VE-VIDES-Beitrag der OneSpin GmbH / Siemens EDA

FKZ: 16ME0248
Sachbericht 1 (Kurzfassung), Autor Jörg Bormann, Siemens EDA (SEDA), November 2024

Übersicht

Die VE-VIDES-Arbeiten wurden von der OneSpin Solutions GmbH beantragt und begonnen. OneSpin Solutions GmbH beabsichtigte mit der Beteiligung an VE-VIDES

- die Erarbeitung von Technologieführerschaft bei der Verifikation der Vertrauenswürdigkeit digitaler Schaltungen (AP 3.5 - Szenariobasierte Verifikation)
- die Erforschung eines weiteren wichtigen Einsatzfalls für das SystemC-Frontend (AP 3.3 - FW/HW-Coverifikation)

Kurz nach Start von VE-VIDES wurde OneSpin von Siemens EDA übernommen. Siemens EDA ersetzte danach OneSpin im VE-VIDES-Konsortium. Das führte zu einer Anpassung der Prioritäten in VE-VIDES.

Siemens EDA bekräftigte das Ziel der Technologieführerschaft bei der Verifikation von Vertrauenswürdigkeit. Der entsprechende VE-VIDES-Prototyp wurde nach Projektende zum Werkzeug Questa Verify Trust™ weiterentwickelt und wird inzwischen erfolgreich aktiv vertrieben.

Der Wert der Forschungsresultate über FW/HW-Coverifikation besteht für Siemens EDA in der Grundlagenforschung in Richtung auf hierarchische Verifikation von Embedded Systems und den Einsatz formaler Verifikation dabei. Durch diese Grundlagenforschung kann SEDA auf Kundenfragen nach Verbesserungen bei Entwurf und Verifikation von Embedded Systems mit interessanten innovativen Vorschlägen antworten. Solche Fragen werden im Zusammenhang mit den immer größeren Embedded Systems erwartet, die in Zukunft entwickelt werden. SEDA kann dabei sogar mit dem Prototypen argumentieren, der in VE-VIDES erforscht wurde und über den ein Patent angemeldet wurde (Publikationsnummer WO 2024/114920 A1), das derzeit beim Europäischen Patentamt geprüft wird.

Highlights der Forschung zur Verifikation der Vertrauenswürdigkeit

(AP 3.5, Szenariobasierte Verifikation) Ausgangspunkt der Forschungen waren verschiedene formale Verifikationsverfahren im OneSpin-Portfolio, die Schwachstellen der Vertrauenswürdigkeit aufdecken können, aber zum Beginn von VE-VIDES über verschiedene Verifikations-Apps und -Prototypen verteilt waren. Ziel der Forschungen war

1. Die Bedeutung der Verfahrens-Resultate für die Vertrauenswürdigkeit herauszuarbeiten
2. Leistungsfähigkeit, Analysemöglichkeiten und Vielfalt dieser Verfahren zu erweitern
3. Die Verfahren in einem Framework für einen Verifikationsprozess zu integrieren
4. Das Framework so zu gestalten, dass es leicht erweitert werden kann

Diese Ziele wurden wie folgt erreicht:

1. Die Bedeutung der Verfahrens-Resultate wird durch Verweis auf den Schwachstellen-Katalog cwe.mitre.org (Common Weaknesses Enumeration) nachgewiesen.
2. Neue Verifikationsverfahren:
    a. Security-Verifikation von Lock-Bits und der davon geschützten Register

b. Security-Verifikation von Schaltungen mit Zustandselementen ohne Reset

Existierende Verfahren aus dem OneSpin-Portfolio (teilweise optimiert):

c. Identifikation von Pfaden, über die Assets verfälscht oder verraten werden könnten, zur Vorbeugung gegen Angriffe auf Confidentiality oder Integrity
d. Deadlock-Identifikation zur Vorbeugung gegen Denial-of-Service-Angriffe
e. Identifikation von Triggern zur Aktivierung der Payload von Trojanern
f. Identifikation unbenutzten Codes zur Verbeugung gegen Seitenkanalangriffe

3. Die Verifikationsverfahren liefern Verdachtsfälle, die vom Benutzer weiter untersucht werden müssen. Zur Unterstützung dieses Verifikationsflows wurde ein GUI entwickelt, in dem diese Verdachtsfälle priorisiert, sortiert, analysiert und dokumentiert ausgeschlossen werden können.
4. Alle Verfahren wurden in das Framwork integriert.

Highlights der Forschung zur FW/HW-Coverifikation

(AP 3.3) Wir schlagen eine neue Methodik vor, bei der ein schmaler Low-Level-FW-Treiber integraler Bestandteil einer hauptsächlich durch ein HW-Modul implementierten Funktionalität wird. Treiber und HW-Modul werden als Einheit („Embedded Module“) angesehen und verifiziert. Spezifiziert wird das API des Embedded Modules und nicht mehr das Signalinterface des HW-Moduls.

Diese Methodik erlaubt die Übertragung der Vorteile der hierarchischen Verifikation reiner SW- oder reiner HW-Systeme auf embedded Systems. Auf der Modulebene sind dies etwa gute Controllability und Observability, frühe Fehlererkennung, einfache Fehlerlokalisierung, geringe Komplexität, für die normale RTL-Simulatoren oder formale Verifikationswerkzeuge reichen. Das sind entscheidende Vorteile gegenüber der heute geübten Praxis, FW und HW erst für die Systemsimulation, und dann entlang des Signalinterfaces zwischen HW und FW zu integrieren. Diese monströse Systemsimulation erfordert Abstraktion und/oder Hardwarebeschleuniger.

Darüber hinaus passt die vorgeschlagene Methodik auch gut zu üblicher top-down-orientierter Systementwurfspraxis durch fortgesetzte Verfeinerung funktionaler Einheiten, die durch Ihre APIs charakterisiert werden. Außerdem ergibt sich auf natürliche Weise die Unterstützung hostbasierter Pre-Silicon-Systemsimulation, mit der FW verifiziert wird. Weitere Vorteile werden im ausführlichen Sachbericht 2, Abschnitte 3.2.2 und 3.2.3 vorgestellt.

Zur Unterstützung dieser Methodik wurde auf der OneSpin-Toolbasis ein Modellgenerierungswerkzeug prototypisch entwickelt, das aus dem HW-Modul und dem Low-Level-Treiber ein gemeinsames Modell baut. Bei der Übersetzung des Treibers kommt OneSpins SystemC-Frontend zum Einsatz, das für diesen Zweck erweitert wurde.

Die Modellgenerierung wurde so gestaltet, dass sie die Timingvarianten der FW-Ausführung auf beliebigen Prozessortypen und beliebigen Bussystemen mit beliebiger zusätzlicher Buskommunikation nachbilden kann und an das Signalinterface des HW-Moduls anlegt. Dadurch identifiziert die formale Verifikation verlässlich alle Synchronisationsprobleme. Der Modellgenerierungsprototyp wurde zum Patent angemeldet, die Prüfung läuft.

Zur Steuerung der formalen Verifikation können Assertions im üblichen SVA-Subset eingesetzt werden. Bei der Erfolgskontrolle wurden darüber hinaus TiDAL-Assertions mit der GapFree-Methodik demonstriert, die strukturiert alle Funktionalität untersucht.

VE-VIDES-Beitrag der OneSpin GmbH / Siemens EDA (FKZ: 16ME0248)

Sachbericht 2 (Langfassung), Autor Jörg Bormann, Siemens EDA, November 2024

Inhalt

Vorbemerkung 2
1 Überblick 2
1.1 Beitrag zu förderpolitischen Zielen 2
1.2 Verwertung 2
2 Verifikation der Vertrauenswürdigkeit von RTL-Beschreibungen von Digitalschaltungen 3
2.1 Geschäftliche Zielsetzungen 3
2.2 Common Weakness Enumeration 4
2.3 Entwicklungsziele 4
2.4 Resultate der Beiträge 4
2.4.1 Konzeption und Funktion des Frameworks 4
2.4.2 Kategorien 7
2.5 Durchführung der Forschungsarbeiten 10
B3.5.1 / 2: Konzeption und Implementierung eines Frameworks für Securityverifikation / Implementierung der Debugging-Features des Frameworks für Security-Verifikation 10
B3.5.3: Prototypische Entwicklung von zwei Plugins 10
B3.5.4: Demonstration an einem Beispiel 11
2.6 Verwertung 11
2.7 Weitere Schritte 12
3 Firmware/Hardware-Koverifikation 12
3.1 Problemstellung 12
3.2 Resultate 13
3.2.1 Problemanalyse 13
3.2.2 Hierarchische Verifikation von Embedded Systems 15
3.2.3 Embedded Modules 15
3.2.4 Berechnungsmodell für die formale Verifikation von Embedded Modules 16
3.2.5 Formale Verifikation eines Embedded Modules 17
3.3 Durchführung der Arbeiten 18
B3.3.3: Analyse des Beispiels von IFX 18
B3.3.4: Konzeption HW/FW-Verifikation für beide Ansätze 18
B3.3.5 Prototypische Implementierung der hochprioren Erweiterungen am C++/SystemC-Frontend und der HW-Verifikationsumgebung 18
B3.3.6 Übersetzer für HW-Randbedingungen in Assertions für die FW 19
B3.3.7 Anwendung der formalen HW/SW Verifikation auf das IFX-Beispiel 19
3.4 Verwertung und nächste Schritte 19

Vorbemerkung

Kurz nach Beginn des VE-VIDES Projektes wurde die OneSpin Solutions GmbH von Siemens EDA übernommen. Siemens EDA ersetzte die OneSpin Solutions GmbH als Partner im VE-VIDES-Konsortium. Nachfolgend wird daher durchgängig das neue Partnerkürzel SEDA statt des alten Partnerkürzels OSS verwendet.

