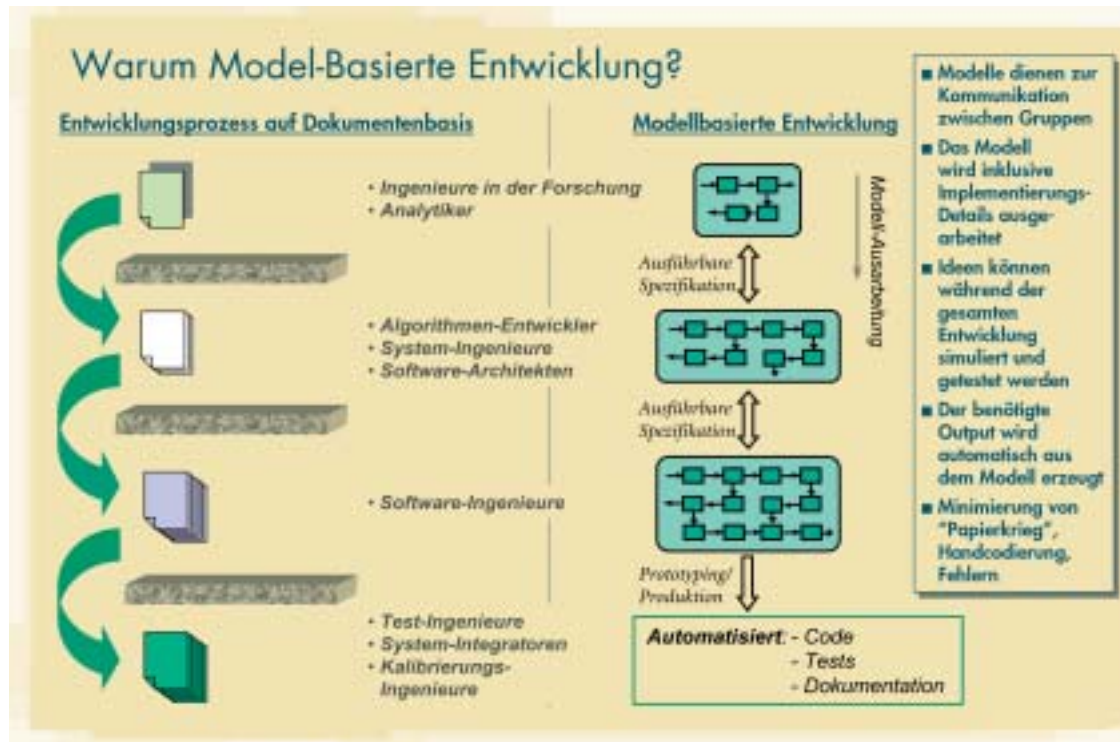


# Effiziente Entwicklung

## Modellbasierte Entwicklung für digitale Signalverarbeitungssysteme

Die Entwickler von digitalen, signalverarbeitenden Systemen sehen sich der Forderung ausgesetzt, immer komplexere und leistungsfähigere Produkte in immer kürzerer Zeit marktreif zu entwickeln. Ein wichtiger Erfolgsträger sind dabei Entwicklungswerkzeuge, die ein Abstraktionsniveau erreichen, das den aktuellen technischen Möglichkeiten der Hardware gerecht wird.

HANS MARTIN RITT



Der modellbasierte Entwicklungsprozess



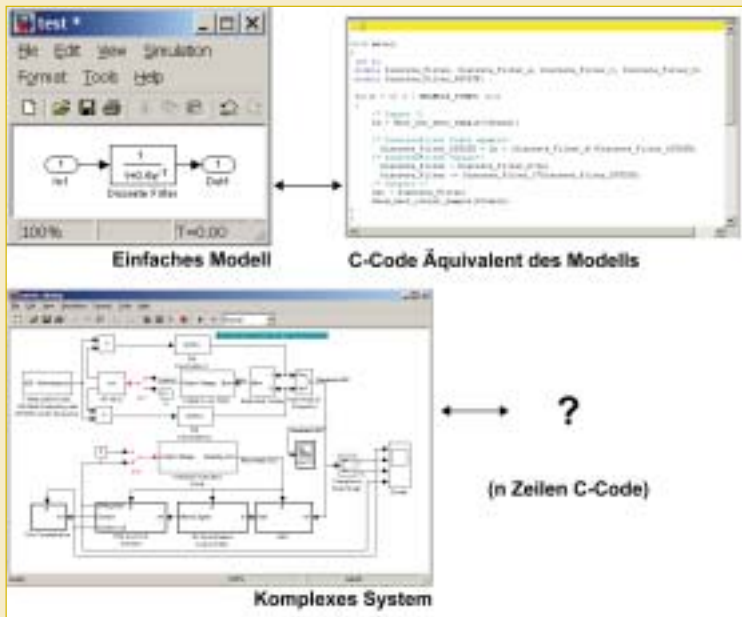
Dr.-Ing. HANS MARTIN RITT  
ist Application Engineer DSP-  
Communications Industry  
bei The MathWorks GmbH

