Leichtgewichtige Datenkompressions-Architektur für IEEE 1149.1-kompatible Testschnittstellen

Sebastian Huhn*† Stephan Eggersglüß*† Rolf Drechsler*†

*Arbeitsgruppe Rechnerarchitektur
Universität Bremen
28359 Bremen, Deutschland
{segg,drechsle}@informatik.uni-bremen.de

†Cyber-Physical Systems
DFKI GmbH
28359 Bremen, Deutschland
Sebastian.Huhn@dfki.de

***Zusammenfassung*—Diese Arbeit stellt eine dynamisch konfigurierbare Datenkompressions-Architektur für IEEE 1149.1-kompatible *Testschnittstellen* (TAPs) vor, welche in heutigen System-on-a-Chip (SoC)-Entwürfen vielfach verwendet werden. Diese Architektur ermöglicht eine Reduktion der Testdaten, ohne dabei das native Schnittstellenprotokoll zu beeinträchtigen. Des Weiteren werden keine zusätzlichen externen Anschlüsse benötigt und außerdem ist die Entwurfsvergrößerung vernachlässigbar. Dementsprechend eignet sich diese Technik für Board- und Feldtests, die typischerweise IEEE 1149.1 als Testzugriffsmechanismus (TAM) verwenden und generell starke Speicherbeschränkungen besitzen. Diese Beschränkungen besitzen insbesondere bei komplexen Entwürfen sowohl Einfluss auf die Diagnosefähigkeit als auch auf die maximale Testkomplexität. Zur Evaluation dieser Architektur wurden verschiedene Experimente sowohl für Zufallstestdaten als auch für voll-spezifizierte Testdaten industrieller Entwürfe durchgeführt. Diese Experimente zeigen eine nennenswerte Reduktion des *Testdatenvolumens* (TDVs) und zum Teil ebenfalls eine messbare Reduktion der Testzyklen, die für die Übertragung benötigt werden.**

I. Einleitung

Seit vielen Jahren nimmt die Komplexität von *Integrierten Schaltkreisen* (ICs) stetig zu, beispielsweise durch den immer stärker werdenden Systemcharakter. Hierdurch erhöht sich ebenfalls das Risiko von physikalischen Defekten, die während der Produktion entstanden sind. Aus diesem Grund sind einerseits Produktions- und Boardtests unerlässlich, andererseits erfordern die komplexen Anwendungsszenarien immer häufiger auch Feldtests in der realen Einsatzumgebung.

Hierbei beeinflusst das TDV direkt die Belegung von limitierten Speicherressourcen der Testgeräte, wodurch die Anzahl an Testsequenzen begrenzt ist, die simultan geladen werden können. Dies hat zur Folge, dass die Testkosten steigen, die typischerweise einen nennenswerten Anteil der Gesamtkosten einnehmen. Mehrere Forschungsarbeiten, beispielsweise [1]–[4], beschäftigen sich mit der Einbindung von spezieller Kompressionshardware in den IC-Entwurf. Bei den meisten Ansätzen wird jedoch der Entwurf wesentlich vergrößert und es werden spezifische Eigenschaften vorausgesetzt, beispielsweise eine hohe Anzahl an X-Werten innerhalb der Testdaten. Des Weiteren erfordern viele dieser Techniken zusätzliche externe Anschlüsse auf dem Toplevel, welche speziell für Untermodule schwer zu realisieren sind. Aus diesem Grund wird eine zentrale *Testschnittstelle* (TAP) benötigt, die einen späteren Datentransfer ermöglicht, welcher typischerweise seriell mit einer stark reduzierten Taktfrequenz erfolgt. Hierbei lassen sich die meisten Verfahren nicht auf Testdaten für die *Funktionale Verifikation* (FV) anwenden. Insbesondere innerhalb komplexer Entwürfe, die multiple Untermodule umfassen, stellt die FV Möglichkeiten bereit, um Startsequenzen oder die Kommunikation zwischen einzelnen Modulen zu testen.

