Verifikation Rekonfigurierbarer Scan-Netze

Baranowski, Rafal; Kochte, Michael A.; Wunderlich, Hans-Joachim

Proceedings of the 17. Workshop Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV'14) Böblingen, Germany, 10-12 March 2014

url: https://cuvillier.de/de/shop/publications/6629-mbmv-2014

Abstract: Rekonfigurierbare Scan-Netze, z. B. entsprechend IEEE Std. P1687 oder 1149.1-2013, ermöglichen den effizienten Zugriff auf On-Chip-Infrastruktur für Bringup, Debug, Post-Silicon-Validierung und Diagnose. Diese Scan-Netze sind oft hierarchisch und können komplexe strukturelle und funktionale Abhängigkeiten aufweisen. Bekannte Verfahren zur Verifikation von Scan-Ketten, basierend auf Simulation und struktureller Analyse, sind nicht geeignet, Korrektheitseigenschaften von komplexen Scan-Netzen zu verifizieren. Diese Arbeit stellt ein formales Modell für rekonfigurierbare Scan-Netze vor, welches die strukturellen und funktionalen Abhängigkeiten abbildet und anwendbar ist für Architekturen nach IEEE P1687. Das Modell dient als Grundlage für effizientes Bounded Model Checking von Eigenschaften, wie z. B. der Erreichbarkeit von Scan-Registern.

Preprint

Verifikation Rekonfigurierbarer Scan-Netze

Rafal Baranowski, Michael A. Kochte, Hans-Joachim Wunderlich
ITI, Stuttgart Universität, Pfaffenwaldring 47, D-70569, Stuttgart
E-Mail: {baranowski, kochte}@iti.uni-stuttgart.de, wu@informatik.uni-stuttgart.de

Zusammenfassung

Rekonfigurierbare Scan-Netze, z. B. entsprechend IEEE Std. P1687 oder 1149.1-2013, ermöglichen den effizienten Zugriff auf On-Chip-Infrastruktur für Bringup, Debug, Post-Silicon-Validierung und Diagnose. Diese Scan-Netze sind oft hierarchisch und können komplexe strukturelle und funktionale Abhängigkeiten aufweisen. Bekannte Verfahren zur Verifikation von Scan-Ketten, basierend auf Simulation und struktureller Analyse, sind nicht geeignet, Korrektheitseigenschaften von komplexen Scan-Netzen zu verifizieren.

Diese Arbeit stellt ein formales Modell für rekonfigurierbare Scan-Netze vor, welches die strukturellen und funktionalen Abhängigkeiten abbildet und anwendbar ist für Architekturen nach IEEE P1687. Das Modell dient als Grundlage für effizientes Bounded Model Checking von Eigenschaften, wie z. B. der Erreichbarkeit von Scan-Registern.

1 Einleitung

Eingebettete Instrumente stellen einen großen Anteil höchstintegrierter Schaltungen dar und werden für Bringup, Post-Silicon-Validierung, Debug und Diagnose benötigt. Diese Instrumente werden auch während des Betriebs im Feld verwendet, z. B. für Power-Up-Initialisierung, Systemüberwachung, Fehlertoleranz oder Reparatur [1, 2, 3]. Rekonfigurierbare Scan-Netze (RSNs) ermöglichen effizienten und skalierbaren Zugriff auf eingebettete Instrumente sowie Test- und Diagnosestrukturen. Der kürzlich vorgestellte IEEE Std. 1149.1-2013 (JTAG) erlaubt schon einfache RSNs, in denen einzelne Scan-Register (Instrumente) aus der Scan-Kette ausgeschlossen werden können [4]. Der Vorschlag IEEE P1687 (IJTAG) strebt die Standardisierung noch flexiblerer Scan-Architekturen an [1].

Abb. 1 stellt ein einfaches RSN dar. Die 1-Bit Scan-Register `S1` und `S2` steuern den Zugriff auf zwei längere Register `S3` und `S4`. Die Scan-In-Daten werden nur durch Register `S3` geschoben, falls zuvor sichergestellt wurde, dass `S1` = `S2` = 1. Die internen Steuersignale kommen aus den Schatten-Latches der Scan-Register `S1` und `S2`, bleiben also während des Schiebens stabil.

Für statische Scan-Architekturen ist die Überprüfung einfacher Entwurfsregeln (DRC) ausreichend, um die Korrektheit des Entwurfs zu gewährleisten [5]. Das Zeitverhalten kann mittels statischer Zeitanalyse (STA) bestimmt werden [6]. Die Funktion komplexerer Strukturen, z. B. entsprechend IEEE Std. 1500, kann durch Simulation validiert werden [7].

Für allgemeine RSNs stellt jedoch die Entwurfsverifikation ein viel schwierigeres Problem dar. Steuersignale für Scan-Register können kombinatorisch von Werten in anderen

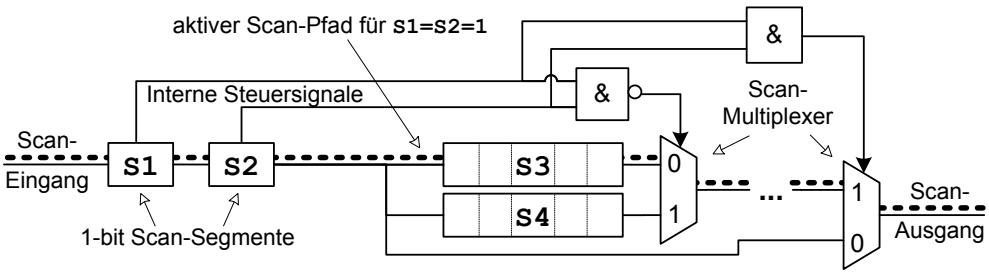

Abbildung 1: Beispiel eines rekonfigurierbaren Scan-Netzes

Scan-Registern erzeugt werden, welche wiederum von Registern in der gleichen oder unterschiedlichen Hierarchiestufe abhängen. Im IP/Core-basierten Entwurf können RSNs von Drittanbietern integriert werden, ohne dass deren Struktur oder Funktion komplett bekannt ist. Folglich können bestimmte Konfigurationen illegal oder widersprüchlich sein und zu Integrationsfehlern führen, z. B. den eingeschränkten Zugriff auf Scan-Register.