**KONTAKT**  
T +49/241/47075-0  
m.ritt@mathworks.de

**T**ypisch für die Entwicklung der Programmierwerkzeuge ist eine immer höhere Abstraktion. Die Sprache C beinhaltet beispielsweise je Codezeile mehr ausführbare Anweisungen als Assemblercode. Ebenso stellen VHDL und Verilog digitale Hardware erheblich kompakter dar als frühere Werkzeuge auf dem Logik-Niveau. Allerdings wurden diese Abstraktionen erst durch eine effiziente Technologie zur Codegenerierung für die allgemeine Entwicklung praktikabel: C-Compiler bzw. die logische Synthese. Diese wurden jedoch erst dann allgemein anerkannt, als die Komplexität der Entwicklungsprobleme ausreichend hoch wurde und die Ingenieure eine Automatisierung dieses Prozesses akzeptierten.

### Modellbasierte Entwicklung

Über Jahrzehnte hinweg wurde für Spezifikationen in Lehrbüchern der Nachrichtentechnik und auf den Reißbrettern der Ingenieure mit Blockdiagrammen gearbeitet, um den Signalfluss, das Timing und die Systemarchitektur darzustellen. Es überrascht daher nicht, dass grafische Werkzeuge die natürliche Art sind, signalverarbeitende Systeme zu entwickeln und zu verifizieren. Hierarchisch angeordnet, spiegeln sie die Systemarchitektur verständlich wider und repräsentieren auf natürliche Weise die tatsächlichen Hard- und Softwarekomponenten und Algorithmen. Eine ideale Entwicklungsumgebung sollte in genau dieser Art in der Lage sein, Algorithmen und Architekturen auf einem angemessenen Abstraktionsniveau



Die signalorientierte, grafische Darstellung reduziert Komplexität

handhabbar zu machen. Sie muss sich an der Darstellung der Signalflüsse im zu entwickelnden System orientieren – diese visualisieren. Dieses ist auch die Grundlage für den modellbasierten Ansatz bei der Entwicklung von signalverarbeitenden Systemen: Die „Idee“ wird in einem grafischen Modell abgebildet, das auch schon sehr früh in der Konzeptphase durch Simulation verifiziert werden kann. Auch andere Elemente des Systems und der Umgebung, in der dieses arbeiten wird, wie z. B. Störungen, können dabei modelliert und die Auswirkungen analysiert werden. Durch kontinuierliches Verfeinern, Simulieren und Testen des Systemmodells entwickelt sich das „Produkt“ von der idealisierten Spezifikation bis hin zu einer bitgenauen und das Timing exakt wiedergebenden Darstellung. Aus dem Modell kann dann durch automatische Codegenerierung Software für das Zielsystem bzw. eine Hardware-Implementierung erzeugt werden. Das Modell wird also zu einer Art „ausführbaren“ Spezifikation.

### Struktur

Der Ansatz erhöht also nicht einfach das Abstraktionsniveau der Entwicklung, sondern strukturiert den Entwicklungs- und Verifikationsablauf für solche programmierbaren Systeme neu. Die Schlüsselemente dieses Ablaufs sind die automatische Codegenerierung aus dem grafischen Systemmodell sowie Verifikationsschnittstellen zu Entwicklungswerkzeugen niedrigerer Ebene auf Hardware- und Softwarebasis. Die Codegenerierung ermöglicht damit das Echtzeit-Prototyping des Entwurfs auf der Target-Hardware und Hardware-in-the-Loop-Tests zur Verifikation der Implementierung unter realen Bedingungen. Gleichmaßen wichtig ist, dass der erzeugte Code es erlaubt, die Entwicklungsumgebung in gängige Verfahrensweisen zur Entwicklung von Software und Hardware einzubinden. Diese Herangehensweise beseitigt Fehler früher im Entwicklungsprozess und verringert somit das