In heutigen Entwürfen werden solche Testschnittstellen häufig eingebunden, um auf einzelne Untermodule, trotz der beschränkten Anzahl an möglichen IO-Pins auf dem Toplevel, gezielt zugreifen zu können. Aus diesem Grund bietet es sich an, diese Schnittstellen mit einem Kompressionsmechanismus zu kombinieren, solange sichergestellt werden kann, dass das zugrundeliegende Zugriffsprotokoll unberührt bleibt. Hierbei skaliert die Testdatenmenge jedoch häufig direkt mit der Komplexität des Entwurfes, welches mit den limitierten Speicherressourcen in Konflikt steht. Unserer Recherche zur Folge existiert momentan noch keine Kompressionstechnik, die einen Ansatz bietet, um dieser ansteigenden Testdatenmenge in diesen Anwendungsszenarien entgegenzuwirken.

Diese Arbeit stellt eine neue Kompressionsarchitektur vor, welche direkt in einen standardisierten *Testzugriffsmechanismus* (TAM) integriert wird. Hierdurch wird eine wesentliche TDV-Reduktion ermöglicht, ohne das logische Verhalten existierender Testsequenzen oder die native Protokollunterstützung zu beeinflussen. Wesentlich hierbei ist, dass diese Architektur nicht für die Anwendung von Testdaten, welche im Rahmen der Automatischen Testmustergenerierung (ATPG) erzeugt wurden, entworfen wurde. Primär dient sie zur Kompression von FV-Testdaten, welche im Rahmen des Board- bzw. Feldtestes zu übertragen sind. Diese Architektur verwendet ein Codewort-basiertes Verfahren, welches sich sowohl aus statischen als auch aus dynamisch konfigurierbaren Codeworten zusammensetzt, wodurch selbst heterogene Testdaten effektiv verarbeitet werden können. Außerdem ermöglicht diese Architektur ebenfalls eine Lauflängen-basierte Codierung, um homogene Sequenzen innerhalb von Testvektoren effektiv abbilden zu können. Ein weiterer Vorteil besteht darin, dass keine weiteren Anschlüsse benötigt werden und eine Anwendung auch auf bereits komprimierte Testdaten gewinnbringend, d.h. es wird eine weitere TDV-Reduktion erzielt, erfolgen kann.

Die Gliederung dieser Arbeit ist wie folgt: Kapitel II grenzt diese Arbeit gegenüber anderen existierenden Arbeiten ab. Außerdem werden in diesem Kapitel die grundsätzliche Idee und das erwartete Resultat beschrieben. Kapitel III legt die benötigten Grundlagen, bevor anschließend im Kapitel I die entwickelte Erweiterung der Testschnittstelle dargestellt wird. Nachfolgend skizziert Kapitel V das ebenfalls entwickelte Werkzug zur Vorverarbeitung existierender Testdaten, welches auch eine entsprechende Konfiguration der Testschnittstelle berechnet. Ausgewählte Experimente, die zur Evaluation dieser Kompressionsarchitektur durchgeführt wurden, werden im Kapitel VI aufgezeigt, bevor abschließend in diesem Kapitel ein Fazit gezogen und aktuelle Folgearbeiten skizziert werden.

II. Verwandte Arbeiten

Generell existieren im Kontext des ATPGs, dessen Datensätze oftmals eine große Anzahl von unspezifizierten Werten besitzen, unterschiedliche Ansätze zur TDV-Reduktion. In [5] hingegen wird ein leistungsstarker Kompressionsansatz vorgestellt, der keine unspezifizierten Werte voraussetzt. In

anderen Arbeiten, wie beispielsweise [1]–[4], [6], werden verschiedene Kompressionsverfahren vorgestellt, die statische und Lauflängen-basierte Kodierungen verwenden oder dynamische Ansätze mittels einer Huffman-Kodierung, eines *LZ77*-Algorithmus und eines sog. *Golomb-Codes* verfolgen. Generell benötigen gerade die komplexeren Verfahren einen großen Hardwaremehraufwand und sind nicht dafür geeignet, im TAP integriert zu werden.