Ein Beispiel eines Entwurfsfehlers zeigt Abb. 1. Diese Struktur ist nach IEEE P1687 gültig. Der Zugriff auf das Scan-Register `S4` ist nicht möglich, da es keine Belegung der Scan-Register `S1` und `S2` gibt, so dass Daten durch `S4` geschoben werden können. Es besteht eine *kombinatorische Abhängigkeit*, die nicht erfüllt werden kann. Um diesen Entwurfsfehler zu finden, kann ein Algorithmus zur kombinatorischen Testmustererzeugung (ATPG) verwendet werden. Allerdings können solche Abhängigkeiten auch sequentiell sein: Eine Belegung, die einen Schiebepfad mit dem Zielregister definiert, kann u. U. nicht aus dem Startzustand des RSNs erreichbar sein. Aufgrund solcher *sequentiellen Abhängigkeiten* sind sowohl die Verifikation der Erreichbarkeit und die Berechnung von Zugriffsmustern für RSNs NP-harte Entscheidungsprobleme, ähnlich wie sequentielle Testmustererzeugung für Haftfehler. Sequentielle ATPG Systeme können mehrere Dutzend Taktzyklen berücksichtigen. Ein Zugriff in einem RSN kann aber Tausende von Takten erfordern, was aktuelle Algorithmen überfordert.

Diese Arbeit stellt eine Modellierung für allgemeine RSNs vor, die auch für komplexe Scan-Architekturen anwendbar ist. Das vorgeschlagene Modell ist geeignet für RSNs entsprechend IEEE Std. 1149.1-2013 und IEEE Std. P1687. Im Vergleich zu einem taktgenauen Modell wird in dem vorgeschlagenen Modell das zeitliche Verhalten abstrahiert. Somit wird die sequentielle Tiefe stark reduziert und eine effiziente Verifikation ermöglicht. Das vorgestellte Modell ist eine Verallgemeinerung bisheriger Modellierungsmethoden, die in [8] und [9] vorgestellt wurden, und wurde um die Behandlung von unbekannten oder unspezifizierten Werten erweitert. Das Modell wird beispielhaft zur Erreichbarkeitsanalyse von Scan-Registern mittels Bounded Model Checking verwendet. Die Methode erlaubt die Verifikation sehr großer RSNs und ist allgemeiner als Algorithmen, die nur für bestimmte Unterklassen von RSNs funktionieren, wie z. B. [10] für streng hierarchische Strukturen.

Der folgende Abschnitt stellt die hier betrachteten RSNs vor. Abschnitt 3 beschreibt das vorgeschlagene RSN-Modell mit Zeitabstraktion. Die darauf basierende Verifikation ist in Abschnitt 4 beschrieben. Experimentelle Ergebnisse werden in Abschnitt 5 diskutiert.

2 Terminologie

Rekonfigurierbare Scan-Netze (RSNs) bestehen aus Scan-Registern, Multiplexern und kombinatorischen Elementen. In der Regel wird auf RSNs durch einen JTAG-konformen Test Access Port (TAP) zugegriffen. RSNs können dann als Scan-Register mit *variabler Länge*

betrachtet werden. Der logische Zustand des RSNs bestimmt, welche der Register momentan zugreifbar sind. Der RSN-Zustand wird durch Überschreiben der Registerinhalte geändert.

Der Grundbaustein eines RSN ist ein *Scan-Segment* (siehe Abb. 2). Im einfachsten Fall ist ein Scan-Segment ein Schieberegister mit einem oder mehreren *Scan Flip-Flops* mit gemeinsamen Steuersignalen. Ein Scan-Segment unterstützt bis zu drei Operationen: Während der *Capture*-Operation wird das Register mit Daten von außerhalb des RSNs geladen, z. B. aus einem On-Chip-Instrument. Während der *Shift*-Operation werden Daten vom Scan-Eingang durch die Register-Bits zu dem Scan-Ausgang geschoben. Während der *Update*-Operation übernimmt ein optionales *Schatten*-Latch Daten aus dem Register. Das Steuersignal *select* eines Scan-Segments gibt an, ob das Segment für eine Capture-, Shift- oder Update-Operation aktiviert ist. Wie in JTAG-Testdaten-Registern bleibt das Schatten-Latch während des Schiebens stabil. Ein Scan-Segment mit einem Schatten-Latch kann für die bidirektionale Kommunikation mit einem On-Chip-Instrument genutzt werden. Optionale Elemente eines Scan-Segments sind in Abb. 2a) gestrichelt dargestellt.