1 Überblick

SEDA dankt für die Initiative des BMBF im Hinblick auf die Vertrauenswürdigkeit von Digitalschaltungen. Die VE-VIDES-Arbeiten, über die hier berichtet wird, hatten zwei Forschungsrichtungen:

- Verifikation der Vertrauenswürdigkeit digitaler Schaltungen (Aufgabe 3.5 Entwicklung eines Szenariobasierten Test-Frameworks)
- FW/HW-Koverifikation (Aufgabe 3.3)

Die Forschungsrichtungen werden in dieser Reihenfolge präsentiert, weil die Verwertung der Vertrauenswürdigkeit derzeit erfolgreicher ist als die Verwertung der FW/HW-Koverifikation.

1.1 Beitrag zu förderpolitischen Zielen

Siemens EDA hat in VE-VIDES die folgenden Schwerpunkte aus der Richtlinie zur Förderung von Forschungsvorhaben für „Vertrauenswürdige Elektronik (ZEUS)“ vom 16.03.2020 aufgegriffen:

- **Neue EDA-Methoden**: Weil sich security-relevante Schwachstellen in Simulationen normalerweise nur sehr selten bemerkbar machen, hat Siemens EDA ein Werkzeug zur **formalen Verifikation** der Identifikation solcher Schwachstellen erforscht.
- Siemens EDA hat neue **Methoden zur Verhinderung von Fehlfunktionen, Schwachstellen und Hintertüren** erforscht.
- Die erforschten Prototypen helfen bei der Identifikation von **Angriffen entlang der Wertschöpfungskette**, indem
    - Eingangstests für third-party IP ermöglicht werden.
    - entwurfsbegleitend Security-Schwachstellen aufgedeckt werden können
    - mit der formalen HW/SW-Co-Verifikation sichergestellt werden kann, dass HW-Alarme korrekt an die sie verarbeitende Software weitergeleitet werden.
- Dadurch tragen die Forschungen zur **Technologiesouveränität Deutschlands** bei.
- Im Rahmen der Forschungen zu HW/SW-Co-Verifikation hat Siemens EDA einen Prototypen zur formalen Verifikation der Integration von HW und SW erarbeitet und schließt damit eine Lücke zwischen jeweils HW- und SW-fokussierten Forschungen in Bezug auf **vertrauenswürdige Elektronik**.
- Für die formale HW/SW-Co-Verifikation wurden neue Herangehensweisen und **neue Modelle für die formale Verifikation** untersucht, die das Verhalten von HW und SW bei **geringer Komplexität** repräsentieren.

1.2 Verwertung

Die Übernahme der OneSpin Solutions GmbH durch Siemens EDA führte zu einer Neubewertung der Arbeitspakete. Dabei erhielt das Ziel der Marktführerschaft bei der formalen Verifikation der Vertrauenswürdigkeit digitaler Schaltungen eine hohe Priorität. Dieser hohen

Priorität ist SEDA erfolgreich nachgekommen und hat einen Prototypen erforscht, der unmittelbar nach Projektende zum Produkt „Questa Verify Trust“ weiterentwickelt wurde und mittlerweile erfolgreich vermarktet wird.

Der Stellenwert der FWHW-Koverifikation relativierte sich vor dem Hintergrund bereits ausgereifter Siemens-EDA-Produkte zur Systemverifikation (z.B. Veloce™) und zur Übersetzung von C/C++/SystemC-Beschreibungen in formale Modelle (wie etwa das Frontend des Äquivalenzcheckers für High-Level-Synthese Questa SLEC). Für SEDA waren die Forschungen an FW/HW-Koverifikation vor allem als Grundlagenforschung über hierarchischen Entwurf und Verifikation von Embedded Systems wichtig, die sich aus GVB und TVB ergab. Durch die VE-VIDES-Forschungen ist SEDA auf Kundenanfragen nach innovativen Entwurfs- und Verifikationsverfahren für Embedded Systems gut vorbereitet. Derartige Kundenanfragen werden wegen der wachsenden Größe von Embedded Systems in Zukunft erwartet. Insbesondere wurde der erforschte Prototyp zum Patent angemeldet, mit dem hierarchische Verifikation von Embedded Systems unterstützt wird. Die Patentprüfung läuft.

2 Verifikation der Vertrauenswürdigkeit von RTL-Beschreibungen von Digitalschaltungen

Aufgabe 3.5 „Entwicklung eines Szenariobasierten Test-Frameworks“, Partner: TU Chemnitz, Zentrale Forschung der Siemens AG.

2.1 Geschäftliche Zielsetzungen

Bei der Mitarbeit im Projekt VE-VIDES und Scale4Edge ging es SEDA darum, Technologieführerschaft in dem Teilbereich der Vertrauenswürdigkeit von RTL-Beschreibungen zu entwickeln. RTL-Beschreibungen sehen aus wie Programme und legen die Funktionalität einer Digitalschaltung fest. Die RTL-Beschreibung ist der Startpunkt eines stark automatisierten Entwurfsprozesses, an dessen Ende die fertige integrierte Schaltung steht.

SEDA hält formale Verifikation für eine Schlüsseltechnologie bei der Verifikation der Vertrauenswürdigkeit von Digitalschaltungen. Denn formale Verifikation kann Probleme auch dann finden, wenn ihr Auftreten so unwahrscheinlich ist, dass sie dem heutigen Verifikationsstandard der Coverage Driven Constrained Random Simulation entgehen. Das ist gerade dann wichtig, wenn das RTL in der böswilligen Absicht verändert wurde, spätere Attacken zu erleichtern. Solche Veränderungen sind so gestaltet, dass sie nahezu nie zufällig aktiviert werden. Wenn aber Angreifer wissen, wie sie aktiviert werden können, können sie darauf aufbauend die Sicherheit der Schaltung attackieren.

SEDA hat in den Förderprojekten VE-VIDES und Scale4Edge zwei sich ergänzende Prototypen entwickelt und nach Ende der Projekte zu folgenden Werkzeugen weiterentwickelt:

- Questa Verify Secure™ zum Prüfen dedizierter im RTL beschriebener Security-Mechanismen, die den Zugriff auf geheime oder wichtige Informationen (sog. Assets) beschränken. Dieses Werkzeug vermarktet Forschungen in Scale4Edge.
- Questa Verify Trust™ zum Prüfen der RTL-Beschreibungen auf versehentlich oder absichtsvoll eingefügte Schwachstellen, die Angriffe auf die Vertraulichkeit geheimer Daten (Confidentiality), Konsistenz wertvoller Daten (Integrity) oder Verfügbarkeit von Daten (Availability) ermöglichen. Dieses Werkzeug vermarktet die Forschungen in VE-VIDES, die in diesem Kapitel dieses Berichts beschrieben werden.

2.2 Common Weakness Enumeration

Zur Erforschung des Prototyps für Questa Verify Trust hat SEDA am VE-VIDES-Arbeitspaket 3.5 „Entwicklung eines Szenario-basierten Test-Frameworks“ mitgearbeitet und dabei den Aufgabenbereich übernommen, der sich mit der Vertrauenswürdigkeit von Digitalschaltungen beschäftigt. Die Szenarien sind dabei öffentlich verfügbare Beschreibungen von Security-Schwachstellen.

SEDA konzentriert sich dabei auf den einschlägigen Quasi-Standard, die Datenbasis Common Weaknesses Enumeration CWE, verfügbar unter cwe.mitre.org. Diese Datenbasis repräsentiert den Stand der Erkenntnis über Sicherheits-Schwachstellen. Wenn in einem Produkt eine konkrete Security-Lücke (Vulnerability) gefunden wird, wird diese analysiert und auf ihren grundlegenden Mechanismus reduziert. Wenn dieser Mechanismus noch nicht in CWE beschrieben ist, wird eine neue Schwachstellenbeschreibung in CWE als neue Weakness eingefügt. Eine Weakness beschreibt also eine ganze Klasse möglicher Vulnerabilities. Systemdesigner oder Verifikationsmanager können CWE nutzen, um Schwachstellen zu identifizieren, die ihr Design haben könnte. Sie können davon ausgehend Systemarchitektur und -verifikation geeignet gestalten, um später potentiellen Kunden versichern zu können, dass das System gegen alle nach dem aktuellem Stand der Technik bekannten Schwachstellen geprüft wurde.