Diese Arbeit fokussiert auf den seriellen Datentransfer zu einem *zu testenden Schaltkreis* (engl. *Circuit Under Test*, CuT) über eine zentrale IEEE 1149.1 Schnittstelle in ein ausgezeichnetes *Testdatenregister* (TDR). Dieses TDR soll die exklusive Schnittstelle zum CuT bilden, welcher die *Funktionale Logic* (FL) realisiert. Dementsprechend kann dieses TDR beispielsweise verschiedene Bitfelder umfassen, die Informationen über einzelne Speicheradressen, Kontrollsignale oder Instruktionen enthalten. Das Ziel dieser Arbeit ist die Entwicklung und Realisierung einer leichtgewichtigen Architektur zur Datenkompression innerhalb eines TAP-Controllers. Hierbei soll eine vollständige Protokollunterstützung umgesetzt werden, jedoch keine zusätzlichen Anschlüsse verwendet und im Vergleich zu einer IEEE 1149.1 Basisimplementierung nur ein geringer Hardwaremehraufwand verursacht werden. Diese Technik soll sowohl für ATPG als auch für FV Daten anwendbar sein und selbst bei bereits komprimierten, hoch entropischen Daten eine messbare Kompression erzielen, ohne dabei nachgeschaltete Dekompressionsverfahren zu beeinflussen.

III. Präliminarien

Der IEEE 1149.1 ist ein weit verbreiteter Standard, auch JTAG genannt, der eine Testschnittstelle beschreibt. In der ursprünglichen Form werden fünf Anschlüsse benötigt: Der Testtakt (TCK), der Testdateneingang (TDI), der Testdatenausgang (TDO), das Reset-Signal (TRST) sowie ein zusätzliches Testmode-Steuersignal (TMS). Der Controller implementiert einen Zustandsautomaten (engl. *Finite State Machine*, FSM), welcher sich in einen Instruktions- sowie Datenpfad aufteilen lässt und durch TMS gesteuert wird. Diese FSM ist in Abbildung 1 gezeigt. Es können somit zuerst Instruktionen geladen werden, wie beispielsweise *BYPASS*, und anschließend Daten seriell übertragen werden. Diese Schnittstelle zeichnet sich durch den geringen Hardwareaufwand, aber auch durch die Flexibilität aus, beispielsweise können durch eine Reihenschaltung von mehreren Einheiten auch mehrstufige Zugriffsmechanismen realisiert werden.

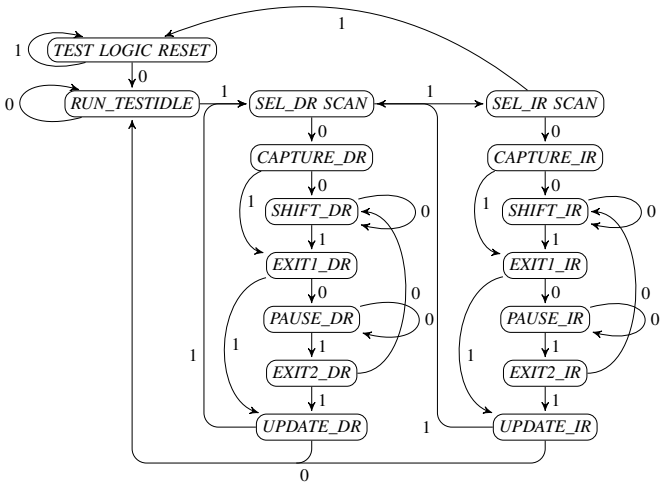

Abbildung 1: Zustandsautomat vom standardisierten IEEE 1149.1 Controller, TMS Wert an den Kanten

IV. Dekompressor-basierte Testschnittstelle

Zur Realisierung dieser Testschnittstellen muss ein Mechanismus zur Aktivierung des Kompressionsverfahrens, d.h. sowohl die Verarbeitung der komprimierten Daten als auch die dynamische Konfiguration selbst, umgesetzt werden. Zusätzlich muss eine entsprechende Code-basierte, dynamisch konfigurierbare Dekompressionseinheit (engl. *Dynamic Decompressing Unit*, DDU), welche die empfangenen Testdaten expandiert, entworfen und innerhalb des Controllers implementiert werden. Außerdem wird ein Werkzeug benötigt, mit dem existierende Testsequenzen für die Verwendung dieser Kompressionstechnik nachbearbeitet werden können, welches in Kapitel V kurz skizziert wird.