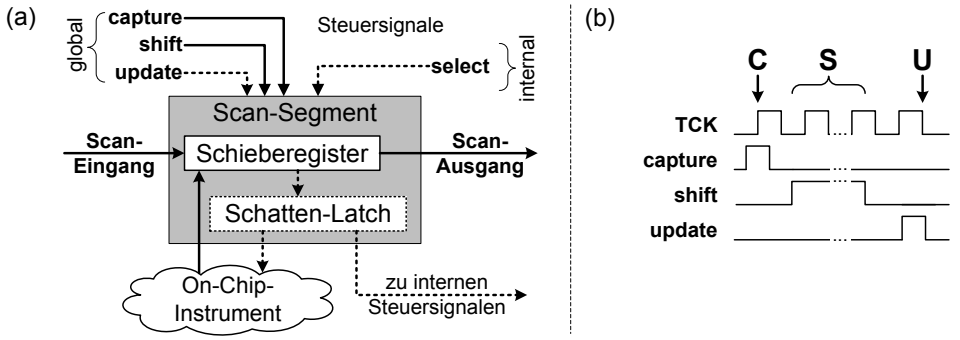

Abbildung 2: (a) Scan-Segment; (b) Capture-, Shift-, Update-Operation (CSU)

Scan-Multiplexer im RSN sind Multiplexer, die den Pfad, durch den Daten geschoben werden, kontrollieren. In Abb. 1 erlauben die zwei Scan-Multiplexer die Scan-Segmente `S3` und `S4` zu umgehen. Das Steuersignal *address* eines Scan-Multiplexers gibt an, welcher Eingang des Multiplexers ausgewählt ist.

Die Steuersignale von Scan-Segmenten und Multiplexern hängen vom Zustand des RSNs ab: Sie können von beliebigen kombinatorischen Logikelementen getrieben werden, welche wiederum von Schatten-Latches von Scan-Segmenten oder von externen Steuereingängen abhängen. In Abb. 1 werden alle Steuersignale von den Schatten-Latches der Scan-Segmente `S1` oder `S2` getrieben.

RSNs haben einen *primären Scan-Eingang* und einen *primären Scan-Ausgang*, sowie globale Steuersignale für die Capture-, Shift- und Update-Operationen. Weitere externe Steuersignale können interne Signale entweder direkt oder durch kombinatorische Logik treiben.

Zwei Scan-Segmente sind *direkt verbunden*, falls ihr Scan-Ausgang und Scan-Eingang durch ein Signal oder einen Multiplexer verbunden sind. Ein *Scan-Pfad* ist eine azyklische Folge von direkt verbundenen Scan-Segmenten, vom primären Scan-Eingang bis zum primären Scan-Ausgang des RSNs. Ein Scan-Pfad ist *aktiv* genau dann, wenn (gdw.) die Select-Signale der Segmente auf dem Pfad aktiv sind, und alle Multiplexer auf dem Pfad die Eingänge auswählen, die auf dem aktiven Pfad liegen. Die Select-Signale aller Segmente, die nicht zum aktiven Pfad gehören, bleiben inaktiv, um Datenverlust zu verhindern. In Abb. 1

läuft der aktive Scan-Pfad über $\texttt{S1}, \texttt{S2}, \texttt{S3}$, falls $\texttt{S1} = \texttt{S2} = 1$.

Eine *Scan-Konfiguration* ist der Zustand aller sequentiellen Elemente und der externen Steuersignale. Eine Scan-Konfiguration ist *gültig* gdw. (i) es einen aktiven Scan-Pfad gibt und (ii) nur Scan-Segmente, die zum aktiven Pfad gehören, ausgewählt sind (select=1). Somit wird sichergestellt, dass Eingabedaten zu den Zielsegmenten und Ausgabedaten korrekt zum primären Scan-Ausgang geschoben werden. Alle Segmente, die nicht zum aktiven Pfad gehören, bleiben stabil.

Ein Zugriff im RSN ist eine atomare Operation mit drei Schritten: *Capture, Shift und Update* (CSU, Abb. 2b). Während Capture registrieren die Segmente auf dem aktiven Pfad neue Daten. Diese Daten werden während der Shift-Phase herausgeschoben und neue Daten werden in die Segmente geschoben. Während Update werden die Daten in Registern auf dem aktiven Pfad im Schatten-Latch übernommen. Lese- und Schreibzugriffe erfordern, dass die Zielregister auf dem aktiven Pfad liegen.

3 RSN-Modellierung auf CSU-Granularität

Die RSN-Modellierung erfasst die Struktur und Funktionalität des RSNs und kann leicht aus einer strukturellen Beschreibung (Gate-, RT- oder höhere Ebene, z. B. Instrument Connectivity Language ICL/P1687) abgeleitet werden. Das Modell erlaubt eine 3-wertige Modellierung der Zustände und Signale mit den Symbolen $\{0, 1, X\}$ nach Kleene's 3-wertiger Logik.