2.3 Entwicklungsziele

Ausgangspunkt für die VE-VIDES-Forschungen waren verschiedene formale Verifikationsverfahren im OneSpin-Portfolio, die eine Auswahl von Schwachstellen aufdecken können. Diese Verifikationsverfahren waren über verschiedene OneSpin-Werkzeuge und -Prototypen verteilt. Ihre Relevanz für die Vertrauenswürdigkeit war nur für Spezialisten offensichtlich. Ziel der Forschung war deshalb

- Die Bedeutung der Verfahrens-Resultate für die Vertrauenswürdigkeit herauszuarbeiten
- Leistungsfähigkeit, Analysemöglichkeiten und Vielfalt dieser Verfahren zu erweitern
- Die Verfahren in einem Framework für einen Verifikationsprozess zu integrieren
- Das Framework so zu gestalten, dass es leicht erweitert werden kann

Zum Erreichen dieser Ziele wurden die folgenden VE-VIDES-Beiträge definiert:

- B3.5.1: Konzeption und Implementierung eines Frameworks für Securityverifikation
- B3.5.2: Implementierung der Debugging-Features des Frameworks für Security-Verifikation
- B3.5.3: Prototypische Entwicklung von zwei Plugins
- B3.5.4: Demonstration an einem Beispiel

2.4 Resultate der Beiträge

2.4.1 Konzeption und Funktion des Frameworks

Integration der Verifikationsverfahren

Die Verifikationsverfahren werden als Plugins in das Framework eingehängt. Um ein Verifikationsverfahren zu einem Plugin zu machen, wird es mit APIs ausgestattet, mit dem das Framework

- Das jeweilige Verifikationsverfahren konfiguriert und startet
- Konsistenz von Resultaten sicherstellt

- Resultate abholt
- Spezifisches Debugging aufruft

Kategorien

Die unterstützten Security-Beweisziele werden als (Verifikations-)kategorien bezeichnet. Eine Liste der Kategorien folgt in Abschnitt 2.4.2. Die SEDA-Forschungen ergaben, dass es zwischen Plugins und Kategorien nicht die bei Antragstellung unterstellte 1-zu-1-Beziehung gibt. Einerseits waren für ähnliche Verifikationsaufgaben (wie etwa die Suche nach unbenutztem Code) mehrere Plugins erforderlich, teilweise wurden mehrere Kategorien mit dem selben Plugin realisiert, das unterschiedlich konfiguriert wird.

Verifikationsprozess

Eine Analyse der vor Projektstart existierenden formalen Verifikationsverfahren zeigte, dass sie nur Verdachtsfälle identifizieren können, die nicht notwendigerweise sicherheitsrelevant sind. Sie müssen weiter analysiert werden, um den Verdacht zu erhärten und durch RTL-Änderungen zu begegnen, oder um den Verdacht zu verwerfen. Der Gewinn für den Benutzer besteht darin, vom Werkzeug auf Stellen im RTL-Code hingewiesen zu werden, die besondere Beachtung erfordern.

Grafisches Benutzerinterface

Diesem Arbeitsablauf trägt das grafische Benutzerinterface (GUI, s. Abbildung 1) des Prototypen Rechnung: Die Verdachtsfälle werden darin als Trust Issues gelistet. Das GUI zeigt dabei Informationen über Details der Trust Issues an wie etwa

- Eine detailiertere Beschreibung der möglichen Schwachstelle
- Die dafür relevanten Schwachstellen-Einträge der CWE-Datenbasis
- Das RTL-Objekt, das für die Schwachstelle relevant ist
- Der Ort im RTL-Code, der für die Schachstelle relevant ist

Das GUI unterstützt den Arbeitsablauf wie folgt:

- Auswahl und Konfiguration der Kategorien
- Auswahl der angezeigten Trust-Issues durch verschiedene Filter.
- Definition der Reihenfolge der Analyse der Trust-Issues durch Verändern von Prioritäten
- Start der Analyse aus dem GUI. Analyse-Unterstützung durch Source-Code-Browsing und durch plugin-spezifische Unterstützung.
- Dokumentation der Gründe für das Verwerfen eines Trust-Issues
- Die Datenbasis der Trust-Issues kann komplett oder gefiltert in ein .csv-File exportiert werden, z.B. für Reporting innerhalb eines Entwicklungsprojekts. Die .csv-Files können in Microsoft Excel™ importiert werden und dort weiterverarbeitet werden.

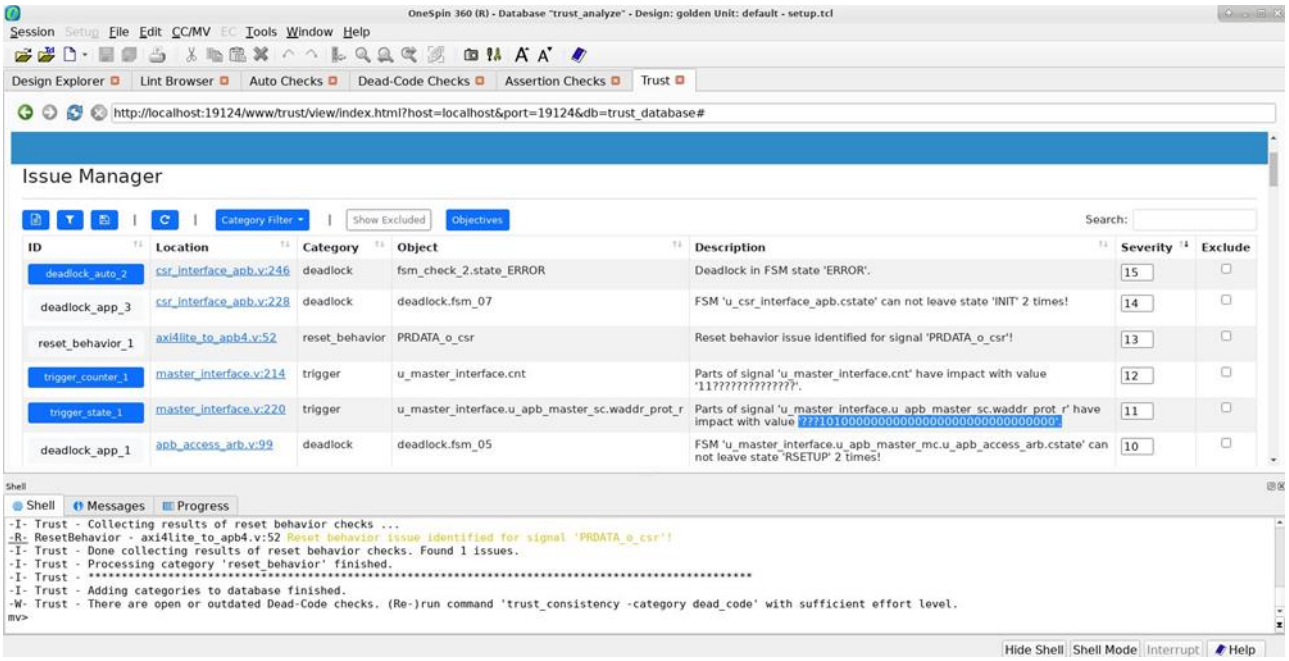

Abbildung 1: Grafisches Benutzerinterface

Beziehung zwischen Kategorien und CWE-Einträgen

Um die formalen Verifikationsverfahren und die Schwachstellenbeschreibungen in CWE in Beziehung zu setzen, war deutlich mehr Aufwand erforderlich, als beim Verfassen von GVB und TVB erwartet wurde. Das liegt einerseits an der schieren Größe der CWE-Datenbasis, die mittlerweile mehr als 1000 Einträge hat. Diese Einträge sind teilweise spezifisch für Software, teilweise spezifisch für Hardware, und teilweise bedienen sie beide Implementierungsarten, ohne dass das immer unmittelbar offensichtlich ist. Es wurde zunächst das Ziel verfolgt, jedem unterstützten CWE-Eintrag eine Kategorie zuzuordnen. Dafür erwiesen sich die formalen Verifikationsverfahren aber als zu universell. So gibt es beispielsweise recht viele CWE-Einträge über die Verletzung von Vertraulichkeit, die sich nur in den unzulänglichen Mechanismen unterscheiden. Das entsprechende Beweisverfahren ist aber auf alle Mechanismen anwendbar. Es wurde deswegen auf einen 1-zu-1-Bezug zwischen CWE-Einträgen und Kategorien verzichtet, sodass eine Kategorie nun mehrere CWE-Einträge bedient.

Umgekehrt scheinen aber auch CWE-Einträge zu fehlen, wie z.B. das Security-Risiko, das von kombinatorischen Schleifen ausgeht. SEDA arbeitet daher in der CWE Special Interest Group für Hardware-Weaknesses mit, um solche CWE-Einträge nachzurüsten. Diese Special Interest Group ist ein firmenübergreifendes Gremium zur Pflege der Hardware-spezifischen Einträge in CWE.

Beziehung zum SA-EDI-Standard

Zwischenzeitlich hat SEDA daran gearbeitet, den VE-VIDES-Prototyp um eine Unterstützung des SA-EDI-Standards (https://www.accellera.org/downloads/standards/ip-security-assurance) von Accelera zu erweitern. Dieser Standard legt Beschreibungsverfahren fest, mit denen Anbieter von Schaltungs-IP den IP-Integratoren helfen können, die Security der IP geeignet zu schützen und diesen Schutz auch gründlich zu verifizieren. Im Prinzip ist diese Aktivität für die Verwertung der hier dokumentierten Forschungen höchst relevant. Eine interne Analyse der ersten Version dieses Standards säte jedoch Zweifel an seiner baldigen Annahme durch Kunden, sodass diese Aktivitäten eingestellt wurden. Statt dessen arbeitet SEDA im Komitee für den Nachfolgestandard IEEE P3164 mit.

