# Keine Abkürzung erlaubt

Funktionale Verifikation von IoT-Chips. Um zuverlässige IoT-Produkte entwickeln zu können, ist eine umfassende Verifikation der Chips vonnöten. Abkürzungen können zu schwerwiegenden Fehlern führen, die sich beim Anwender nur mit großem Aufwand beheben lassen.


1 | Verifikationsablauf: Die Überprüfung muss sowohl systematische als auch zufällige Fehler abdecken

Die Annahme, dass die Entwicklung und Verifikation von IoT-Chips um Längen einfacher ist als die für Smartphones, Datenspeicher oder KI-Anwendungen, trifft nur selten zu. Die Mehrheit der IoT-Bausteine basiert auf komplexen SoC-Designs (System-on-Chip) mit eingebetteter Software. Die neuesten Arm- und Intel-Atom-Bausteine, die für das IoT geeignet sind, enthalten mehrere Prozessorkerne und komplexe Spezial-Engines. Dies gilt insbesondere für High-End-Applikationen, etwa in autonomen Fahrzeugen.

Einige Entwickler gehen davon aus, dass die Messlatte für die Überprüfung von IoT-Chips niedriger ist als für andere Arten von ICs. Sie nehmen an, dass sich die betroffene Hardware im Falle eines Fehlers im Feldeinsatz einfach austauschen lässt. Dabei wird jedoch nicht berücksichtigt, dass sich viele der installierten IoT-Systeme an schwer zugänglichen Stellen befinden. Tritt ein Fehler auf, ist es nicht so einfach, eine Sicherheitskamera auf einem schwer zugänglichen Dach zu reparieren, Umgebungssensoren, die in einem Gebäude verstreut sind, auszutauschen oder gar Komponenten in einer aktiven Produktionsstraße zu ersetzen. Daher ist es umso wichtiger, alle Fehler vor der Distribution und Installation innerhalb der Endprodukte zu finden und zu beheben.

Darüber hinaus ist die Annahme falsch, dass versteckte Fehler oder solche, die erst im Feldeinsatz auftreten, für IoT-Applikationen weniger kritisch sind. Schaltungen für Fahrzeuganwendungen haben signifikant höhere Verifikationsanforderungen. Aber auch bei wesentlich simpleren Endgeräten sind die Erwartungen der Verbraucher hoch. Selbst Smartphones müssen immer zuverlässiger sein. In Abwesenheit von Festnetztelefonen können sie im Notfall die einzige Möglichkeit sein, Hilfe zu rufen.

## Verifikation auf der Blockebene

Aus den genannten Gründen ist die Verifikation von IoT-Chips maßgeblich entscheidend für den erfolgreichen Einsatz der Endprodukte. Dabei gibt es sieben wichtige Prozessschritte, denen jeder IoT-Verifikationsingenieur folgen sollte: statische Analyse auf Modulebene, formale Analyse auf Modulebene, Simulation auf Chipebene, formale Analyse auf Chipebene, Equivalence Checking, Sicherheitsverifikation und Hardware-/Software-Co-Verifikation. Diese sieben Schritte sind erforderlich, unabhängig davon, ob der IoT-Baustein auf vollständig spezifischen Schaltungen, ASIC-, oder FPGA-Technologie basiert. Zu beachten ist ebenso, dass moderne FPGAs auch SoC sind, die dieselben Verifikationsanforderungen wie ASICs haben.

Die statische Analyse ist der Ausgangspunkt für neuen RTL-Code, insbesondere auf Modulebene, obwohl ein Großteil dieser Analyse auch auf großen Blöcken und sogar auf vollen Chips ausgeführt werden kann. Ein umfassendes Linting-Tool, das nach häufigen Codierungsfehlern sucht, ist daher unerlässlich. Es kann ganze Klassen einfacher Fehler erkannen und effizient beseitigen. Anwendungsbeispiele sind hierbei die Fehlanpassungen zwischen Synthese und Simulation, Buskonflikte, X-Wert-Ausbreitung, falsche Signalverbindung und Registerimplementierung oder Busprotokolle, die nicht den Spezifikationen entsprechen.