Definition 3.1 *Das CSU-genaue RSN-Modell (Capture-shift-update-Accurate Model, CAM)* $\mathcal{M} = \{S, I, C, c_0, \texttt{Active}\}$ *besteht aus Zustandselementen* S*, externen Steuereingängen* I*, der Menge der Scan-Konfigurationen* $C \subseteq \{0, 1, X\}^{|S \cup I|}$*, der Startkonfiguration* $c_0 \in C$ *und einem Prädikat* $\texttt{Active}$*. Jedes Zustandselement* $s \in S$ *entspricht einem 1-Bit Scan-Register im RSN. Eine Konfiguration* $c \in C$ *bestimmt den Zustand aller Elemente in* S *und der externen Eingänge* I*.* c *kann auch als Funktion interpretiert werden:* $c : S \cup I \to \{0, 1, X\}$*, die jedem* $e \in S \cup I$ *einen Zustand* $c(e)$ *zuordnet. Das Prädikat* $\texttt{Active} : C \times S \to \{0, 1, X\}$ *weist jedem Element* $s \in S$ *in Konfiguration* $c \in C$ *einen Wert* $\texttt{Active}(c, s)$ *zu:*

$$\texttt{Active}(c, s) := \begin{cases} 0 & \textit{falls } \texttt{Select}(c, s) = 0, \\ 1 & \textit{falls } (\texttt{Select}(c, s) = 1) \textit{ und } c \textit{ ist gültig}, \\ X & \textit{sonst.} \end{cases}$$

$\texttt{Select}(c, s)$ *ist der Zustand des* select*-Signals des Scan-Segments* $s \in S$ *in Konfiguration* $c \in C$*. Element* s *ist Teil des aktiven Scan-Pfads gdw.* $\texttt{Active}(c, s) = 1$*.*

3.1 Gültige Scan-Konfigurationen

Um die Prädikate zu erzeugen, wird mittels der Booleschen Funktion $V : C \to \{0, 1\}$ zwischen gültigen und ungültigen Scan-Konfigurationen unterschieden (s. Abschnitt 2). V ist wahr gdw. eine Konfiguration gültig ist, also ein wohldefinierter Scan-Pfad existiert. V wird iterativ als Konjunktion $V(c) = \bigwedge_{s \in S} v(c, s)$ aufgebaut, wobei $v(c, s)$ genau dann wahr ist, wenn die lokale Scan-Konfiguration des Segments s in $c \in C$ gültig ist:

Für ein Segment s mit einem Vorgänger $p \in \texttt{pred}(s)$ und einem Nachfolger $n \in \texttt{succ}(s)$ (Abb. 3a), müssen p und n ausgewählt sein, wenn s ausgewählt ist, damit Scan-Daten nicht verlorengehen: $v(s) := (\mathit{select}(s) = 1) \Rightarrow [(\mathit{select}(p) = 1) \wedge (\mathit{select}(n) = 1)]$.

Für Segment s mit einem Vorgänger p und mehreren Nachfolgern (Abb. 3b) gilt, dass *genau ein* Nachfolger von s ausgewählt sein muss, wenn s ausgewählt ist. Mit

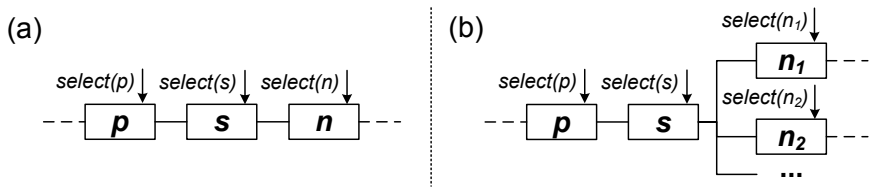

Abbildung 3: (a) Serielle und (b) verzweigende Scan-Strukturen

$A := (select(s) = 1) \Rightarrow \bigvee_{n \in \mathtt{succ}(s)} (select(n) = 1)$ wird sichergestellt, dass *mindestens ein* Nachfolger von s ausgewählt ist. Mit $B := \forall_{n_k, n_l \in \mathtt{succ}(s), n_k \neq n_l} : [(select(n_k) = 1) \Rightarrow (select(n_l) \neq 1)]$ gilt, dass *höchstens* ein Nachfolger von s ausgewählt sein kann. Mit (A) und (B) ergibt sich $v(s) := [(select(s) = 1) \Rightarrow (select(p) = 1)] \wedge (A) \wedge (B)$. Somit wird an Verzweigungen des Pfads sichergestellt, dass nur ein Nachfolger ausgewählt ist.

Entsprechend lassen sich die Funktionen für gültige Konfigurationen für Segmente mit einem Nachfolger und mehreren Vorgängern, und für Segmente mit mehreren Nachfolgern und mehreren Vorgängern ableiten (s. [8]).

Mit der oben eingeführten Funktion V ergibt sich für das Prädikat `Active` :

$$\mathtt{Active}(c, s) := \begin{cases} 0 & \text{falls } \mathtt{Select}(c, s) = 0, \\ 1 & \text{falls } (\mathtt{Select}(c, s) = 1) \wedge V(c), \\ X & \text{sonst.} \end{cases} \tag{1}$$

Die *Select*-Funktionen werden aus den Eingangskegeln der Kontrollsignale im strukturellen RSN-Modell erstellt.

Die Modellierung erfordert, dass der aktive Scan-Pfad nur Scan-Segmente, Multiplexer, Puffer und Inverter enthält, damit Scan-Daten während des Schiebens nicht verändert werden. Dies entspricht auch IEEE P1687. Enthält das RSN komplexere Logik, z. B. Test-Dekompressionsstrukturen, werden diese als Black-Box behandelt und in gültigen Scan-Konfigurationen vom aktiven Pfad ausgeschlossen.

3.2 Übergangsrelation des RSN-Modells

Die Übergangsrelation des CSU-genauen RSN-Modells (CAM) modelliert die Auswirkung einer CSU-Operation, nämlich eine Änderung der Zustände der Scan-Segmente auf dem aktiven Pfad.