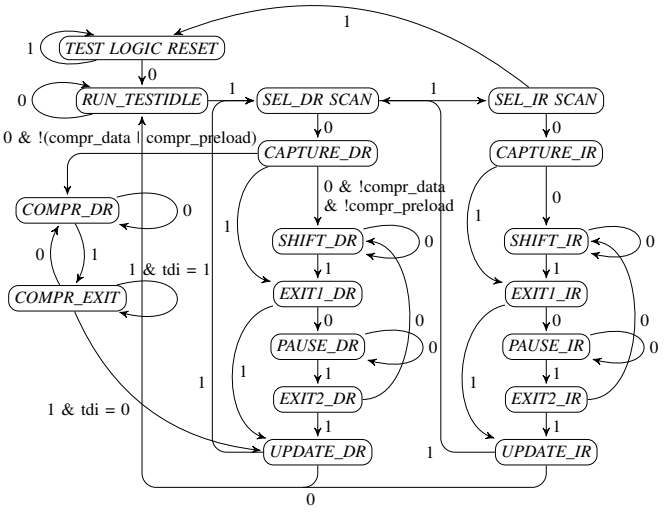

Abbildung 2: Zustandsautomat vom modifizierten IEEE 1149.1 Controller, TMS Wert an den Kanten

A. Erweiterung des TAP-Controllers

Zur Aktivierung bzw. erneuten Deaktivierung des Kompressionsverfahrens werden innerhalb des Controllers zwei zusätzliche Instruktionen *compr_data* und *compr_preload* implementiert. Des Weiteren werden im Datenbereich der FSM, die in Abbildung 2 gezeigt ist, zwei neue Zustände *compr_dr* und *compr_exit* eingefügt. Durch diese zwei zusätzlichen Zustände wird eine klare Trennung zwischen dem nativen Protokoll und der vorgestellten Erweiterung erzielt, sodass ebenfalls dezidierte Erweiterungen umgesetzt werden können. Ist weder die Instruktion *compr_data* noch *compr_preload* aktiv, verhält sich der Kontroller neutral, wodurch die native Protokollunterstützung sichergestellt wird. Im Fall, dass eine dieser beiden Instruktionen aktiv ist, finden die Zustandsübergange gemäß den Transitionen aus Abbildungen 2 statt.

Solange sich der Kontroller im Zustand *compr_dr* befindet, wird das TDI Signal als Teil des Komprimierten Datenwortes (engl. *Compressed Data Word*, CDWs) interpretiert und in einem ausgezeichneten *Kompressionsregister* (engl. *Compress Register*, CR) gespeichert. Ist die Übertragung mit dem nächsten Segment abgeschlossen, erfolgt ein Wechsel in den Zustand*compr_exit*. In diesem Zustand wird der Inhalt vom CR der DDU übergeben. Anschließend wird entweder ein weiteres CDW übertragen oder der reguläre Ablauf wird über den Zustand *update_dr* fortgesetzt. Neben dieser *compr*-Technik, wurde eine Lauflängen-basierte Erweiterung μ*-compr* umgesetzt, die eine n-fache Wiederholung des zuletzt übertragenen CDWs erlaubt, ohne das CDW erneut zu übertragen. Um dieses zu ermöglichen kann die FSM nach dem Zustandsübergang

CDW	UDW	Gewinn β	Dyn. konfigurierbar
$\varnothing$	CDW@$t-1$	-	✗
0	1	0	✗
1	00000000	7	✗
00	1111	2	✗
01	0101	2	✗
10	0110	2	✗
11	0	-1	✗
000	01010101	5	✓
001	1010	1	✓
010	0000	1	✓
011	10101010	5	✓
100	1000	1	✓
101	1001	1	✓
110	0001	1	✓
111	11111111	5	✓

