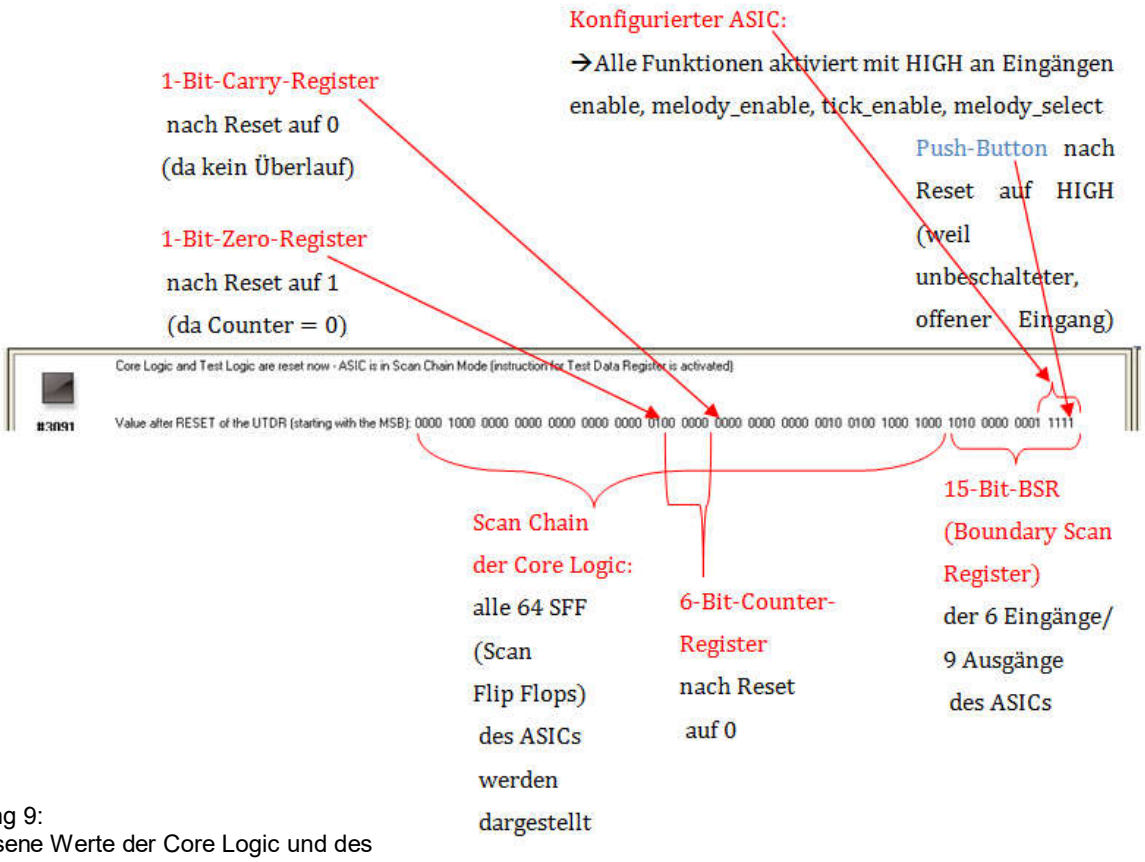


Abbildung 9:
ausgelesene Werte der Core Logic und des Boundary Scan Registers (BSR) des „Rolling Dice“ nach Reset der Testlogik

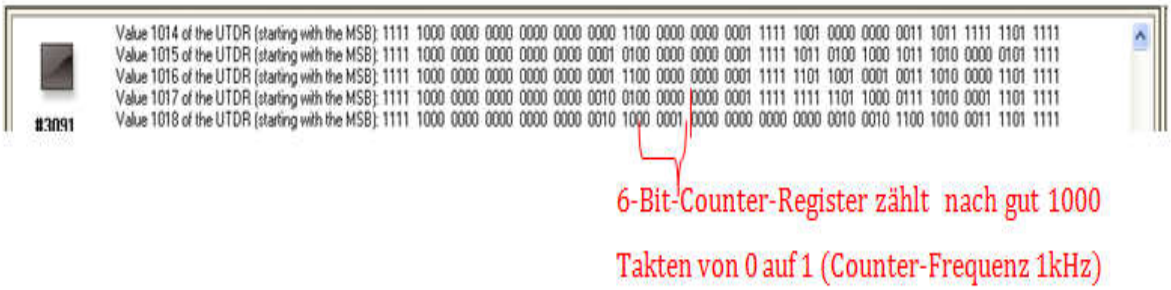


Abbildung 10: ausgelesene Werte der Core Logic und des Boundary Scan Registers (BSR) des „Rolling Dice“