Definition 3.2 *Die Übergangsrelation des CAMs* $\mathcal{M} = \{S, I, C, c_0, \mathtt{Active}\}$ *ist die Menge* $T \subseteq C \times C$ *mit allen Paaren von Scan-Konfigurationen* $(c_1 \in C, c_2 \in C)$*, für die gilt:* c_2 *ist durch eine CSU-Operation von* c_1 *erreichbar. Ihre charakteristische Funktion ist:*

$$T(c_1, c_2) := \bigwedge_{s \in S} \big[[(\mathtt{Active}(c_1, s) = 0) \Rightarrow (c_2(s) = c_1(s))] \wedge [(\mathtt{Active}(c_1, s) = X) \Rightarrow (c_2(s) = X)]\big].$$

Die Übergangsrelation definiert die Bedingung für Zustandsänderungen im RSN: Für Elemente s, die nicht zum aktiven Pfad gehören, ändert sich der Zustand nicht zwischen aufeinander folgenden Konfigurationen c_1 und c_2. Außerdem wird angenommen, dass der Zustand von s in c_2 unbekannt ist, wenn die Konfiguration c_1 ungültig ist, oder wenn unbekannt ist, ob s zum aktiven Scan-Pfad gehört $(\mathtt{Active}(c_1, s) = X)$. Der Zustand von s kann sich also nur ändern, wenn s in einer gültigen Konfiguration c_1 ausgewählt ist, d. h. $\mathtt{Active}(c_1, s) = 1$.

4 Formale Verifikation

Die folgenden Abschnitte beschreiben die Verwendung des Modells in der formalen Verifikation und seine Einschränkungen. Mittels Bounded Model Checking wird die Erreichbarkeit im RSN überprüft. Schließlich werden *robuste* RSNs definiert und eine Methode vorgestellt, mit der die Robustheit bewiesen werden kann.

4.1 Verifikation basierend auf CSU-genauer Modellierung (CAM)

Das CAM kann als *abstrakte* FSM verstanden werden, die den Zustand des RSNs exakt abbildet, aber das Zeitverhalten abstrahiert. Ein Übergang im CAM entspricht einer kompletten CSU-Operation, also mehreren Takten im RSN. Abb. 4 zeigt ein Beispiel, wo die k Takte einer CSU-Operation zu einem Übergang im CAM zusammengefasst werden.

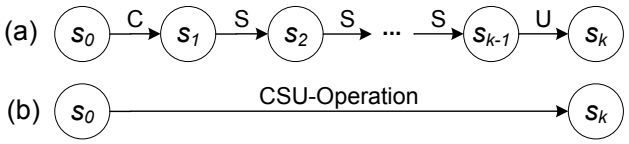

Abbildung 4: Zustandsübergänge während einer CSU-Operation (a) taktgenau; (b) im CSU-genauen Modell (CAM)

Die CSU-genaue Modellierung ist wohldefiniert: Eigenschaften, die im CAM gelten, sind auch für das taktgenaue RSN gültig, falls die internen Steuersignale (z. B. Multiplexer-Adresse) während der Capture- und Shift-Phase stabil sind. Für Signale, die im RSN generiert werden, gilt dies, da sie von Schatten-Latches getrieben werden. Für externe Signale muss dies sowieso im System sichergestellt werden, da sich sonst der aktive Scan-Pfad während des Schiebens ändern und Daten verloren gehen können.

Die CSU-genaue Modellierung ist pessimistisch: Nach Def. 3.2 ist nach einer ungültigen Scan-Konfiguration der Zustand der Scan-Segmente undefiniert, selbst wenn im taktgenauen Modell Segmente einen definierten Zustand besitzen können. So kann mit dem CAM ein Gegenbeispiel für eine Eigenschaft erzeugt werden, die im taktgenauen Modell stets gilt. Sind ungültige Konfigurationen nicht vom Startzustand aus erreichbar (s. Abschnitt 4.3), dann sind im CAM widerlegte Eigenschaften auch im taktgenauen Modell ungültig.

4.2 Bounded Model Checking

In diesem Abschnitt wird die Verwendung des CAM in Bounded Model Checking gezeigt (BMC, [11]): Sei P eine Eigenschaft in temporaler Logik, und T die Übergangsrelation des CAMs eines RSNs. Im CSU-genauen BMC wird nach einer Zustandsfolge (Gegenbeispiel) mit gegebener maximaler Länge gesucht, die P widerlegt. Dabei entspricht die Länge der Anzahl an CSU-Operationen. P ist widerlegt, wenn für eine gewisse Zahl $n \in \mathbb{N}^+$ an Schritten die folgende Formel erfüllt ist: $\varphi^n := I \wedge \left[\bigwedge_{i=0}^{n-1} T_i\right] \wedge (\neg P^n)$, ($I$: charakteristische Funktion der Startzustände von $\mathcal{M}$, T_i: charakteristische Funktion der Übergangsrelation T im i-ten Schritt, P^n: Eigenschaft P über n Schritte). Diese Formel kann in konjunktive Normalform (CNF) überführt und ihre Erfüllbarkeit mit einem SAT-Solver untersucht werden.

Mit BMC kann z. B. die Erreichbarkeit von Scan-Segmenten verifiziert werden. Dazu muss ein aktiver Scan-Pfad vom primären Scan-Eingang über das Segment bis zum Scan-Ausgang existieren. Für ein CAM $\mathcal{M} = \{S, I, C, c_0, \texttt{Active}\}$ ist der Beweis der Erreichbarkeit des Segments $s \in S$ äquivalent mit einem Gegenbeispiel für folgende LTL Formel: $\boldsymbol{G}\,[\texttt{Active}(s) \neq 1]$.