2.4.2 Kategorien

Aus Benutzersicht verfolgt der Prototyp bestimmte Beweisziele, die sog. Kategorien bilden. Diese Kategorien sollen nachfolgend vorgestellt werden.

Identifikation von Trojanern durch ihre Trigger

Angriffe auf die Sicherheit einer Schaltung können durch bösartige zusätzliche Logik vorbereitet werden, die häufig als "Trojaner" oder "Trojanisches Pferd" bezeichnet wird.

Trojaner bestehen in der Regel aus einem Trigger und einer Nutzlast. Der Trigger entscheidet, wann die Nutzlast aktiviert wird. Die Nutzlast hilft während eines Angriffs auf die Security. Herzstück eines Triggers ist eine Bedingung, die nur ganz selten erfüllt wird. Solche Bedingungen werden von dem Plugin gesucht.

Trojaner können im Prinzip auch anhand ihrer Nutzlast identifiziert werden, aber dafür sind andere Kategorien erforderlich.

Relevante CWE-Einträge:

CWE-Eintrag	Titel
506	Embedded Malicious Code
507	Trojan Horse
511	Logic/Time Bomb
1242	Inclusion of Undocumented Features or Chicken Bits

Identifikation von Deadlocks

Ein Deadlock ist eine Situation, in der eine Schaltungskomponente einen Zustand ohne weiteren Fortschritt erreicht. Deadlocks können als versehentliche Fehler in das RTL eingeführt werden, oder absichtsvoll als Nutzlast eines Trojaners. In beiden Fällen ermöglichen sie Denial-of-Service-(DoS-)Angriffe auf die Verfügbarkeit (Availability) der Komponente.

Ein einfacher Deadlock entsteht z.B. durch FSM-Zustände, die nur durch Zurücksetzen verlassen werden können. Komplexe Deadlocks entstehen z.B. in einem Bussystem, in dem Daten- oder Adresspuffer zirkulär darauf warten, dass der nächste Puffer leer wird. Während formale Tools bei der automatischen Identifizierung einfacher Deadlocks effektiv sind, stellen komplexe Deadlocks oft erhebliche Herausforderungen dar und können sich aller Verifikation bis zur Produkteinführung entziehen.

CWE-Eintrag	Titel
835	Loop with Unreachable Exit Condition ('Infinite Loop')
1245	Improper Finite State Machines (FSMs) in Hardware Logic (In Bezug auf Deadlock States)

Identifikation von ungenutzer Logik

Logik wird als „ungenutzt“ bezeichnet, wenn sie nicht zum Ein-Ausgabeverhalten der Schaltung beiträgt. Typische Fälle sind Logik, die durch dead code im RTL erzeugt wird, und Logikanteile ohne Verbindung zu einem Ausgang.

Auf dem Chip wird ungenutzte Logik mit Takt und Power versorgt und führt daher Berechnungen durch. Allerdings werden die Resultate immer verworfen, bevor sie ein Outputsignal beeinflussen können. Ungenutzte Logik kann die Security einer Schaltung wie folgt kompromittieren:

- Elektromagnetische Abstrahlung geheimer Information (Radiation-Sidechannel)
- Kodierung geheimer Information über den Stromverbrauch (Power-Sidechannel)
- Lokale Überhitzung der Schaltung als Fault-Insertion-Angriff
- Übermäßiger Stromverbrauch, der zu frühem Ausfall eines batteriegetriebenen Produkts führt

Synthesewerkzeuge identifizieren viele Fälle von ungenutzter Logik und entfernen sie. Allerdings ist die Erkennung von totem Code im allgemeinen NP-vollständig, sodass es Fälle gibt, bei denen die Synthese sicherheitshalber ungenutzte Logik auf dem Chip implementiert.

Keiner der nachfolgend gelisteten CWE-Einträge trägt dem tatsächlichen Security-Risiko von ungenutzter Logik Rechnung. SEDA hat daher in der Special Interest Group für HW-Schwachstellen eine passende Ergänzung vorgeschlagen.

CWE-Eintrag	**Titel**
561	Dead Code (Bemerkung: Dieser CWE-Eintrag beschreibt eine SW-Schwachstelle mit weniger kritischen Konsequenzen für die Security)
563	Assignment to Variable without Use (Bemerkung: Dieser CWE-Eintrag beschreibt eine SW-Schwachstelle mit weniger kritischen Konsequenzen für die Security)
1245	Improper Finite State Machines (FSMs) in Hardware Logic (In Bezug auf unerreichbare Zustände)

Verifikation von Lock-Bits und davon geschützter Datenregister

In vielen digitalen Hardware-Designs werden Register nach dem Reset des Chips programmiert und dann für weitere Änderungen gesperrt. Dieses Verhalten wird häufig mit einem Lockbit implementiert. Wenn das Lockbit gesetzt ist, werden Schreibvorgänge in gewisse geschützte Datenregister verhindert.

Zu den damit verbundenen Sicherheitslücken gehört, dass das Sperrbit gelöscht werden kann, nachdem es gesetzt wurde. Darüber hinaus muss das Lock-Bit nicht alle vorgesehenen Register wirklich schützen. In beiden Fällen können Angreifer auf die eigentlich geschützten Register zugreifen und diese ändern.

CWE-Eintrag	**Titel**
1224	Improper Restriction of Write-Once Bit Fields
1231	Improper Prevention of Lock Bit Modification
1233	Security-Sensitive Hardware Controls with Missing Lock Bit Protection

Absicherung der Integrität wertvoller Daten

Wenn ein System wertvolle Daten enthält, verfügt es über konstruktive Maßnahmen gegen böswilliges oder versehentliches Überschreiben der wertvollen Daten, wie z. B. Isolierung und Zugriffskontrolle. Diese Kategorie führt eine grundlegende Überprüfung dieser Maßnahmen durch. Die Prüfung identifiziert Pfade, die zum Ändern der wertvollen Daten verwendet werden können. Bei der Analyse der gefundenen Probleme muss zwischen den beabsichtigten Pfaden, die zum Speichern der wertvollen Daten verwendet werden, und den bösartigen Pfaden unterschieden werden.

CWE-Eintrag	Titel
923	Improper Restriction of Communication Channel to Intended Endpoints
1256	Improper Restriction of Software Interfaces to Hardware Features
1274	Improper Access Control for Volatile Memory Containing Boot Code
1314	Missing Write Protection for Parametric Data Values

Absicherung der Vertraulichkeit von Daten

Wenn ein System vertrauliche Daten enthält, verfügt es über konstruktive Maßnahmen gegen unerlaubtes Lesen dieser Daten. In dieser Kategorie wird eine grundlegende Überprüfung dieser Maßnahmen durchgeführt. Beispiele dafür sind Isolation oder Zugriffskontrolle. Das Werkzeug bestimmt Pfade, über die der Inhalt eines vom Benutzer vorzugebenden Assets das Verhalten eines Ausgangssignals beeinflussen kann. Der Benutzer analysiert den Pfad, um herauszufinden, ob es sich um einen beabsichtigten Pfad handelt, der von berechtigten Benutzern zum Lesen der Daten verwendet wird, oder ob es sich um einen bösartigen Pfad handelt, der von nicht berechtigten Angreifern verwendet werden kann.

CWE-Eintrag	Titel
201	Insertion of Sensitive Information Into Sent Data
203	Observable Discrepancy
213	Exposure of Sensitive Information Due to Incompatible Policies
359	Exposure of Private Personal Information to an Unauthorized Actor
1243	Sensitive Non-Volatile Information Not Protected During Debug
1258	Exposure of Sensitive System Information Due to Uncleared Debug Information
1272	Sensitive Information Uncleared Before Debug/Power State Transition

Security einer Schaltung mit Zustandselementen ohne Reset

Zustandselemente ohne Reset machen das physikalische Design einer Schaltung billiger, bedrohen aber die Security, wenn die einschlägige Verifikation nicht alle möglichen Resetzustände abdeckt, deren Anzahl exponentiell mit der Anzahl der Zustandselemente ohne Reset wächst. Diese Kategorie bietet einen Ausweg aus diesem Konflikt, indem sie beweist, dass das Ein-/Ausgabeverhalten der Schaltung unabhängig von der tatsächlichen Wahl des Reset-Zustands ist. Wenn die Kategorie keine Abhängigkeit von der Wahl des konkreten Resetzustands findet, reicht die simulationsbasierte Verifikation von einem einzigen beliebig gewählten Resetzustand.

CWE-Eintrag	Titel
1271	Uninitialized Value on Reset for Registers Holding Security Settings
Bemerkung: Es gibt verschiedene CWE-Einträge über fehlende Initialisierung, aber alle diese Einträge behandeln die fehlende Initialisierung als Fehler, den es aufzudecken und zu beseitigen gilt. Die wirtschaftlichen Randbedingungen des Hardwareentwurfs bleben derzeit unberücksichtigt. SEDA arbeitet an der Aufnahme einer Weakness, die hier Abhilfe schafft.	

2.5 Durchführung der Forschungsarbeiten

B3.5.1 / 2: Konzeption und Implementierung eines Frameworks für Securityverifikation / Implementierung der Debugging-Features des Frameworks für Security-Verifikation

Das **Konzept des Frameworks** wurde teilweise von einem amerikanischen Kollegen entwickelt, für den keine Förderung in Anspruch genommen wurde. Es ist im Deliverable D03-3C dokumentiert, das fristgerecht abgegeben wurde.