Tabelle I: Exemplarische Abbildungsfunktion Ψ, $CS = 3$

von *compr_dr* in *compr_exit* im Zustand *compr_exit* für weitere n Taktzyklen verbleiben. Hierfür ist es notwendig im Zustand *compr_exit* bei aktiven TMS, ebenfalls das TDI-Signal auszuwerten. Analog hierzu werden diese beiden Zustände in der Konfigurationsphase bei aktiver *compr_preload* Instruktion verwendet. Hierbei wird bei jedem *compr_dr* $\rightarrow$ *compr_exit* Zyklus ein UDW übertragen, welches zur Konfiguration eines CDWs in der DDU dient.

B. Dynamisch konfigurierbare Dekompressionseinheit

Die Aufgabe der DDU Ψ besteht darin, ein CDW zu expandieren, d.h. es auf ein *Unkomprimiertes Datenwort* (UDW) abzubilden. Entsprechend realisiert Ψ die Funktion $\Psi(CDW^c) \rightarrow UDW^u$. Hierbei muss gelten $c \leq CS$, wobei Segmentgröße (engl. *Chunk Size*, CS) der maximalen Länge eines CDWs entspricht. Auch wenn CS frei wählbar ist, beeinflusst es direkt die zusätzlich benötigten Hardwareressourcen. In dieser Arbeit wird für CS 3 gewählt, welches einen guten Trade-off bietet und $\sum_{i=0}^{3} 2^i = 15$ unterschiedliche CDW ermöglicht:

- $\varnothing$, $'0'$, $'1'$, $'00'$, $'01'$, $'10'$, $'11'$
- $'000'$, $'001'$, $'010'$, $'100'$, $'101'$, $'110'$, $'111'$

In Tabelle I wird eine exemplarische Umsetzung von Ψ dargestellt, welche u.a. die statische Kodierung der CDW enthält und 2^{CS} Einträge umfassen, die dynamisch konfiguriert werden können. Zusätzlich ist jedem CDW ein sog. Gewinn β zugeordnet, der die eingesparten ($\beta > 0$) bzw. zusätzlich verursachten Kosten ($\beta < 0$) repräsentiert, wenn das spezifische CDW zur Kodierung des verknüpften UDWs verwendet wird. Bei $\varnothing$ handelt es sich hierbei um ein das leere CDW, welches für eine Wiederholung der letzten Übertragung im Rahmen der Lauflängencodierung reserviert ist.

Die Auswahl der Abbildungen ist entscheidend für die spätere Effektivität des Kompressionsverfahrens. Zuerst wurden diese Einträge vor der Synthese statisch bestimmt. Im Anschluss werden mit Hilfe einer heuristischen Methode entsprechende Konfigurationen für die DDU bestimmt, wodurch insbesondere die erzielbare Kompressionsrate für heterogene Testdaten erhöht wird. Die Länge des UDWs wird durch u bestimmt, welches konform zum verwendeten TDR gewählt werden muss. Hier wird $u = \{1, 4, 8\}$ angenommen, wobei der Sonderfall $u = 1$ benötigt wird, um sog. *Einzelbit-Injektionen* (engl. *Single Bit Injections*, SBIs) zu ermöglichen. Diese SBIs stellen die Vollständigkeit des Verfahrens sicher, d.h. jede Testdaten-Bitfolge b der Länge n kann durch eine Sequenz von CDWs repräsentiert werden. SBIs sollten möglichst vermieden werden, da sie starken Einfluss auf die erreichbare TDV-Reduktion haben und außerdem *künstlich* die Gesamtanzahl der notwendigen Testzyklen erhöhen.