Der Beweis der Erreichbarkeit von s in max. n CSU-Operationen entspricht also einer erfüllenden Belegung für die Formel:

$$\texttt{Check}(s, c_0, n) := \mathbf{1}(c_0) \wedge \left[\bigwedge_{i=1\ldots n} T(c_{i-1}, c_i)\right] \wedge \left[\bigvee_{i=0\ldots n} [\texttt{Active}(c_i, s) = 1]\right],$$

wobei $\mathbf{1}(c_0)$ die charakteristische Funktion der Startkonfiguration $c_0 \in C$ ist. Die Erfüllbarkeit dieser Formel wird für eine steigende Zahl von CSU-Operationen ($n = 1, 2, \ldots$) untersucht, bis eine erfüllende Belegung gefunden wurde oder eine vorgegebene maximale Zahl an Schritten erreicht ist.

4.3 Verifikation der Robustheit

Ein RSN ist *robust*, wenn alle vom Startzustand aus erreichbaren Scan-Konfigurationen gültig sind, d. h. die ausgewählten Scan-Segmente bilden stets einen aktiven Scan-Pfad unabhängig von den Eingangsdaten. Formal ausgedrückt ist ein RSN robust gdw. die LTL Eigenschaft $\boldsymbol{G}\, V$ im CAM gültig ist, also die Boolesche Funktion V (s. Abschnitt 3.1) stets wahr ist.

Für robuste RSNs ist die Verifikation basierend auf dem CSU-genauen Modell *vollständig*: Eine Eigenschaft, die im RSN gilt, ist auch im CAM gültig, d. h. die Abstraktion auf atomare CSU-Operationen verursacht keine falschen Gegenbeispiele, da alle ausgewählten Segmente immer zum aktiven Scan-Pfad gehören (s. Abschnitt 4.1). Entsprechend kann der Zustand eines ausgewählten Segments uneingeschränkt von einer CSU-Operation geändert werden. Die CAM-Übergangsrelation (Def. 3.2) modelliert also exakt die Auswirkung einer CSU-Operation in einem taktgenauen Modell. Da *alle erreichbaren* Konfigurationen gültig sind, kann die Funktion V, welche die Gültigkeit ausdrückt, entfernt und das CAM stark vereinfacht werden: $\texttt{Active}(\texttt{c}, \texttt{s}) := \texttt{Select}(\texttt{c}, \texttt{s})$ für alle $c \in C$ und $s \in S$.

Die Robustheit ($\boldsymbol{G}\, V$) eines RSNs kann mit jedem unbeschränkten LTL Model-Checking-Verfahren auf dem CAM erfolgen. Hier wird eine effiziente SAT-basierte induktive Methode [12] verwendet. Ein RSN mit CAM $\mathcal{M} = \{S, I, C, c_0, \texttt{Active}\}$ und Übergangsrelation T ist robust gdw.:

1. V gilt im Startzustand von $\mathcal{M}$, d. h. $V(c_0)$ ist wahr, und
2. V ist eine Invariante der Übergangsrelation T, d. h. $\forall_{(c_1,c_2)\in T}[V(c_1) \Rightarrow V(c_2)]$.

Gelten beide Bedingungen, ist das RSN robust, da die Startkonfiguration gültig ist und jeder Übergang im CAM die Gültigkeit bewahrt.

Beide Bedingungen können als Erfüllbarkeitsproblem formuliert und mit einem SAT-Solver überprüft werden. $\mathcal{M}$ ist robust, wenn die folgenden Formeln nicht erfüllbar sind:

$$\mathbf{1}(c_0) \wedge \neg V(c_0), \tag{2}$$

$$V(c_i) \wedge T(c_i, c_{i+1}) \wedge \neg V(c_{i+1}), \tag{3}$$

wobei $\mathbf{1}(c_0)$ die charakteristische Funktion der Startkonfiguration ist.

Die erste Anforderung ist notwendig, die zweite hinreichend. Insbesondere gibt es robuste RSNs, in denen die Funktion V keine Invariante der Übergangsrelation ist. Dies gilt, wenn alle gültigen Konfigurationen, die zu ungültigen führen, nicht von der Startkonfiguration aus erreichbar sind. Die hier vorgeschlagene effiziente Methode kann also pessimistisch ein robustes RSN als nicht robust einstufen. Der Vorteil ist allerdings, dass das unbeschränkte LTL-Verifkationsproblem auf eine einfache Boolesche SAT-Instanz abgebildet wird.

5 Evaluierung

Die Skalierbarkeit der CSU-genauen Modellierung wird in BMC-basierten Verifikationsexperimenten evaluiert. Die Experimente werden auf einem Intel Xeon Prozessor mit 3,33 GHz durchgeführt.

5.1 Benchmark-RSNs

Zwei hierarchische Scan-Architekturen werden betrachtet: eine Implementierung mit Segment Insertion Bits (SIB), wie in [13], und eine spezielle Architektur, die zum effizienten Zugriff zwei Betriebsmodi unterstützt (MUX). Die Benchmark-RSNs basieren auf den ITC'02 Schaltungen [14] und werden in [8] ausführlich beschrieben.

Aus Platzgründen werden nur die vier größten Benchmark-RSNs für die zwei Architekturen betrachtet. Die ausgewählten Schaltungen beinhalten bis zu 3 Hierarchiestufen und 5 000 bis 100 000 Scan-Zellen, die in 160 bis 1 200 Scan-Segmente gruppiert sind.