Wegen der Übernahme des ursprünglichen Projektpartners OneSpin Solutions durch Siemens EDA verschob sich die **Implementierung dieses Konzepts** sowie des zugehörigen **Debuggers** (B 3.5.2) um 6 Monate. Die Verschiebung des zugehörigen Deliverables D06-3E „Prototypisches Framework für die formale Security-Verifikation“ von August 2022 auf Februar 2023 wurde von Herrn Yilmaz am 12. April 2022 genehmigt. Das Deliverable wurde rechtzeitig zum neuen Termin angefertigt. Die Verschiebung hatte keine Auswirkungen auf andere Arbeiten im VE-VIDES-Konsortium.

B3.5.3: Prototypische Entwicklung von zwei Plugins

In das Framework wurden Plugins entsprechend der folgenden Tabelle integriert:

Kategorie	Beitrag lt. GVB	Zustand vor VE-VIDES	Arbeiten innerhalb von VE-VIDES
Triggeridentifikation		Prototyp vorhanden	Integration als Plugin
Deadlocks		Prototyp vorhanden	Integration von zwei Plugins
Ungenutzte Logik		Funktion bestehender Apps	Integration als Plugin
Integrität	3.5.1d	Scale4Edge-Prototyp	Integration als Plugin, Optimierung von Kapazität und Laufzeit
Vertraulichkeit	3.5.1d	Scale4Edge-Prototyp	Integration als Plugin, Optimierung von Kapazität und Laufzeit
Zustandselemente ohne Reset	3.5.3	Nicht existent	Entwicklung der Funktionalität, Integration als Plugin, Optimierung von Kapazität und Laufzeit, Entwicklung von Debugging-Unterstützung, Dokumentation charakteristischer Anwendungsfälle
Lock-Bits	3.5.3	Nicht existent	Entwicklung der Funktionalität, Integration als Plugin, Entwicklung von Analyseunterstützung, Dokumentation charakteristischer Anwendungsfälle

Die Grundfunktionen für die Kategorien Triggeridentifikation, Deadlocks und Ungenutzte Logik existierten bereits bei Start von VE-VIDES. Diese Grundfunktionen wurden zu Plugins weiterentwickelt. Die Plugins wurden in das Framework integriert.

Die Grundfunktionalität für Integrität und Vertraulichkeit ergaben sich aus dem Forschungsprojekt Scale4Edge während der Projektlaufzeit von VE-VIDES. Für die Entwicklung dieser Grundfunktionalität wurde keine VE-VIDES-Förderung in Anspruch genommen.

Mit VE-VIDES-Förderung wurde die Grundfunktionalität zu Plugins weiterentwickelt, optimiert und in das Framework integriert, und so die Ankündigungen in der TVB erfüllt, dass diese Verifikationsziele unterstützt werden. Im Ggs. zur TVB zählt SEDA sie aber nicht als Plugins, die im VE-VIDES-Beitrag B3.5.3 neu entwickelt wurden.

Bei Antragstellung zu VE-VIDES wurde bereits erwartet, dass die Plugin-Entwicklung dynamisch werden würde. Daher wurde in der Beitragsbeschreibung zu B3.5.3 vorgesehen, dass die zwei in VE-VIDES zu entwickelnden Plugins kurzfristig festgelegt werden. Die Wahl fiel auf die Plugins für

- Verifikation der Security einer Schaltung mit Zustandselementen ohne Reset und
- Verifikation von Lock-Bits und davon geschützter Datenregister

Diese Plugins sind in VE-VIDES neu erforscht und gefördert worden.

Die beiden VE-VIDES-Plugins sind fristgerecht entwickelt worden und im Deliverable D10-3H „Zwei Plugins für formale Securityverifikation“ anhand jeweils eines charakteristischen Einsatzfalls beschrieben.

Die Integration der Verifikationsverfahren als Plugins zeigte Schwierigkeiten auf, die beim Verfassen von GVB und TVB nicht erwartet wurden. Sie erfordern größere Umbaumaßnahmen im Basiswerkzeug, auf die hier nicht näher eingegangen werden soll. Die Umbaumaßnahmen wurden während VE-VIDES spezifiziert und nach Abschluß von VE-VIDES in Angriff genommen. Bis zu ihrer vollständigen Realisierung bleibt die Integration neuer Kategorien jedoch ein fallspezifisches Unterfangen, das nicht so schnell durchgeführt werden kann wie ursprünglich erhofft.

Insgesamt war der Aufwand rund um die Integration der Beweisverfahren in das Framework deutlich höher als vorher erwartet. Der Zusatzaufwand wurde ausgeglichen durch Aufwandsersparnis bei den Forschungen an der FW/HW-Koverifikation.

B3.5.4: Demonstration an einem Beispiel

Wegen der großen Geschäftserwartungen im Zusammenhang mit dem VE-VIDES-Prototypen wurde schon vor Ende von VE-VIDES mit Vorbereitungen zur Produktisierung des Prototypen als neues Siemens-EDA-Werkzeug „Questa Verify Trust“ begonnen. Bei der Einarbeitung von kundennah arbeitendem Personal entstand fristgerecht das Deliverable D12-3H „Demonstration der Plugins an einem Beispiel“. Da dieses neue Personal nicht förderberechtigt war, wurde für die Anfertigung dieses Deliverables keine Förderung in Anspruch genommen.

2.6 Verwertung

SEDA betrachtet die Weiterentwicklung des VE-VIDES-Prototypen in das Siemens-EDA-Produkt „Questa Verify Trust“ und die ausgesprochen positive Kundenresonanz als stärksten Beleg dafür, dass die einschlägigen Forschungsanstrengungen in VE-VIDES erfolgreich waren.

SEDA scheint mit Questa Verify Trust ein Werkzeug anzubieten, das derzeit keine Konkurrenz hat.

Das Werkzeug wird zumeist als Teil des eingangs skizzierten umfassenderen formalen Angebots für die Verifikation der Vertrauenswürdigkeit von Digitalschaltungen präsentiert. Derartige Präsentationen gibt es im direkten Kundenkontakt, auf selbst organisierten Veranstaltungen wie etwa Webinaren und der Nutzerkonferenz Osmosis (17.10.2024, München), als auch auf externen Events wie

- DVCon Europe (15.-16.10.2024, München)
- Automotive Tech Day (24.9.2024, München)
- Design Automation Conference (23.-27.6.2024, San Francisco)
- Tag der vertrauenswürdigen Elektronik (4.-5.6.2024, München)
- EDA-Workshop (9.-10.4.2024, Dresden, Vortrag und Poster)

Eine weitere Verwertungsaktivität ist die oben erwähnte Mitarbeit in folgenden firmenübergreifenden Gremien:

- CWE Special Interest Group für Hardware-spezifische Schwachstellen (CWE HW SIG)
- Arbeitsgruppe zur IEEE-Standardisierung P3164 von Security Issurance Collaterals im IP-Business

2.7 Weitere Schritte

Das Werkzeug wird in drei Richtungen weiterentwickelt:

- Mehr Automatisierung – automatische Auffindung von problematischen Stellen
- Weitere CWEs
- Vermeidung falscher Alarme
- Höhere Kapazität, d.h. größere Schaltungen

Bei diesen Weiterentwicklungen arbeitet SEDA eng mit verschiedenen akademischen Partnern zusammen, insbesondere mit Prof. Kunz von der RPTU Kaiserslautern.

3 Firmware/Hardware-Koverifikation

Aufgabe 3.3 „Formale HW-SW-Co-Verifikation“, Partner Infineon Technologies AG. Infineons Forschungsschwerpunkt ist die automatische Generierung von Eigenschaften über FW mit und ohne HW, SEDAs Forschungsschwerpunkt war der Ausbau formaler Methoden für die FW/HW-Koverifikation.

3.1 Problemstellung

Die meisten heutigen Digitalprodukte sind Embedded Systems, d.h. ihre Funktionalität wird teilweise durch Hardware (HW) und teilweise durch Software implementiert. Die Software wird üblicherweise als Firmware (FW) bezeichnet, weil sie integraler Bestandteil der Implementierung der Produktfunktionalität ist.

Nach dem Stand der Technik werden FW und HW über lange Zeit getrennt entwickelt und erst ziemlich spät und als ziemlich große Monolithen zusammengebracht. Das macht die Verifikation ihres Zusammenspiels langsam, unhandlich, aufwändig und birgt ein hohes Risiko von unerkannten Missverständnissen zwischen Hardware und Firmware, die sich beim späteren Einsatz des Produktes als Fehler manifestieren.

Im Bereich der Security sind diese Missverständnisse besonders gefährlich, weil sie Security-Mechanismen versagen lassen können und das Versagen von Security-Mechanismen nur dann bemerkt wird, wenn sie eigentlich einen Angriff abwehren sollten. Darüber hinaus birgt jeder funktionale Fehler die Gefahr der Nutzung bei einem Angriff auf die Security des Produkts.

Ziel der Aufgabe war es, die Gefahr funktionaler Fehler aufgrund von Missverständnissen zwischen FW und HW durch den Einsatz formaler Verifikation generell zu reduzieren, und dadurch die Vertrauenswürdigkeit des Embedded Systems zu erhöhen. Die

Komplexitätsbeschränkungen formaler Methoden erfordern dabei Divide-and-Conquer-Ansätze, die letztendlich nahelegten, Embedded Systems hierarchisch zu verifizieren.