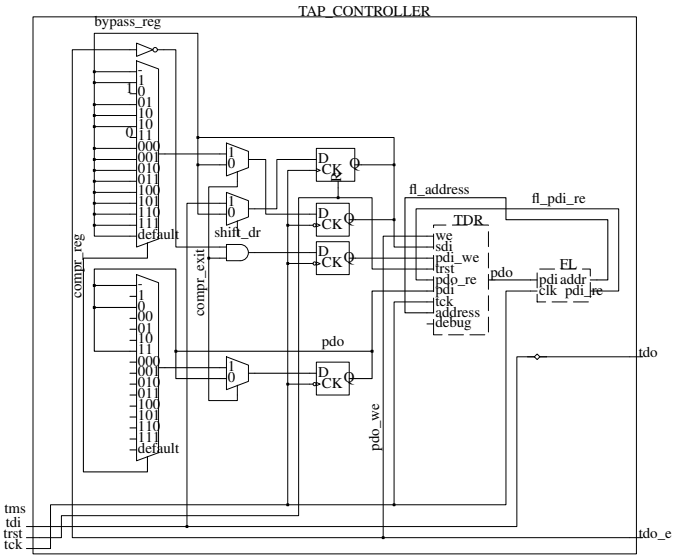

Abbildung 3: Modulsicht der Dekompressor-Integration (nur Datenfluss)

Die Abbildung 3 zeigt die prinzipielle Integration des Dekompressors. Das TDR bildet hierbei die Schnittstelle zwischen der FL und dem TAP. Jeweils ein *Multiplexer* (MUX) dient zur Realisierung der SBIs und ein weiterer MUX bindet die parallele 4- bzw. 8-Bit Schnittstelle ein, welche durch den Inhalt des CRs gesteuert werden. Zusätzlich steuern insgesamt drei weitere 1-Bit MUX den Datenfluss in Abhängigkeit des aktuellen Zustandes. Entsprechende Flip-Flops stellen eine Synchronisation bzgl. des Testtaktes und dem Reset-Signal sicher. Insbesondere in Anbetracht des geringen Hardwareaufwandes einer IEEE 1149.1 Standardimplementierung weist diese Erweiterung nur einen geringfügigen Hardwaremehraufwand auf.

V. Werkzeug zur Testvektor-Vorverarbeitung

Zur Verwendung der beschriebenen Kompressionsarchitektur müssen die zu übertragenden Testdaten einmalig vorverarbeitet werden, wofür ein spezialisiertes Werkzeug benötigt wird, das in diesem Kapitel skizziert wird.

Im Rahmen der Testdatenvorverarbeitung wird zuerst eine Konfiguration für die DDU erzeugt, mit der es den TAP-Controller vor Beginn der eigentlichen Übertragung zu konfigurieren gilt. Diese Konfiguration wird unter Berücksichtigung der statischen Einträge in der DDU Ψ und den Eigenschaften des Testvektors Ω heuristisch erzeugt. Nachfolgend eine prinzipielle Darstellung des Verfahrens:

1) Die Häufigkeiten der jeweiligen Segmente innerhalb von Ω werden ermittelt und in einer Datenstruktur Γ gespeichert. Dementsprechend wird ein (i, j) bestimmt, für das gilt: $\omega_i ... \omega_j \subseteq \Omega$ sowie $i \leq j \wedge Distanz(i, j) \in u$.
2) Das am häufigsten auftretende Segment $\alpha = \omega_i ... \omega_j$ wird in die DDU konfiguriert, sofern α nicht bereits enthalten ist, d.h. es muss gelten $\forall c \in \{\text{CDWs}\} : \Psi(c) \neq \alpha$.
3) Im Anschluss werden alle Einträge in Γ aktualisiert, sofern sie von α überdeckt wurden.
4) Schritt 2 wird wiederholt bis eine vollständige Überdeckung erzielt oder die Anzahl an maximal konfigurierbaren CDWs (hier: 2^{CS}) erreicht wurde.

Diese Konfiguration dient als Grundlage für den nachfolgenden Verarbeitungsschritt, in dem eine genauere Analyse von Ω durchgeführt und einzelne Segmente (UDWs) durch entsprechende Datenbit-reduzierende CDWs ersetzt werden.