5.2 Verifikation der Erreichbarkeit

Die Erreichbarkeit der Scan-Segmente wird mittels des BMC-Verfahrens aus Abschnitt 4.2 verifiziert. Tabelle 1 stellt den Verifikationsaufwand dar. Spalte "Zugriffslänge" gibt die durchschnittliche und maximale Anzahl der CSU-Operationen (BMC-Schritte) an, die zum Zugriff auf ein Scan-Segment benötigt werden. Unter "Klauseln" wird die maximale Anzahl von Klauseln in der erfüllbaren SAT-Instanz (für den letzten BMC-Schritt) angeführt. t_{solve}^{max} ist die maximale Lösungszeit für ein Scan-Segment, und t_{total} ist die Gesamtverifikationszeit.

Tabelle 1: Verifikation der Erreichbarkeit

Benchmark		Zugriffslänge	Klauseln	Konflikte	t_{solve}^{max}	t_{total}
Design	Architektur	avg / max	max	avg / max	[s]	[s]
g1023	MUX	3,6 / 5	32 917	2,0 / 27	0,03	2,6
	SIB	2,3 / 3	16 826	0 / 0	0,02	1,2
p22810	MUX	3,9 / 7	150 787	5,6 / 88	0,15	36,5
	SIB	2,5 / 4	77 376	0 / 0	0,05	17,2
p34392	MUX	4,4 / 7	67 367	9,1 / 107	0,07	8,1
	SIB	2,7 / 4	33 028	0 / 0	0,03	3,5
p93791	MUX	4,1 / 7	322 891	6,5 / 223	0,57	187,1
	SIB	2,5 / 4	172 082	0 / 0	0,14	94,5

Obgleich die Anzahl von Klauseln in den SAT-Instanzen auf bis zu 323 000 steigt, beträgt die maximale Verifikationszeit für ein Scan-Segment im größten RSN (p93791) höchstens 0,6 Sek. und ca. 0,15 Sek. durchschnittlich. Die Gesamtverifikationszeit vom größten RSN mit über 1 200 Scan-Segmenten beträgt ca. 3 Min.

Bei der Verifikation von SIB-basierten RSNs muss der SAT-Solver kein Backtracking durchführen. Im Gegensatz dazu weisen die MUX-basierten RSNs komplexere sequenzielle Abhängigkeiten auf, die Konflikte bei der SAT-Suche verursachen. Die durchschnittliche und max. Anzahl von Backtracking-Vorgängen ist in Tabelle 1 unter "Konflikte" angegeben.

5.3 Verifikation der Robustheit

Die Robustheit der Benchmark-Schaltungen wird mittels der SAT-basierten induktiven Methode geprüft (s. Abschnitt 4.3). Es wurde erfolgreich bewiesen, dass alle Benchmark-RSNs robust sind, d. h. die Scan-Konfiguration bleibt in diesen Schaltungen für beliebige Eingangssequenzen gültig. Maximal werden 84 Sek. für die Verifikation benötigt.

5.4 Verifikation fehlerhafter RSNs

Im Folgenden werden mittels der CSU-genauen BMC-Methode (s. Abschnitt 4.2) fehlerhafte RSNs verifiziert. Hierzu werden die MUX-basierte Benchmark-Schaltungen aus Abschnitt 5.1 mit folgenden Entwurfsfehlern versehen:

- *Pfad-Bug*: Die Nachfolger von zwei zufällig gewählten Scan-Segmenten oder Multiplexern werden vertauscht.
- *Steuer-Bug*: Zwei *address*-Steuersignale von zufällig gewählten Multiplexern werden vertauscht.
- *Mux-Bug*: Die Scan-Eingänge von einem zufällig gewählten Scan-Multiplexer werden vertauscht.

Ein Scan-Segment wird als unerreichbar klassifiziert, wenn innerhalb von 30 CSU-Operationen kein Zugriff darauf möglich ist (die maximale Anzahl von BMC-Schritten beträgt also 30). Diese Schranke ist signifikant größer als die maximale Zugriffslänge von 7 Schritten in den fehlerfreien Benchmarks (s. Tabelle 1). Um die Unerreichbarkeit im unbeschränkten Sinne zu verifizieren, z. B. für sicherheitsrelevante Eigenschaften, müssen symbolische oder interpolationsbasierte Model-Checking-Verfahren benutzt werden, z. B. [15].

Aus Platzgründen werden nur drei MUX-basierte RSNs (g1023, p22810, und p93791) untersucht. Für jede Art von Entwurfsfehlern und für jedes Benchmark-RSN wurden 100 zufällige Mutationen untersucht. Tabelle 2 zeigt die Verifikationsergebnisse: Spalte "Bugs gefunden" gibt die Rate der fehlerhaften RSNs an, in denen der Fehler detektiert wurde (d. h. innerhalb von 30 CSU-Operationen ist mindestens ein Scan-Segment nicht erreichbar). Spalte "Unerreichbar" nennt den durchschnittlichen Anteil von unerreichbaren Scan-Segmenten in einem fehlerhaften RSN. t^{avg}_{total} und t^{max}_{total} geben die durchschnittliche bzw. maximale Gesamtverifikationszeit für ein RSN.

