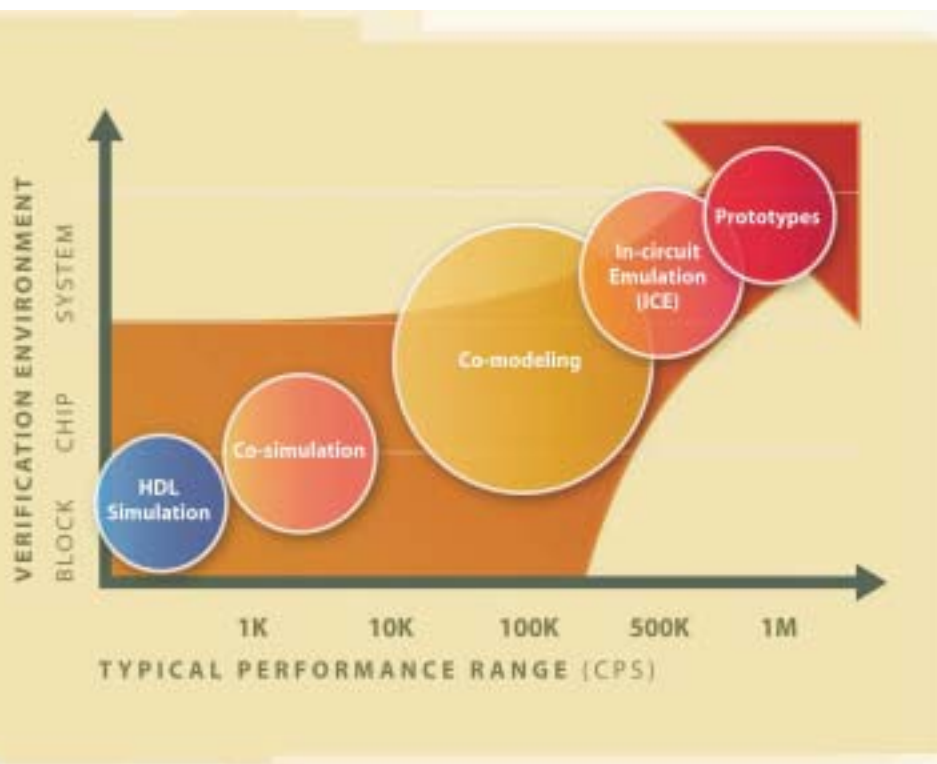


Auswege aus dem Engpass der Verifikation

Die vielfachen Dimensionen der Skalierbarkeit einer Systemverifikationsstrategie



Eine skalierbare Verifikationsumgebung umfasst den kompletten Entwurfsprozess

Die funktionale Verifikation ist heute eine der größten Herausforderungen für Elektronik-Designer, da die Komplexität der Gesamtsysteme wächst und Produkte mit immer größerer Funktionalität ausgestattet werden. Das Problem der bestehenden Entwurfs- und Verifikationsmethoden liegt darin, dass die Verifikation dem Entwurf gegenüber eine zweitrangige Rolle spielt. Dies muss sich ändern, denn die funktionale Verifikation, als der zum

Auffinden von Funktionsfehlern verwendete Prozess, stellt den größten Engpass im Ablauf des Entwurfsprozesses dar. MARTIN REUTER

Mit den Jahren sind Methoden zur Umgehung der größten Engpässe im Entwurfsprozess entwickelt worden. Im Mittelpunkt standen dabei zumeist der Umgang mit Komplexität und das Erfordernis einer Aufteilung des Entwurfsprozesses in eine Anzahl von übersichtlicheren Abschnitten. Um dies zu ermöglichen, wurde von den meisten Unternehmen eine Variante des klassischen V-Diagramms verwendet, wie es in der Abbildung