In Prof. Djones Lettnin, Bryan Olmos, Johannes Grinschgl, Paulette Hatem und anderen Infineon-Kollegen fand SEDA Partner, die über die FW/HW-Koverifikation ähnlich denken und mit denen diese Aufgabe partnerschaftlich bearbeitetet wurde. SEDA dankt den Infineon-Kollegen für interessante Beispiele, Diskussionen und Ermutigung.

3.2 Resultate

3.2.1 Problemanalyse

Infineon stellte dankenswertertweise das Design eines safety-kritischen Sensors zur Verfügung, um SEDA einen Einblick in Problemstellungen zu ermöglichen, die für die Embedded Systems im Automobil typisch sind. Diese Einblicke wurden bei SEDA mit Berufserfahrung aus Design und Verifikation von Mobilfunk-Modems komplementiert, die ihrerseits komplexe Embedded Systems sind.

Die Problemanalyse bewertete die heute übliche Verifikationsmethodik für embedded Systems:

1. Entsprechend heute geübter Praxis beginnt die Verifikation des Zusammenspiels zwischen HW und FW bereits mit großen HW- und FW-Teilen, sodass sie eigentlich gleich eine Simulation auf Systemebene bildet.
2. Aufgrund der Komplexität kommen Simulationen mit abstrakten HW-Modellen bzw. FPGA-Beschleuniger zum Einsatz, um für hinreichende Performance zu sorgen. Ob dieser Vorgehensweise auch für zukünftige noch größere Embedded Systems skaliert, erscheint fraglich.
3. Simulationen mit abstrakten HW-Modellen sind nicht präzise genug, um alle Fehler aufzudecken.
4. FPGA-Beschleuniger sind teuer. Ihre Bedienung erfordert besondere Expertise, und teilweise sogar Änderungen im HW-Design, mit denen wiederum Fehler maskiert werden können. FPGA-Beschleuniger erschweren Identifikation und Analyse von Fehlverhalten, weil sie nur eingeschränkte Sicht in die Abläufe des untersuchten Systems ermöglichen.
5. Die heute geübte Praxis hat besondere Probleme bei der strukturierten Verifikation der folgenden Fehlerklassen:
    - Fehlerhafte Synchronisation zwischen FW und HW
    - Die FW generiert Transaktionsfolgen, die von der HW nicht unterstützt werden
    - Die FW übergibt Daten in einer anderen Kodierung als von der HW erwartet
    - Die FW übergibt Daten bzw. Befehle, die von der HW nicht unterstützt werden

    Derartige Fehler werden hier als „Missverständnisse zwischen FW und HW“ bezeichnet.
6. Missverständnisse zwischen FW und HW sind vergleichsweise häufig, weil hier zwei unterschiedliche Entwicklergruppen ein gemeinsames Interface bedienen. Die Fehleranfälligkeit solcher Integrationspunkte ist bereits aus der reinen HW-Verifikation bekannt. Darüber hinaus sind hier Entwicklergruppen mit fundamental unterschiedlicher Expertise beteiligt, was die Fehlerwahrscheinlichkeit weiter erhöht.
7. FW-Entwickler äußern immer wieder den Wunsch, dass die HW-Interfaces durch zusätzliche HW-Logik einfacher oder sogar gegen Missverständnisse tolerant gestaltet werden. Aber das führt zu zusätzlichen HW-Kosten aufgrund der zusätzlichen Logik, während entsprechende FW-Anpassungen nahezu kostenneutral sind.
8. Weitere Kosten entstehen bei der HW-Verifikation, die vorsorglich mehr potentielles FW-Verhalten verifiziert als später von der tatsächlichen FW verlangt wird.

9. Missverständnisse zwischen FW und HW führen nicht zu einer expliziten Fehlermeldung, sondern die HW macht irgendwie weiter. In einer Simulation auf Systemebene kann das sich ergebende Fehlverhalten unbemerkt bleiben (mangelnde Observability) oder erst einige Simulationszeit nach dem Auftreten des Missverständnisses detektiert werden, was die Fehleranalyse aufwändig macht.
10. Aufgrund der heutigen Größe des simulierten Systems können spezifische Situationen im Zusammenspiel von FW und HW nur mit Mühe detailliert eingestellt werden (mangelnde Controllability). Strukturierte Verifikation zur Vermeidung von Missverständnissen zwischen FW und HW ist damit zumindest sehr aufwändig, wenn nicht unmöglich.
11. Die heutige Verifikationspraxis mit großen FW- und HW-Monolithen erfordert die Synchronisation vieler Entwicklerteams. Die FW-HW-Koverifikation kann erst beginnen, wenn das letzte Entwicklerteam geliefert hat. Das führt für die anderen Entwicklerteams zu größeren Pausen zwischen Entwicklung und Verifikation, während denen Einsichten in die Entwicklung vergessen werden.
12. Insgesamt beginnt die HW/FW-Koverifikation häufig erst so spät, dass Fehler wie etwa FW/HW-Missverständnisse nicht mehr durch HW-Änderungen korrigiert werden können, weil der physikalische HW-Entwurf bereits begonnen hat. Sie müssen dann durch FW-Änderungen beseitigt werden, die entweder risikoreich sein können oder teuer, weil sie die Aufgabe von Produktfeatures erfordern.
13. Die eigentliche Aufgabe der Systemsimulation ist die Überprüfung von Systemaspekten. Dazu gehören nicht nur systemweite Algorithmen, sondern auch systemweit relevante Entwurfsentscheidungen wie die Auswahl von Arbitrierungsmechanismen oder die Dimensionierung von Systemresourcen. Die Beseitigung von Missverständnissen zwischen HW und FW stören hierbei, indem sie immer wieder Fehleranalysen erfordern, die unnötig aufwändig sind, weil sie auf der Systemebene durchgeführt werden müssen.

Die Problemanalyse ergab im Hinblick auf formale Verifikation:

14. Heute kommunizieren FW-Routinen, die Schreib- und Lesetransaktionen der Hardware direkt auslösen, häufig mit mehreren Spezial-HW-Modulen. Die Komplexität dieser FW-Routinen variiert stark zwischen verschiedenen Embedded Systems. Die Verifikation des Zusammenspiels einer solchen FW-Routine mit der HW führt dadurch zu sehr unterschiedlichen Komplexitäten und dadurch zu schwer vorhersagbaren Aufwänden insbesondere der bei der formalen Verifikation.
15. Kompositionale formale Verifikation bietet sich als Ausweg aus der hohen Komplexität eigentlich an, erfordert aber eine Abbildung von FW-spezifischen Aussagen über die Reihenfolge von Interrupts und Schreib-/Leseinstruktionen in HW-spezifische Aussagen über Signale und Takte. Diese Abbildung muss die Übersetzung von Schreib-/Leseinstruktionen in das entsprechende Signalspiel auf dem Bus leisten und Timing-Varianten berücksichtigen, die sich z.B. durch Querverkehr auf dem Bus ergeben können. Das macht die kompositionale formale Verifikation zu einem intellektuellen Kraftakt, den nicht viele Anwender stemmen mögen.
16. HW-Module funktionieren häufig nur dann spezifikationsgemäß, wenn ihre Umgebung gewisse Bedingungen (Constraints) einhält. Derartige Constraints werden bei formaler Verifikation und Constrained-Random-Pattern-Simulation explizit gemacht. Die FW ist verantwortlich für die Einhaltung von Teilen dieser Constraints. Es gibt jedoch HW-Constraints, die durch keinerlei FW-Implementierung eingehalten werden können. Solche Probleme erschweren die Übersetzung von FW-spezifische in HW-spezifische Bedingungen.

3.2.2 Hierarchische Verifikation von Embedded Systems

Probleme ähnlich 1 bis 13 haben beim Entwurf reiner HW oder reiner SW dazu geführt, dass dort hierarchische Verifikationsansätze selbstverständlich sind. In der HW-Verifikation wird mit relativ kleinen Verifikationsobjekten (DUTs) einzelner HW-Module begonnen (im Ggs. zu 1), die unabhängig vom Fortschritt anderer Designteams verifiziert werden können (im Ggs. zu 11). Die Verifikationen werden auf der RTL-Beschreibung vorgenommen, die präzise ist (im Ggs. zu 3) und schnell genug mit einfachen Werkzeugen (RTL-Simulatoren) oder besonders leistungsfähigen Werkzeugen (formale Verifikation) untersucht werden kann, und jedenfalls keine teuren Spezialwerkzeuge erfordert (im Ggs. zu 4). Damit wird die Klasse der funktionalen Fehler stark ausgedünnt (vergl. 5). Dies geschieht zu einem frühen Zeitpunkt (im Ggs. zu 12), zu dem die Beseitigung von HW-Fehlern einfach ist. Die anschließenden Verifikationen auf höheren Hierarchieebenen werden von funktionalen Fehlern seltener gestört (im Ggs. zu 13). Je komplexer die Interaktion zwischen zwei Modulen aus unterschiedlichen Entwurfsteams ist, desto höher ist das Risiko von Integrationsfehlern (vergl. 5) und desto niedriger ist deshalb die Hierarchieebene, auf der ihre Integration verifiziert wird (im Ggs. zu 6), weil eine niedrige Hierarchieebene für gute Controllability (vergl. 9) und Observability (vergl. 10) sorgt und eine schnelle Analyse von Fehlverhalten ermöglicht (vergl. 10). Dadurch werden vorbeugende konstruktive Maßnahmen zur Vermeidung von Integrationsfehlern (vergl. 7) vermieden.