Die weiterentwickelten statischen Überprüfungen erfordern die Verwendung formaler Engines, um eine tiefgehende Analyse zu erreichen. Da diese Prüfung durch Formal-Apps durchgeführt

### FAZIT

Die Verifikation eines IoT-Chips sollte sieben definierten Prozessschritten folgen, unabhängig davon, ob es sich um spezifische ICs oder FPGAs handelt. Formale Methoden kommen dabei sowohl auf Modul- als auch auf Chipebene zum Einsatz. Sie schließen nicht nur systematische sondern auch zufällige Fehler aus. Das Equivalence Checking stellt am Ende sicher, dass die Verifikation das Design nicht auf unbeabsichtigte Weise verändert hat. Das Zusammenspiel von Hardware und Software sollte vor Freigabe in einem Feldtest geprüft werden.

Bild: OneSpin Solutions

wird, ist keine manuelle Spezifikation von Assertions oder genaue Kenntnis von formaler Verifikation erforderlich. Alle benötigten Assertions werden von der App selbst generiert. Die volle Leistungsfähigkeit formaler Methoden wird erreicht, wenn das Entwicklungsteam gezielt Assertion-basierte Verifikation (ABV) anwendet und dabei die Spezifikationen der Assertions einschließt, um das beabsichtigte Verhalten des Designs zu dokumentieren.

**Im Fehlerfall kann es aufwändig sein, ein schwer zugängliches Gerät, wie eine Sicherheitskamera auf einem Dach, zu reparieren**

Viele RTL-Designer tun sich im Schreiben von Assertions leicht, da diese um einiges einfacher sind als die Entwicklung eines objektorientierten Simulationsprüfstands (Testbench). Im Regelfall führen Designer einige formelle Testläufe selbst durch, verlassen sich aber auf das Verifikationsteam, wenn es um Einschränkungen, Abstraktionen und andere Aspekte geht, die ein tieferes Verständnis von formaler Verifikation erfordern. Einige Modultypen werden nur mit formalem ABV verifiziert, ohne zusätzlich Simulationen bis zum Chiplevel zu nutzen. Da Module zu größeren Blöcken zusammengefasst werden, kann die formale Analyse selektiv erneut ausgeführt werden, um zu überprüfen, ob die Interaktion zwischen den Modulen keine Konstruktionsfehler verursacht hat.

## Chiplevel-Verifikation

Auf dem Full-Chip-Level ist die Simulation (Schritt 3 bei der IoT-Verifikation) sehr weit verbreitet. Ein moderner objektorientierter Prüfstand mit eingeschränktem Zufall kann das Design viel gründlicher durchführen als handgeschriebene Tests. Nahezu alle Chiplevel-Simulationen verwenden heute SystemVerilog und halten sich an den UVM-Standard (Universal Verification Methodology). Kommerzielle Verifikations-IP-Anbieter stellen vielfältige Bausteine für die Erstellung des Prüfstands, insbesondere für Onchip-Bus- und externe Schnittstellenstandards, zur Verfügung.

Die formale Analyse spielt auch auf der Chipebene eine wichtige Rolle. Viele Verifikationsarten, die formale Apps normalerweise nur auf kleineren Blöcken durchführen, müssen auf dem gesamten Chip ausgeführt werden, da nur auf dieser obersten Ebene das vollständige Design sichtbar ist. Diese Apps überprüfen die korrekte Signalverbindung, einschließlich Verbindungen zu den I/O-Pads, die ordnungsgemäße Synchronisierung über die Taktdomänenübergänge hinweg sowie zulässige Kombinationen von Stromdomänen, die ein-, ausgeschaltet oder mit reduzierter Spannung oder Frequenz ausgeführt werden. Dabei kommt keine Simulation für diese und andere Tests an die Genauigkeit von formaler Verifikation heran.