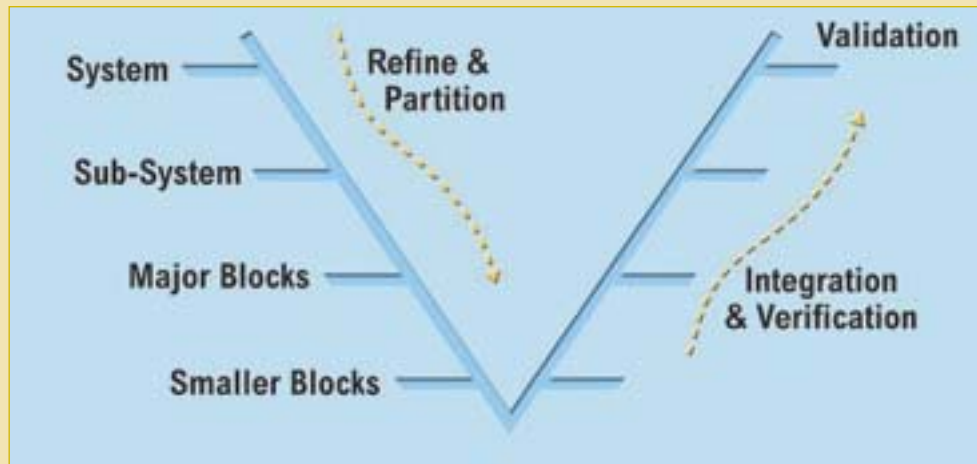
auf Seite 54 zu sehen ist. Die Bezeichnungen in diesem Diagramm und die Anzahl der verwendeten Ebenen sind zufällig gewählt. Diese Strategie stellt einen Divide-and-Conquer-Ansatz dar, bei dem an der Basis des V Blöcke auf RTL-Ebene implementiert werden. Die rechte Seite des Diagramms zeigt die Verifikationsstrategie. Diese beginnt mit der Verifikation der einzelnen Blöcke, die, nachdem alle Fehler gefunden wurden, mit den Blöcken anderer Designer integriert werden, um ►



MARTIN REUTER ist Technical Director Central Europe bei Mentor Graphics Deutschland GmbH

KONTAKT
T +49/89/57096-0
martin_reuter@mentor.com

Klassischer V-Prozess
in der Verifikation



größere Subsysteme zu erstellen. Die Verifikation erfolgt anschließend auf dieser Ebene. Dieser Prozess wird fortgesetzt, bis alle Blöcke zum kompletten System zusammengesetzt wurden.

Späte Überprüfung der Spezifikation

Bei vielen Unternehmen besteht erst an diesem Punkt der vollständigen Systemintegration die Möglichkeit, die Spezifikation selbst zu überprüfen. Dieser Prozess wird als Validierung bezeichnet. Bis dahin erfolgt jegliche Verifikation allein um sicherzustellen, dass die Implementierung mit der Spezifikation übereinstimmt, nicht zur Verifikation der Spezifikation selbst. Die zunehmende Bedeutung der funktionalen Verifikation ergibt sich aus dem Wachstum bei Entwurfsgröße und Komplexität, zu dem der höhere Anteil von Software und Analogverarbeitung in gemischten Entwürfen beiträgt. Die zunehmende Größe bezieht sich auf die enorme Anzahl von Transistoren und Gattern in System-on-Chip-Entwürfen. Nach Einschätzung der International Technology Roadmap for Semiconductors von 2001 werden System-on-Chip-Lösungen im Jahr 2006 eine Milliarde Transistoren enthalten. Ein System-on-Chip kann bereits aus mehreren zehn Millionen Gattern bestehen, so dass ein entsprechendes Fehlerpotenzial entsteht und die Verifikation komplexer wird. Der verstärkte Einsatz von Software und Analogverarbeitung auf dem Chip trägt nicht nur zur Systemkomplexität bei, sondern stellt außerdem die traditionelle Herangehensweise in Frage. Auf Digitaltechnik spezialisierte Ingenieure sehen sich mit unvertrauten Problemen der Analogtechnik konfrontiert. Bei vielen Hardwareentwürfen ist es erforderlich, dass die Firmware oder maschinennahe Software einsatzbereit ist, damit die RTL-Funktionalität verifiziert werden kann. Dazu müssen Firmwaredesigner eine wichtige Rolle beim Entwurf von Hardware übernehmen und das

Zusammenspiel von Hardware und Software berücksichtigen.