Tabelle 2: Verifikation der Erreichbarkeit in fehlerhaften RSNs

Benchmark	Entwurfsfehler	Bugs gefunden	Unerreichbar	t^{avg}_{total} [s]	t^{max}_{total} [s]
g1023-MUX	Pfad-Bug	100%	13,3%	2,8	15
	Steuer-Bug	5%	1,9%	1,8	16
	Mux-Bug	100%	20,0%	4,6	18
p22810-MUX	Pfad-Bug	100%	7,6%	29,3	251
	Steuer-Bug	2%	1,0%	21,7	267
	Mux-Bug	100%	8,0%	39,0	363
p93791-MUX	Pfad-Bug	100%	7,9%	139,4	152
	Steuer-Bug	1%	1,0%	109,5	1301
	Mux-Bug	100%	5,9%	172,1	1859

Die durchschnittliche Verifikationszeit der fehlerhaften Benchmarks ist ähnlich wie bei den fehlerfreien Schaltungen (s. Tabelle 1). Im schlimmsten Fall beträgt die Verifikationszeit eines fehlerhaften RSNs 31 Min. Die *Pfad-Bugs* und *Mux-Bugs* werden immer gefunden (jedes fehlerhafte RSN beinhaltet mindestens ein unerreichbares Scan-Segment), wobei bis zu 99% der *Steuer-Bugs* keinen Einfluss auf die Erreichbarkeit von Segmenten haben. Es ist also oft möglich, eine Zugriffssequenz zu allen Scan-Segmenten zu finden, selbst wenn die Steuersignale von zwei zufällig gewählten Scan-Multiplexer vertauscht sind.

Entwurfsfehler führen oft dazu, dass ein ansonsten robustes RSN in eine ungültige Scan-Konfiguration versetzt werden kann. Deshalb können viele Entwurfsfehler effizient durch

die Verifikation der Robustheit detektiert werden. Mittels des Ansatzes aus Abschnitt 4.3 konnte bewiesen werden, dass *alle* untersuchten Bugs die Robustheit der Benchmark-RSNs verletzen. Im schlechtesten Falle werden nur 2 Min. für die Widerlegung der Robustheit eines fehlerhaften RSNs benötigt. Die dabei erzeugten Gegenbeispiele können zur Lokalisierung der Entwurfsfehler benutzt werden.

6 Zusammenfassung

Die Komplexität von rekonfigurierbaren Scan-Netzen und ihr Einsatz im IP-basierten Entwurf erfordern neue Werkzeuge für die Entwurfsverifikation. Diese Arbeit stellt eine Modellierung vor, die durch temporale Abstraktion eine hohe Skalierbarkeit von bestehenden Model-Checking-Verfahren gewährleistet. Die Modellierung unterstützt unbekannte Werte und ist für unterschiedliche konfigurierbare Scan-Architekturen verwendbar. Komplexe Eigenschaften von Scan-Netzen können damit effizient verifiziert werden.

Danksagung: Diese Arbeit wurde durch die Deutsche Forschungsgesellschaft (DFG) mit den Projekten WU 245/13-1 (RMBIST) und WU 245/11-1 (OASIS) gefördert.

Literatur

[1] N. Stollon, *On-Chip Instrumentation: Design and Debug for Systems on Chip.* Springer US, 2011.

[2] E. Larsson and K. Sibin, "Fault management in an IEEE P1687 (IJTAG) environment," in *IEEE Int'l Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2012, p. 7.

[3] J. Rearick and A. Volz, "A Case Study of Using IEEE P1687 (IJTAG) for High-Speed Serial I/O Characterization and Testing," in *Proc. IEEE International Test Conference (ITC)*, 2006, paper 10.2.

[4] "IEEE Standard for Test Access Port and Boundary-Scan Architecture 1149.1-2013," IEEE Computer Society, 2013.

[5] E. Eichelberger and T. Williams, "A logic design structure for LSI testability," in *Proc. Design Automation Conf. (DAC)*, 1977, pp. 462–468.

[6] J. Remmers, M. Villalba, and R. Fisette, "Hierarchical DFT Methodology - A Case Study," in *Proc. IEEE Int'l Test Conf. (ITC)*, 2004, pp. 847–856.

[7] A. Benso, S. Di Carlo *et al.*, "IEEE Standard 1500 Compliance Verification for Embedded Cores," *IEEE Trans. VLSI Syst.*, vol. 16, no. 4, pp. 397–407, 2008.

[8] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Modeling, Verification and Pattern Generation for Reconfigurable Scan Networks," in *Proc. IEEE Int'l Test Conf. (ITC)*, 2012, paper 8.2.

[9] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Scan Pattern Retargeting and Merging with Reduced Access Time," in *Proc. IEEE European Test Symposium (ETS)*, 2013.

[10] F. Ghani Zadegan, U. Ingelsson *et al.*, "Access Time Analysis for IEEE P1687," *IEEE Trans. Computers*, vol. 61, no. 10, pp. 1459–1472, October 2012.

[11] A. Biere, A. Cimatti *et al.*, "Bounded Model Checking," *Advances in Computers*, vol. 58, pp. 117–148, 2003.

[12] M. Sheeran, S. Singh, and G. Stålmarck, "Checking Safety Properties Using Induction and a SAT-Solver," in *Formal Methods in Computer-Aided Design.* LNCS, Springer, 2000, vol. 1954, pp. 127–144.

[13] F. Zadegan, U. Ingelsson *et al.*, "Design Automation for IEEE P1687," in *Proc. Design, Automation Test in Europe Conference (DATE)*, 2011, pp. 1412–1417.

[14] E. Marinissen, V. Iyengar, and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SOCs," in *Proc. IEEE Int'l Test Conf. (ITC)*, 2002, pp. 519–528.

[15] K. McMillan, "Interpolation and SAT-Based Model Checking," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2725, pp. 1–13.