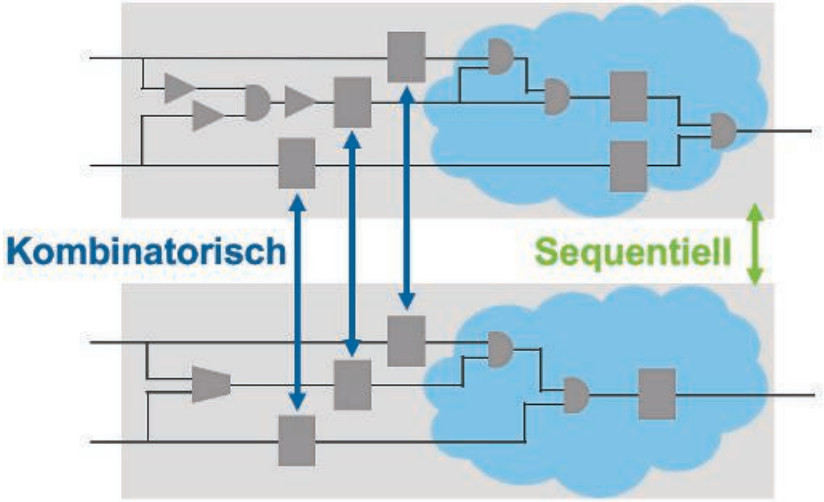
Trotzdem ist die Kombination von formaler Analyse und Simulation immer noch eine Grundvoraussetzung für die funktionale Verifikation eines IoT-Chips. Aus diesem Grund müssen Designer bereits im Planungsprozess entscheiden, welchen Ansatz sie für welchen Teil des Entwurfs verwenden wollen. Die Designanforderungen aus der Produktspezifikation werden einzelnen Merkmalen zugeordnet, von denen jedes mit den gewählten Ansätzen und Werkzeugen verifiziert wird. Die Ergebnisse aller Tools werden kombiniert, was für die Bewertung des Überprüfungsfortschritts und die Entscheidung, was als Nächstes zu tun ist, wesentlich ist.

Zu diesem Zeitpunkt sind die ersten vier Schritte der IoT-Verifizierung abgeschlossen. Idealerweise wurden alle systematischen RTL-Fehler (Designfehler) gefunden und behoben. Aber auch wenn dies zutrifft, ist die Arbeit des Verifikationsteams noch lange nicht abgeschlossen. Viele Chips müssen den funktionalen Sicherheitsanforderungen entsprechen, die in Normen wie IEC 61508 und ISO 26262 (**Bild 1**) festgelegt sind. Selbst wenn keine Konstruktionsfehler mehr vorhanden sind, muss ein IC auch auf zufällige Fehler, wie ein Alphapartikel, das ein Speicherbit umdreht, geprüft werden.

Um dieser fatalen Eventualität vorzubeugen, kann ein IoT-Chip Hardware-Sicherheitsmechanismen enthalten, die erkennen, wenn ein Zufallsfehler die Ausgabe des Designs geändert hat. Es muss mindestens ein Alarm ausgelöst werden, damit passende Korrekturmaßnahmen ergriffen werden können. Beispielsweise könnte ein autonomes Fahrzeug sicher an den Straßenrand heranfahren und das

**KONTAKT**

OneSpin Solutions GmbH,
Nymphenburger Straße 20a,
80339 München,
Tel. 089 990130,
E-Mail info@onespin-solutions.com,
www.onespin-solutions.com




2 | Equivalence Checking: Diese formale Verifikationsmethode überprüft, ob die Designfunktionen verändert worden sind

Bild: OneSpin Solutions

System in der Hoffnung auf Behebung des Problems neu starten. Noch besser ist es, wenn das fehlerhafte Verhalten korrigiert werden kann und der Fehler anschließend zur Diagnose und Analyse protokolliert wird.

Bei der Sicherheitsverifikation spielt die formale Analyse eine Schlüsselrolle. Bei einem geeigneten Fehlermodell können Fehler eingefügt und analysiert werden, um zu sehen, ob sie sich zu einem Ausgang ausbreiten. Eine weitere Analyse bestimmt, ob der Hardware-Sicherheitsmechanismus diese propagierten Fehler erkennt und der prozentuale Anteil des Entwurfs, der gegen zufällige Fehler geschützt ist, berechnet werden kann. Einige Sicherheitsstandards erfordern ein hohes Maß an Schutz, daher ist ein robuster und automatisierter Analyseprozess unerlässlich.

## Verifikation auf der Systemebene

Schritt 6 der IoT-Verifikation berücksichtigt ein anderes Problem: Fehler, die durch Testeinfügungswerkzeuge, Logiksynthese oder den Place-And-Route-Prozess in das Design eingeführt wurden. Die schrittweise Simulation des RTL-Designs und der endgültigen Netlist nebeneinander ist langsam, teuer und kann Fehler übersehen. Nur ein umfassendes formales Equivalence Checking (**Bild 2**) kann sicherstellen, dass die Tools zur Überprüfung das Design nicht auf unbeabsichtigte Weise verändert haben.