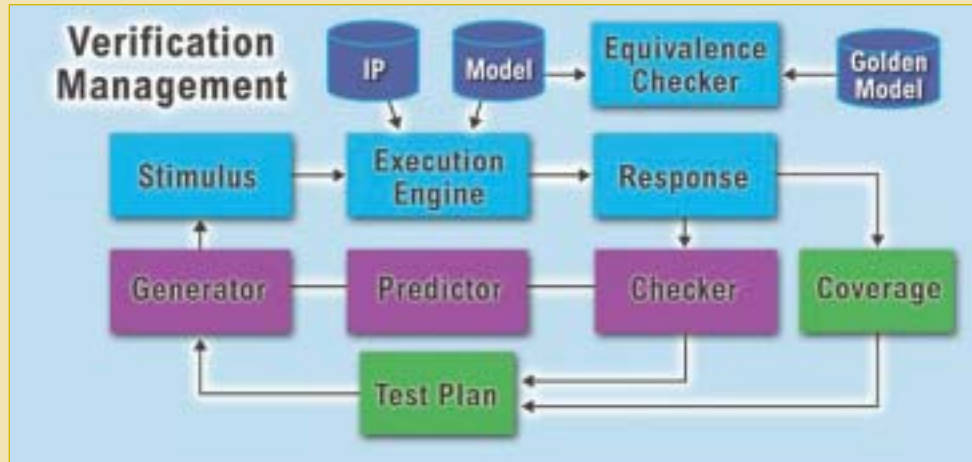
Möglichkeiten der Skalierbarkeit

Die Verifikationslösung sollte ein Paket von Tools für den vollständigen Ablauf von der Simulation in Hardwarebeschreibungssprache (HDL) bis hin zur In-Circuit-Emulation enthalten. Das bedeutet bessere Simulatoren und Emulatoren zur Beschleunigung des Verifikationsprozesses auf allen Integrationsebenen. Eine toolübergreifende Skalierbarkeit ist notwendig, weil verschiedene Verifikationsverfahren zu unterschiedlichen Lösungen in unterschiedlichen Leistungsbereichen führen. Bei jeder Lösung müssen Kompromisse zwischen Iterationszeit, Leistung, Kapazität, Sichtbarkeit bei der Fehlerbeseitigung und Kosten gefunden werden. So wird zur Verifikation von Architekturentscheidungen in Bezug auf das System kein HDL-Softwaresimulator verwendet, sondern ein abstraktes Modell oder eine Hardware-Software-Umgebung auf Transaktionsebene, mit der sich die nötigen Informationen ermitteln lassen. Andererseits wäre eine In-Circuit-Emulation nicht die angemessene Lösung zur Verifikation kleiner Unterblöcke eines Chipentwurfs, wenn dieselbe Aufgabe mit einem HDL-Softwaresimulator schnell und einfach erledigt werden könnte. Abgesehen von der Möglichkeit der Verwendung verschiedener Tools ist auch das Maximieren der Produktivität der Tools selbst ein wichtiger Punkt. Der Verifikationsprozess kann auf diese Weise innerhalb einer einzigen Umgebung durchgeführt werden, bis es absolut unumgänglich ist, zu einer anderen Lösung zu wechseln. Diese Skalierbarkeit innerhalb der Tools kann sich auf unterschiedliche Weise zeigen. So muss bei Regressionstests eine große Zahl von Tests sehr häufig ausgeführt werden. Es ist unwahrscheinlich, dass ein einzelner Simulator über genügend Leistung verfügt, um eine Aufgabe dieser Größe in angemessener Zeit zu bewältigen. Durch eine „Simulations-

Farm“, wo sich mehrere Jobs in eine Warteschlange einreihen lassen und auf jeweils verfügbaren Rechnern ausgeführt werden, sind Regressionstests einfacher und gleichzeitig besser durchführbar. Sind sehr lange Ausführungszeiten Teil der Tests im Regressionspaket, ist möglicherweise eine Emulation erforderlich. Ein einzelner Emulator ist schon an sich äußerst skalierbar, da seine Kapazität einfach an verschiedene Entwurfsgrößen angepasst werden kann. Gegebenenfalls kann die Kapazität erweitert werden, indem zwei oder mehrere Emulatoren zusammen angeschlossen werden.