8. Einführung von DFT-Strukturen: Abwägung von Nutzen gegenüber Aufwand

Anzahl Standardzellen des ASIC-Designs	mit DFT	ohne DFT
„Rolling Dice“	1170	400
„PDA V.2“	17000	16000

Tabelle 1: benötigte Standardzellen mit/ohne DFT je nach Größe eines ASIC-Designs

Die Einführung der DFT-Strukturen in das „kleine ASIC-Design des „Rolling Dice“ zeigt einen verhältnismäßig großen zusätzlichen schaltungstechnischen Aufwand, der bei der Umsetzung der Schaltung auf Ebene der Standardzellen zu erbringen ist. Das Maß der benötigten Standardzellen erhöht sich beinahe auf 300%. Bei Implementierung in das relativ „große“ ASIC-Design des „PDA V.2“ verringert sich der Mehraufwand relativ betrachtet, das Maß der benötigten Standardzellen erhöht sich auf weniger als 110%.

Sich nun festzulegen und zu sagen, dass Einführung nur in „große“ ASIC-Design lohnt, will ich mir nicht anmuten. Ob der je nach ASIC-Design nötige Mehraufwand sich lohnt einzugehen, muss im Einzelfall abgewogen werden. Hierbei sind die enormen Vorteile positiv ins Kalkül zu ziehen, die sich durch die Funktion „Automatische Tests“ und „Debugging“ des ASICs, ergeben.

9. Fazit und Ausblick

Die DFT-Struktur Boundary Scan mit integrierter DFT-Struktur Scan (zur Implementierung des Scan Chain der Core Logic) wurde erfolgreich in das einfache ASIC-Design des „Rolling Dice“, in das Frontend-ASIC-Design des „DQPSK“ und in das Prozessor-ASIC-Design des „PDA V.2“ eingeführt.
Am Design des „Rolling Dice“ konnten die durch die Einführung dieser Struktur zu erzielenden Funktionserweiterungen „Debugging-Funktion“ und „Automatische Tests“ vollständig nachgewiesen, grundlegende Vorgehensweisen erarbeitet und verifiziert sowie die Grundsteine gelegt werden für vertiefende Entwicklungsarbeiten bezüglich dieser neuen Funktionen von ASIC-Designs.
Das Design des PDA V.2 konnte im zeitlichen Rahmen bisher nicht überprüft werden, hier sind zukünftige Aufgaben noch auszuführen, zu denen das Verifizieren des ASIC-Designs mittels Manueller und Automatischer Tests gehört.

Allgemeine Entwicklungsarbeiten, die hinsichtlich der eingeführten DFT-Struktur Boundary Scan noch angegangen werden müssen, umfassen beispielsweise aus dem Bereich der Automatischen Tests die weitere Erforschung der Möglichkeiten, die Testabdeckung des Designs mittels zu kreierender scan-basierender ATPG-Test-Pattern zu erhöhen, dann das Entwickeln einer hard-/softwareseitigen Architektur, um gefertigte ASIC-Designs über den implementierten Boundary Scan mittels der Test-Pattern Automatischen Tests unterziehen zu können.
Im Bereich des Debugging ist spezifische Software zu erschaffen, um bei der Entwicklungsarbeit ASIC-Design über den eingeführten Boundary Scan möglicherweise über eine Benutzeroberfläche geeignet im Einzeltaktbetrieb betreiben zu können.

Am Ende dieser Dokumentation bleibt zu sagen, dass sich durch DFT-Strukturen vielseitige Möglichkeiten ergeben, weitere Entwicklungsarbeit zu betreiben, da dieses weite Feld noch längst nicht ausgereizt wurde.

10. Literaturverzeichnis

IEEE Standard Test Access Port and Boundary Scan Architecture 1149.1-2001. (27. März 2008).
Synopsys BSD Compiler User Guide. (Version D-2010.03-SP2, June 2010).
Synopsys BSD Compiler Reference Manual (Version D-2010.03-SP2, June 2010).
Synopsys DFT Compiler Scan User Guide. (Version C-2009.06, September 2009).
Synopsys TetraMAX® ATPG User Guide. (Version C-2009.06, June 2009).
Synopsys Library Compiler™ Methodology and Modeling Functionality in Technology Libraries User Guide . (Version C-2009.06, September 2009).

Synopsys Synthesis Commands. (Version D-2010.03, March 2010).
Synopsys DFT Overview. (Version A-2007.12, December 2007).