Nr.	Testname	Laufzeit [s]		#Testzyklen				Größe [Bit]				TDV-Reduktion [%]	
		∅compr	∅μ-compr	leg	config	∅compr	∅μ-compr	leg	config	∅compr	∅μ-compr	compr	μ-compr
1	RTDR_512	1,98	2,16	4101	31	4831	4528	4096	28	3302	2999	18,7	26,1
2	RTDR_1024	4,57	4,86	8197	27	9354	8915	8192	24	6450	6011	21,0	26,3
3	RTDR_2048	9,25	9,68	16389	27	19119	17986	16384	24	13218	12085	19,2	26,1
5	p100k	2,71	2,95	5909	65	4718	4471	5902	44	3180	2933	45,4	49,6
6	p267k	8,45	9,12	17338	62	14122	13421	17332	41	9514	8814	44,9	48,9
7	p330k	8,73	9,43	18016	61	14659	13977	18012	40	9895	9214	44,8	48,6

Tabelle II: Benchmarking der Kompressionstechniken

1) Zuerst werden $(i_1, j_1), ..., (i_n, j_n)$ analog zum vorherigen Schritt (1) bestimmt, wobei zusätzlich noch $j_{n-1} < i_n$ gelten muss, d.h. die Segmente sind überschneidungsfrei.
2) Nachfolgend wird eine Menge initialer Ersetzungen $\widehat{\Delta}$ in der Art bestimmt, dass die (k, l) sukzessiv ersetzt werden, die in der DDU enthalten sind, d.h. $\exists c \in \{\text{CDWs}\}$: $\Psi(c) = \omega_k ... \omega_l$, und deren Gewinn β maximal ist.
3) Im Anschluss werden alle nicht überdeckten, jedoch zusammenhängenden Segmente $\omega_k ... \omega_l$ mit $Distanz(k, l) > 1$ in Ω betrachtet, um für diese ebenfalls (möglichst) weitere Ersetzungen durch CDWs zu bestimmen.
4) Abschließend werden verbleibende Segmente, die nicht ersetzt werden konnten, durch SBIs abgedeckt, um somit die notwendige vollständige Überdeckung zu erzielen.
5) Der Gewinn von allen Ersetzungen wird ermittelt und ggf. von Schritt (2) aus mit einer heuristisch permutierten Menge $\widehat{\Delta}$ erneut begonnen.

Für den Fall, dass die *μ-compr* Kompressionstechnik aktiv ist, werden identische aufeinanderfolgende CDWs zusammengefasst. Auf Basis der Ersetzungen $\widehat{\Delta}$ wird anschließend der komprimierte Testvektor $\widehat{\Omega}$ erstellt, bevor abschließend ein entsprechender IEEE 1149.1-kompatibler Datensatz wie folgt erzeugt wird:

- Aktivierung von *compr_preload*
- Zeilenweise Übertragung der DDU-Konfiguration
- Aktivierung von *compr_data*
- Übertragung der komprimierten Testdaten

Dieser resultierende Datensatz kann somit direkt zur Datenübertragung an einen CuT verwendet werden, sofern dieser eine IEEE 1149.1-kompatible Schnittstelle besitzt, welche die vorgestellte Kompressions-Architektur implementiert.

VI. Experimente & Zusammenfassung

In diesem Abschnitt werden die durchgeführten Experimente beschrieben. Als Basis wird eine Verilog-Implementierung eines IEEE 1149.1-konformen TAP-Controllers [7] verwendet, auf dessen Basis die beschriebene Kompressionsarchitektur umgesetzt ist. Zusätzlich findet diese Referenzimplementierung zur Validierung Verwendung, indem alle Experimente auf beiden Implementierungen des TAP-Controllers ausgeführt bzw. simuliert werden und anschließend die Simulationsergebnisse verglichen werden. Das Werkzeug zur Vorverarbeitung des Testvektors ist in *C++* implementiert. Die Experimente werden einheitlich auf einem *Intel Xenon E3-1240v2 3,4 GHz* mit *32 GB* RAM ausgeführt.