Funktionale Verifikation in der Anfangsphase

Langfristig müssen einige Aspekte der funktionalen Verifikation in die Anfangsphasen des Entwurfsprozesses verschoben werden. Dazu muss die Verifikation abstrakter werden, indem Modelle und Transaktoren auf höheren Ebenen verwendet werden. Das Verschieben der Verifikation im Entwurfsprozess bringt mehrere bedeutende Vorteile mit sich. Modelle können in diesem Stadium sehr viel schneller geschrieben werden, haben einen größeren Durchsatz und können daher konstruktiv für Entwurfsentscheidungen genutzt werden. Früher oder später ist der Punkt erreicht, an dem abstraktere Darstellungen des Entwurfs absolut notwendig sind. Das gilt sowohl für den Entwurf als auch für die Testumgebung. Das Erstellen dieser Prototypen auf hoher Ebene in Sprachen wie C, C++ oder SystemC ermöglicht die unmittelbare Verifikation der getroffenen Architektur- oder Partitionierungsentscheidungen. Auch herkömmliche HDL-Sprachen wie Verilog und VHDL oder das neu entwickelte SystemVerilog können für Ebenen oberhalb von RTL effektiv verwendet werden. Bei der Partitionierung des Systems in Hardware- und Softwareblöcke oder bei der Verfeinerung der Entwurfshierarchie können Verifikationstools zur Unterstützung der ver-



Voraussetzung für die vollständige Verifikation des Systems ist eine durchdachte Verifikationsstrategie

wendeten Schnittstellen eingesetzt werden, sodass jeder Block verarbeitet werden kann, ohne dass gewartet werden muss, bis alle Blöcke dieselbe Abstraktionsebene erreicht haben, um sie zu verifizieren. Damit eine Abstraktionsstrategie mit mehreren Ebenen funktioniert, müssen Technologie und IP-Komponenten kombiniert werden. Die Modelle, mit denen Designer die Abstraktionsebenen wechseln und die Ebenen verknüpfen können, spielen eine wesentliche Rolle. Die hierarchische Verifikation wird durch eine Gruppe von Transaktoren für die grundlegenden Schnittstellen eines Entwurfs ermöglicht. Auf diese Weise können gemischte Entwurfsbeschreibungen auf verschiedenen Abstraktionsebenen verwendet werden.

Verbesserte Lösungen für die Fehlerbeseitigung

Eine Scalable Verification-Lösung muss über integrierte Tools zur Fehlerbeseitigung verfügen, die über unterschiedliche Abstraktionsebenen und Skalierbarkeitstools konsistent sind. Damit sollen schneller Fehler erkannt, Ursachen festgestellt und Probleme behoben werden können, um die Iterationsschleifen auf ein Minimum zu reduzieren. Die Fehlerbeseitigung auf Systemebene wird durch mehrere Abstraktionsebenen und unterschiedliche Semantiken innerhalb eines Systems kompliziert. Dies stellt insbesondere in heterogenen Umgebungen wie Hardware/Software- oder Digital/Analog-Umgebungen eine größere Herausforderung dar. Daher müssen Daten nicht nur verfügbar gemacht werden, sondern es muss auch der korrekte semantische Kontext gegeben sein. Bei der Software-Fehlerbeseitigung sind beispielsweise alle Informationen zur Programmausführung im Arbeitsspeicher der Hardware enthalten, und doch sind keine dieser Daten unmittelbar verfügbar. Das Wissen, wo sich eine Variable befindet, steht nur am Anfang der Lösung. Es muss außerdem bestimmt werden, auf welchem