Der letzte Schritt der IoT-Verifikation besteht darin, die Hardware und Software zusammen in einer Umgebung auszuführen, die einem realen Feldeinsatz möglichst nahekommt. Dies wird normalerweise durch Abbilden des Chipdesigns in einen In-Circuit-Emulator oder ein FPGA-basiertes Prototyping-System erreicht, bei dem die Produktsoftware auf den eingebetteten Prozessoren ausgeführt und das gesamte System in den Sockel eingesteckt wird, der am Ende den gefertigten Chip hält. Diese Art der Co-Verifikation von Hardware und Software läuft wesentlich schneller als jede Simulation.

### WISSENSWERT

Funktionen bleiben erhalten. Um maximale Leistung und Kapazität aus der zugrundeliegenden Technologie herauszuholen, führen FPGA-Entwurfsprozesse Optimierungen durch, die die innere Struktur des Entwurfs ändern, zum Beispiel durch das Umordnen von Zustandsmaschinen oder Verschieben von Logik auf eine Seite oder eines Registers auf die andere. Nur das formale, sequenzielle Equivalence Checking kann sicherstellen, dass dabei die Gesamtheit der Funktionen erhalten bleibt. Es kann zu jedem Zeitpunkt im Entwurfsprozess oder im initialen RTL und der platzierten Routing-Netzliste durchgeführt werden.

## Ein IoT-Anwendungsbeispiel

Ein Beispiel für die Anwendung der oben genannten sieben Schritte ist ein vernetztes Heimsicherheitssystem. Es gibt verschiedene Szenarien, die auftreten können, sollte das System sich aufhängen. Beispielsweise frieren alle Schlösser ein, das Alarmsystem springt an oder geht aus und Sensoren senden keine Signale mehr. Statische Analyse kann Zustandsmaschinen-Deadlocks erkennen, die diese höchst unerwünschten Bedingungen verursachen könnten. Assertions erfassen komplexere Szenarien, wie ein Alarm, der nach Schließen der Tür erneut scharf geschaltet werden muss. Die formale Analyse kann Fehler finden, die gegen diese Assertions verstoßen, und sie werden dann als gültig befunden, wenn alle Fehler behoben sind.

Die Simulation auf Chipebene ist die Hauptmethode, um alle im Verifikationsplan angegebenen Funktionen durchzuarbeiten und deren Funktionsfähigkeit zu überprüfen. Beispiele sind das Auslösen eines Alarms bei geöffneter Tür, das Unterdrücken des Alarms bei Eingabe des richtigen Sicherheitscodes und das Auslösen des Alarms bei einem Timeout oder einem falschen Code. Eine formale Analyse auf Chipebene kann sicherstellen, dass die Module und I/O-Pads unter allen Bedingungen ordnungsgemäß angeschlossen sind und beispielsweise keine Konfiguration vorhanden ist, die das Pad und die Stifte, die zum Türöffnungssensor führen, trennen würde.

Das Equivalence Checking stellt sicher, dass das beabsichtigte Design ordnungsgemäß in der ausgewählten ASIC- oder FPGA-Technologie implementiert wird. Dies schützt vor Fehlern in Synthese- oder Optimierungswerkzeugen und überprüft, ob absichtlich schädliche Funktionen eingefügt wurden. Die Sicherheitsverifikation stellt sicher, dass die kritische Logik auch dann funktioniert, wenn zufällige Fehler im Feld auftreten. Schließlich führt die Hardware-/Software-Co-Verifikation Produktionscode auf den eingebetteten Prozessoren aus, um das gesamte System zu testen. Die Kombination aus formaler Verifikation und umfassender Simulation realer Szenarien macht es viel wahrscheinlicher, dass das endgültige Sicherheitssystemprodukt unter allen Bedingungen ordnungsgemäß funktioniert.

mey

**Autor**

Tom Anderson ist technischer Marketingberater bei OneSpin Solutions.

**Online-Service**

Interview „Fehler aufdecken und Reparaturen vermeiden“ in elektronik informationen 9/2018

www.elektronik-informationen.de/76018