Die Tabelle II stellt ausgewählte Ergebnisse einiger durchgeführter Experimente dar, welche die folgenden Informationen spaltenweise enthält:

Laufzeit Benötigte Zeit für Vorverarbeitung
#Testzyklen Nr. Testzyklen für Testdatenübertragung
Größe Anzahl an Datenbit innerhalb des Testvektors
TDV-Reduktion Erzielte TDV-Reduktion in %

Die Spalte *Leg* verweist hierbei auf den regulären Datentransfer ohne Kompressionstechnik, *config* auf die reine Konfiguration der DDU und *compr* bzw. *μ-compr* auf die Ergebnisse bei Verwendung der jeweiligen Kompressionstechnik. Als Testdaten wurden hierbei zum einen Zufallstestdaten verwendet, die durch einen *Mersenne Twister*-Algorithmus erzeugt wurden. Zum anderen wurden voll-spezifizierte Testdaten industrieller Schaltkreise verwendet, die von NXP Semiconductors zur Verfügung gestellt wurden. Diese Zufallstestdaten bilden eine obere Schranke für die maximal erreichbare Kompression [8].

Diese Experimente zeigen, dass diese Technik ebenfalls für hoch-entropische Testdaten, beispielsweise bereits komprimierte Testdaten, eine nennenswerte Reduktion von bis zu 26,3% erzielt. Für voll-spezifizierte Testdaten industrieller Schaltkreise wird nicht nur eine TDV-Reduktion bis zu 49,6% erreicht, sondern die Anzahl an benötigten Testzyklen wird ebenfalls merklich reduziert.

In dieser Arbeit wird eine leichtgewichtige, dynamisch konfigurierbare Kompressionsarchitektur für IEEE 1149.1 konforme TAPs beschrieben, welche eine merkbare TDV-Reduktion ermöglicht, ohne dabei die native Protokollunterstützung einzuschränken. Aufbauende Folgearbeiten sollen, das in diesem Ansatz enthaltende, Werkzeug zur Vorverarbeitung der Testsequenz um formale, exakte Methoden ergänzen, wodurch die Effektivität erfahrungsgemäß weiter gesteigert werden kann. Zusätzlich wird eine Schnittstellenkomponente entwickelt, durch welche die DDU direkt an existierende Strukturen im FL-Block, wie beispielsweise Scan-Ketten, angebunden werden kann. Hiermit einhergehend werden die TDO-Daten ebenfalls in die Modellierung mit einbezogen, d.h. eine notwendige Resynchronisation durch das Werkezug erfolgt. Denkbar wäre ebenfalls, dass eine TDO-Kompression durch gezielte Wiederverwendung der DDU umgesetzt wird.

Literatur

[1] A. Jas and N. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs," in *Test Conference, 1998. Proceedings., International*, Oct 1998, pp. 458–464.

[2] V. Iyengar, K. Chakrabarty, and B. Murray, "Deterministic built-in pattern generation for sequential circuits," *Journal of Electronic Testing*, vol. 15, no. 1-2, pp. 97–114, 1999.

[3] F. Wolff and C. Papachristou, "Multiscan-based test compression and hardware decompression using LZ77," in *Test Conference, 2002. Proceedings. International*, 2002, pp. 331–339.

[4] A. Chandra and K. Chakrabarty, "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 20, no. 3, pp. 355–368, Mar 2001.

[5] A. Wurtenberger, C. Tautermann, and S. Hellebrand, "Data compression for multiple scan chains using dictionaries with corrections," in *Test Conference, 2004. Proceedings. ITC 2004. International*, Oct 2004, pp. 926–935.

[6] A. Jas, J. Ghosh-Dastidar, and N. Touba, "Scan vector compression/decompression using statistical coding," in *VLSI Test Symposium, 1999. Proceedings. 17th IEEE*, 1999, pp. 114–120.

[7] I. Mohor, "JTAG test access port (TAP)," 2009, http://opencores.org/project,jtag.

[8] K. Balakrishnan and N. Touba, "Relationship between entropy and test data compression," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 386–395, Feb 2007.