Chip und an welcher relativen Adresse auf dem Chip sich die Daten befinden – vorausgesetzt, sie befinden sich nicht im Cache oder in einem Register. Darüber hinaus befinden sich in vielen Fällen aufgrund der Verschachtelung von Daten oder Adressen usw. die Daten nicht in logischer Ordnung auf dem Chip. Als Reaktion auf einige dieser Schwierigkeiten werden häufiger neue Methoden der Fehlerbeseitigung eingesetzt, z.B. Assertions und Checker, doch werden sie noch nicht in vollem Umfang genutzt. Ein weiterer Bereich, in dem Unklarheit herrscht, betrifft die Coverage. Vielen Ingenieuren ist nicht bewusst, dass zufrieden stellende Maßzahlen bei der Code-Coverage noch keine adäquate Verifikation des Systems bedeuten. Zusätzliche Maßzahlen wie funktionale Coverage oder Assertion-Coverage müssen eingesetzt werden, um eine vollständige Verifikation des Systems zu gewährleisten.

Testumgebungen werden durch zwei unabhängige Faktoren beschränkt: Steuerbarkeit und Beobachtbarkeit. Die Steuerbarkeit kann mit der Fähigkeit einer Testumgebung gleichgesetzt werden, einen Fehler des Entwurfs durch Einführen von Eingangsdaten zu aktivieren. Dieser Vorgang steht in sehr enger Beziehung zum Maß der Code-Überdeckung. Daher muss die Code-Überdeckung mit Sorgfalt eingesetzt werden, da hierbei die anderen Aspekte des Verifikationsproblems nicht in Betracht gezogen werden. In Bezug auf Beobachtbarkeit müssen nach der Ausführung des Fehlers zwei Dinge geschehen. Zunächst muss ein unerwünschter Effekt des Fehlers an einen primären Ausgang weitergeleitet werden. Anschließend wird die Differenz zwischen den gewünschten und den unerwünschten Effekten festgestellt. Assertions wirken sich hier positiv auf die Beobachtbarkeit aus. Mit ihnen lassen sich zudem die primären Ursachen der Fehlfunktion erkennen, sodass die Fehlerbeseitigung viel einfacher und schneller vonstatten gehen kann. Dies liegt daran, dass virtuelle primäre Ausgänge erstellt werden können, die automatisch auf erwünschtes oder unerwünschtes Verhalten prüfen. In der

Testumgebung müssen daher die fehlerhaften Effekte nicht bis zu den tatsächlichen primären Ausgängen weitergeleitet werden. Die Entwicklung der Testumgebungen wird dadurch vereinfacht. Darüber hinaus werden große Mengen von Daten verifiziert, die andernfalls ignoriert worden wären.

Schlussfolgerung

Entwurfsteams haben im Prinzip zwei Möglichkeiten, vorhandene Methoden zu verbessern. Sie können Tools verwenden, die sich an die Komplexität von Entwürfen anpassen lassen, und sie sollten mehrere Abstraktionsebenen verwenden. Mit einer skalierbaren Lösung können Ingenieure ihre Arbeit besser und schneller erledigen und dabei häufiger im Zeitrahmen bleiben.

Die Verifikationstools werden benutzerfreundlicher, und es können mehr Vektoren in einen Entwurf eingeführt werden. Jede effektive Systemverifikationsstrategie sollte wirklich das gesamte System betrachten und nicht nur die digitale Hardware umfassen. Mit anderen Worten: Für eine sinnvolle Lösung müssen Analogverarbeitung, Softwarelösungen, Echtzeitbetriebssysteme sowie die Umgebung, in der diese betrieben werden sollen, berücksichtigt und zu einem einheitlichen System zusammengefasst werden. Darüber hinaus ermöglichen Design-for-Verification-Techniken eine bessere Wiederverwendung der erstellten Verifikationskomponenten. ■

Dieser Beitrag als PDF und weiterführende Informationen (ähnliche Beiträge, technische Daten, Direktlinks zum Hersteller etc.) sind online verfügbar auf www.duv24.net

more @ click DV044252 >