Eine ähnliche Gegenüberstellung läßt sich auch für die hierarchische Verifikation von SW anfertigen.

Der Vorschlag, Embedded Systems hierarchisch zu verifizieren, liegt also eigentlich nahe. Dabei hilft die Einteilung eines HW-Systems in ein Chassis aus CPU, Bus und Speichersystem und in Module mit produktspezifischer Funktionalität („Spezial-HW-Module“, s. Abbildung 2). Die Funktion eines Embedded Systems wird bestimmt von der FW und den Spezial-HW-Modulen. Aufgabe des Chassis ist vor allem, für die semantikkonforme Ausführung der FW zu sorgen und dabei entsprechende Schreib-/Lesezugriffe an die Spezial-HW-Module anzulegen, sowie auf Interrupts zu reagieren.

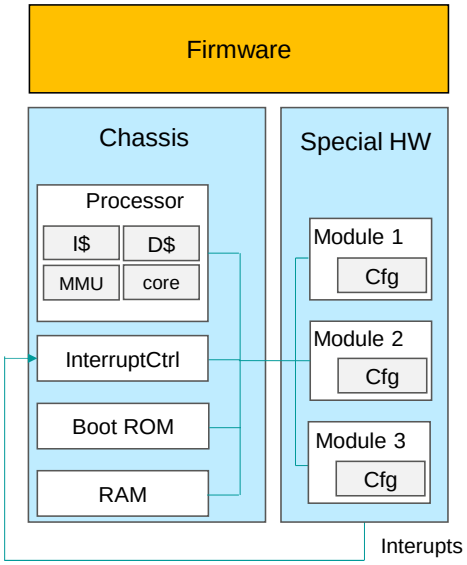

Abbildung 2: Typisches Embedded System

Zur Unterstützung der hierarchischen Verifikation von embedded Systems schlägt SEDA deshalb das Konzept eines „Embedded Modules“ vor, für das ein Prototyp zur formalen Verifikation erforscht wurde.

3.2.3 Embedded Modules

Ein Embedded Module ist ein Spezial-HW-Modul, das um einen Low-Level-Treiber der FW ergänzt ist, der die HW-Funktion über ein API bereitstellt. Dieser Low-Level-Treiber wird integraler Bestandteil des Embedded Modules. In der Spezifikation des Embedded Modules wird vor allem das API festgelegt. Die Funktion soll so spezifiziert sein, dass sie weitgehend durch HW implementiert wird. Der Low-Level-FW-Treiber verbirgt vor allem die Kommunikation mit der HW unter dem spezifizierten API. Eine Verifikation dieses Embedded Modules hat das Ziel, die korrekte Implementierung des APIs nachzuweisen. Sie wird dazu alle FW-Aktivitäten durch API-

Aufrufe darstellen und nicht mehr durch Signal-Sequenzen am Interface des HW-Moduls. Diese Verifikation wird so früh im Entwurfsablauf durchgeführt, dass Fehler im Zusammenspiel zwischen Low-Level-Treiber und HW-Module auch durch HW-Änderungen beseitigt werden können (im Ggs. zu Pt. 9 der Problemanalyse).

Die Entwickler des Embedded Modules können gewisse Funktionalität zwischen HW und Low-Level-FW-Treiber verschieben. Sie können sich für ein billigeres aber schwerer verständliches Signalinterface der HW entscheiden, und gleichzeitig verifizieren, dass die Fallstricke vom Low-Level-Treiber gegenüber den höheren FW-Schichten verborgen werden (vgl. Pt. 7 der Problemanalyse). Das ermöglicht kostenoptimierte HW.

Der Low-Level-Treiber beschränkt das FW-Verhalten, mit dem das HW-Modul angesteuert wird und verringert so möglicherweise sogar die Verifikationsaufwände auf Modulebene gegenüber der heutigen Verifikation von Spezial-HW-Modulen (vgl. 8).

Ferner sollte ein Low-Level-Treiber C/C++-Assertions enthalten, die feuern, wenn die aufrufenden FW-Schichten Benutzungsregeln des APIs verletzen. Derartige Benutzungsregeln ergeben sich häufig aus HW-Constraints (s. 16), die in einer reinen HW-Verifikation auf der Basis des Signalinterface formuliert würden. Für die Verifikation eines Embedded Modules werden sie aber direkt als Assertions im Low-Level-Treiber formuliert und haben damit eine viel leichter nutzbare Form: Die Modulverifikation wird nachweisen, dass das API immer dann korrekt implementiert ist, wenn diese Assertions nicht feuern. In einer Systemsimulation machen solche Assertions Fehlbedienungen des APIs explizit und unterstützen daher die Fehleranalyse (vgl. Pt. 9). Bevor die FW Teil des Produkts wird, sollten diese Assertions wegkonfiguriert werden.

Für eine pre-Silicon-Verifikation des gesamten FW-Systems kann ein Embedded Module durch ein geeignetes Software-Modell ersetzt werden. Auf diese Weise ergibt sich ein natürlicher Einstieg in die hostbasierte Simulation, eine besonders schnelle Variante der Systemsimulation. Wenn dabei die Assertions des vorherigen Abschnitts wiederverwendet werden, können mögliche Fehlbedienungen der HW schon in der pre-Silicon-Verifikation aufgedeckt und vermieden werden.

Die Entwicklung eines Embedded Modules erfordert ein Design-Team, das zusätzlich zur HW auch den Low-Level-Treiber entwickeln kann. Die zusätzliche Kompetenz für die Entwicklung der Low-Level-Treiber kann entweder von HW-Ingenieuren erworben werden oder durch einen FW-Entwickler beigestellt werden. SEDA nimmt an, dass die Low-Level-Treiber syntaktisch und semantisch einfach sein werden und Ähnlichkeiten mit Code für kombinatorische Logik in Verilog oder gar SystemC hat. Trotzdem sollten Verifikationswerkzeuge für Embedded Modules die Designer beim Umgang mit der mglw. ungewohnten FW-Welt unterstützen und z.B. dafür sorgen, dass die Treiber syntaktisch und semantisch korrekt implementiert werden. Die Verifikationsingenieure sparen sich einen Teil der SystemVerilog-Sequenzen in der Testbench, die bei heutiger Verifikation von Spezial-HW-Modulen die FW-Aktivität nachstellen.

3.2.4 Berechnungsmodell für die formale Verifikation von Embedded Modules

Die formale Verifikation von Embedded Modules erfordert vor allem ein Werkzeug zur Modellgenerierung. Dieses Werkzeug muß die HW-Beschreibung und den Low-Level-Treiber in ein gemeinsames Berechnungsmodell übersetzen.

Der Prototyp eines solchen Werkzeugs wurde konzipiert, implementiert, zum Patent angemeldet und Infineon für ihre Forschungsarbeiten im Hinblick auf die Generierung von Eigenschaften zur Verfügung gestellt. Die Patentschrift (Publikationsnummer WO2024/114920 A1) ist diesem

Bericht beigefügt und beschreibt den Prototypen detailliert. Die Patentschrift ist derzeit in Prüfung durch das Europäische Patentamt. Diese Darstellung beschränkt sich auf das Wesentliche.

Das Werkzeug unterstützt Low-Level-Treiber in C oder C++ und Spezial-HW-Module in VHDL, Verilog oder SystemVerilog, die mit der FW durch Schreiben und Lesen von Konfigurationsregistern kommunizieren. Das entspricht der heutigen Entwurfspraxis. Die Low-Level-Treiber rufen Zugriffsfunktionen zum Schreiben und Lesen der Konfigurationsregister auf. Zur Ausführung im endgültigen Produkt werden die Zugriffsfunktionen vom Compiler in Schreib- oder Leseinstruktionen des Prozessors übersetzt. Die Hardware des Prozessors generiert aus diesen Instruktionen Transaktionen auf dem Datenbus. Diese Transaktionen werden vom Datenbus arbitriert und weitergeleitet, bis sie letztendlich das Spezial-HW-Modul erreichen und dort die gewünschten Schreib- oder Leseoperationen auf den Konfigurationsregistern auslösen.

Bei der Konzeption des Werkzeugs zur Modellgenerierung musste vermieden werden, Compiler, Prozessor und Bussystem in das Berechnungsmodell zu integrieren, weil dies zumindest zu übermäßiger Komplexität geführt hätte, die eine formale Verifikation verhindert hätte. Tatsächlich ist für die Verifikation eines Embedded Modules aber auch nur wesentlich, welche Transaktionen durch eine Treiberfunktion ausgelöst werden und welche Berechnungen die Treiberfunktionen intern durchführen. Dafür genügt aber eine Umsetzung entsprechend der C-/C++-Semantik.

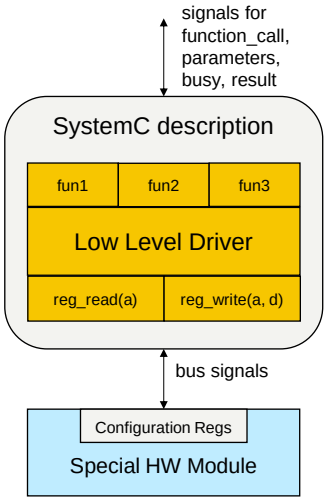

Abbildung 3: Berechnungsmodell

Das Werkzeug generiert aus dem Low-Level-Treiber eine SystemC-Beschreibung (s. Abbildung 3), die synchron mit dem Takt des Spezial-HW-Moduls läuft. Diese SystemC-Beschreibung legt Transaktionen entsprechend der Treiberfunktionen an das Businterface des Spezial-HW-Moduls an. Dabei kann sie alle möglichen Timingalternativen generieren, die sich aufgrund unterschiedlicher Prozessormodelle, Busarchitekturen, oder aufgrund des Datenverkehrs auf dem Bus ergeben können. Zur Steuerung der Verifikation enthält die SystemC-Beschreibung Eingabesignale zur Auswahl von Aufrufen der Low-Level-Treiberfunktionen und zu ihrer Versorgung mit Parametern, sowie Ausgabesignale über Rückgabewerte der Treiberfunktionen und zur Anzeige der andauernden Bearbeitung einer Treiberfunktion.

Die SystemC-Beschreibung wird mit dem Spezial-HW-Modul verschaltet und von dem OneSpin-Werkzeug in ein gemeinsames Berechnungsmodell übersetzt. Dabei wird besonders das bereits bestehende SystemC-Frontend und die Mixed-Language-Fähigkeit des OneSpin-Werkzeugs genutzt.

3.2.5 Formale Verifikation eines Embedded Modules

Mit der oben beschriebenen Modellgenerierung kann die formale Verifikation des Embedded Modules mit einem Werkzeug zur formalen Verifikation von HW-Assertions durchgeführt werden, wie es im OneSpin-Werkzeug zur Verfügung steht. Assertions können in SVA spezifiziert werden. Dabei ist es von Vorteil, dass die Modellgenerierung Signale bereitstellt, mit denen das API des Low-Level-Treibers angesprochen werden kann. Bei allen formalen Beweisen wird vorausgesetzt, dass die Assertions im Low-Level-Treiber nicht feuern, mit denen Fehlbedienungen der HW durch höhere FW-Ebenen abgefangen werden sollen.

SEDA demonstrierte die formale Verifikation an einem Spezial-HW-Modul zur CRC-Kodierung aus dem Sensor-Baustein, den Infineon für die Forschungen zur Verfügung stellte.

SEDA ist noch einen Schritt weitergegangen und hat auch demonstriert, dass Transactional Assertions entsprechend der GapFree-Methodik verwendet werden können. Diese Methodik führt zu einer besonders gut strukturierten, umfassenden formalen Verifikation, die erfolgreich aufgesetzt wurde.

3.3 Durchführung der Arbeiten

B3.3.3: Analyse des Beispiels von IFX

SEDA dankt der Infineon Technologies AG für den safety-kritischen Sensor, der für die VE-VIDES-Forschrungen zur Verfügung gestellt wurde. Der Sensor weist keine Security-Features auf, sodass SEDA sich in Bezug auf Security auf die Verhinderung von Missverständnissen zwischen HW und FW konzentrierte, weil diese security-relevant sein könnten.

SEDA analysierte HW und FW und stellte fristgerecht das Deliverable D03-3B „Erste Resultate der Analyse des IFX-Beispiels für formale HW/SW-Co-Verifikation“ zur Verfügung. Ein Abgleich mit Erfahrungen aus dem Entwurf von Mobilfunkmodems führte zu der oben angeführten Problemanalyse (cf. 3.1).

Das Deliverable D10-3E „Modellierung und Generierung von HW-Eigenschaften“ wurde federführend von Infineon unter Nutzung der SEDA-Beiträge angefertigt.

B3.3.4: Konzeption HW/FW-Verifikation für beide Ansätze

Bei der Abfassung der GVB und TVB wurde befürchtet, dass ein gemeinsames Berechnungs-modell aus HW und unteren FW-Schichten zu komplex werden könnte für eine formale Verifikation. Der Projektvorschlag verfolgte deshalb zwei Ansätze:

1. Formale Verifikation auf einem gemeinsamen Berechnungsmodell
2. Formale Verifikation mit separaten Berechnungsmodellen für HW und untere FW-Schichten, ergänzt um eine Verifikation ihrer Integration

Aufgrund der Problemanalyse wurde die zweite Alternative verworfen, weil

- Selbst die unterste FW-Schicht zu sehr komplexen Verifikationen führen kann (vgl. Punkt 14 der Problemanalyse)
- Eine semantisch korrekte Integrationsverifikation zu schwierig für die Anwendung im Feld ist (vgl. 15).

B3.3.4 wurde deswegen genutzt, um die Methodik der Embedded Modules und der damit zusammengehörenden hierarchischen Verifikation von Embedded Systems zu konzipieren und das Konzept im fristgemäß abgegebenen Deliverable D06-3D „Konzept für die Implementierung der HW-SW-Coverifikationsansätze“ festzuhalten.

B3.3.5 Prototypische Implementierung der hochprioren Erweiterungen am C++/SystemC-Frontend und der HW-Verifikationsumgebung

In diesem Beitrag wurde das SystemC-Frontend und die HW-Verifikationsumgebung erweitert, z.B. um für FW übliche Sprachkonstrukte wie Funktionen und Schleifen besser zu behandeln. Die Entwicklungsarbeiten wurden teilweise in das Jahr 2021 vorgezogen. Infineon erhielt diese Erweiterungen im Rahmen der üblichen Versions-Updates.

B3.3.6 Übersetzer für HW-Randbedingungen in Assertions für die FW

Da in B3.3.4 die formale Verifikation mit separaten Berechnungsmodellen verworfen wurde, war eine explizite Übersetzung von HW-Randbedingungen in Assertions für die FW nicht mehr erforderlich. Statt dessen wurde in Gestalt der C-Assertions im Low-Level-Treiber des Embedded Modules eine Formulierung von HW-Randbedingungen gefunden, die sich ohne Übersetzung elegant wiederverwenden lässt für

- Verifikation eines Embedded Modules
- FW-HW-Systemverifikation
- Hostbasierte pre-Si Systemverifikation
- Pre-Si Systemverifikation mit abstrakten SystemC-HW-Modellen

Deswegen wurde in diesem Beitrag der Prototyp für die Modellgenerierung für die Verifikation von Embedded Modules entwickelt. Dieser Prototyp wurde den Infineon-Kollegen zur Verfügung gestellt, die damit ihre Forschungen zur modellbasierten Generierung von Eigenschaften über FW und HW durchführen. Über den SEDA-Prototyp berichtet das fristgerecht übergebene Deliverable D10-3F „Übersetzer für HW-Constraints in Assertions für die SW“. Der Prototyp wurde zum Patent angemeldet, die Patentprüfung läuft.

B3.3.7 Anwendung der formalen HW/SW Verifikation auf das IFX-Beispiel

Zur Demonstration des Prototypen wurde ein Spezial-HW-Modul zur CRC-Generierung aus dem Sensor durch einen Low-Level-Treiber zu einem Embedded Module erweitert. Dieses wurde vom Prototypen aus B3.3.6 in ein Berechnungsmodell überführt. Auf diesem Berechnungsmodell wurden SVA-Assertions entsprechend der Standard-Methodik assertionbasierter formaler Verifikation bearbeitet. Außerdem wurde eine Variante der GapFree Verification erfolgreich angewandt. Die Resultate sind im Deliverable D10-3F „Übersetzer für HW-Constraints in Assertions für die SW“ dokumentiert.

Das Deliverable D12-3D „Anwendung der formalen HW/SW Co-Verifikation und der Generierung von Eigenschaften auf das Beispiel aus A3.3“ wurde von Infineon während der von Infineon beantragten kostenneutralen Verlängerung des VE-VIDES-Projekts geliefert. SEDA hat auf die Verlängerung verzichtet, aber zu diesem Deliverable in Diskussionen und durch Support des Prototypen auch nach Ende seiner VE-VIDES-Forschungen beigetragen.

3.4 Verwertung und nächste Schritte

Durch die Übernahme der OneSpin Solutions GmbH durch Siemens EDA erfuhr die Aufgabe 3.3 „HW/SW-Koverifikation“ eine Neupriorisierung vor dem Hintergrund ausgereifter Systemsimulationswerkzeuge von Siemens EDA wie z.B. Veloce™ und den Frontends für C, C++ und SystemC, beispielsweise für den Äquivalenzchecker Questa SLEC™ für High-Level-Synthese.

Der Wert der VE-VIDES-Forschungsresultate besteht für Siemens EDA in der Grundlagen-forschung in Richtung auf hierarchische Verifikation von Embedded Systems und den Einsatz formaler Verifikation dabei. Durch diese Grundlagenforschung kann SEDA auf Kundenfragen nach Verbesserungen bei Entwurf und Verifikation von Embedded Systems mit interessanten innovativen Vorschlägen antworten. Solche Fragen werden im Zusammenhang mit den immer größeren SoCs erwartet, die für die Zukunft erwartet werden. SEDA kann dabei sogar mit dem Prototypen argumentieren, der zum Patent angemeldet wurde.

Bei hinreichendem Kundeninteresse werden sich zukünftige Forschungsanstrengungen mit der Übertragung der Ideen auf eine simulationsbasierte Verifikation von Embedded Modules beschäftigen.