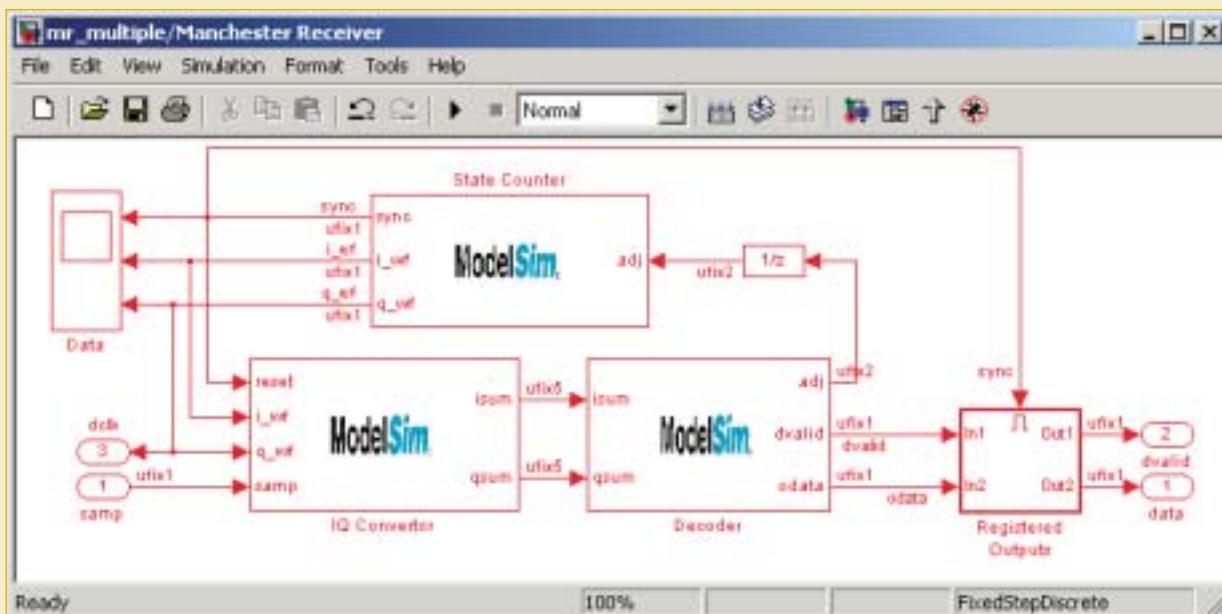
Risiko von Fehlentwicklungen und verfehlten Terminen. Außerdem bietet sie den Vorteil, nicht auf eine einzige konkrete Implementierung festgelegt zu sein. Verändern sich die Hardware-Plattformen, kann der Entwickler die nötigen Modifikationen an den relevanten Modulen des Modells vornehmen, ohne die Kernfunktionalität anpassen zu müssen.

### Effizienzsteigerung

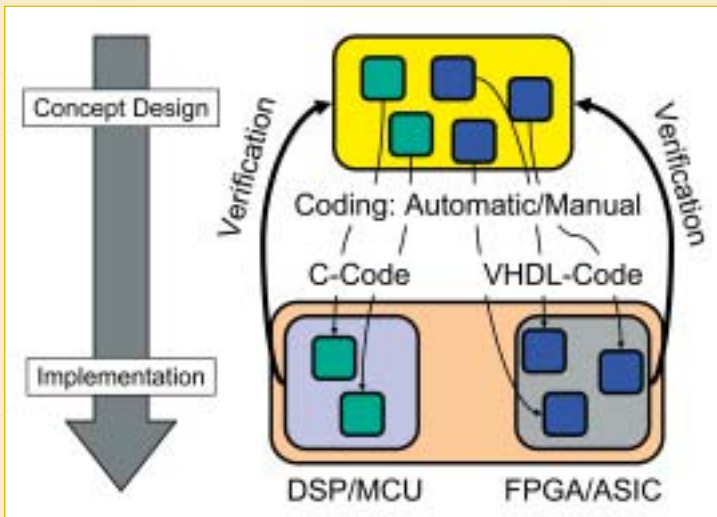
Die Entwicklungsumgebung Matlab/Simulink implementiert die Methode „Modellbasierte Entwicklung“, wobei Simulink als grafisch orientiertes Werkzeug die Kernanwendung für die Entwicklung eines Produktes als „Modell“ darstellt. In dieser Umgebung ist eine erste „Idee“ skizzierbar und kann bereits in frühem Stadium durch Simulation getestet werden. Aus dieser Skizze entsteht durch schrittweise Detaillierung und anschließende Code-Generierung mit dem Real-Time Workshop das endgültige Produkt. Wichtigster Unterschied zur Programmierung einer signalverarbeitenden Anwendung in einer herkömmlichen Programmiersprache oder Hardwarebeschreibungssprache ist die Darstellung der Signalzusammenhänge, ergänzt um Informationen über die Eigenschaften der Signale, wie z. B. die Abtastrate. Schlüssel für die deutliche Effizienzsteigerung ist, dass die für die koordinierte Berechnung von Einzel-funktionen notwendige Analyse und Ordnung bei Simulink von dem Simulations-Engine gewährleistet und später auch in dem Code abgebildet wird.

### Arbeitsweise

Für die Strukturierung der Darstellung von komplexen Systemen können Gruppen von ►



Verifikation von VHDL-Code in Simulink



Top-Down Entwurf mit flexibler Partitionierung der Implementierung

Blöcken zu Subsystemen zusammengefasst werden. Damit implementiert Simulink die Forderung, neben den Algorithmen auch die Architektur eines Produktes zu visualisieren. Durch Auslagerung von Subsystemen in eigenständige Files – so genannte Bibliotheken – kann die Arbeitsweise weiter strukturiert, die Wiederverwendung vereinfacht und die Zusammenarbeit von mehreren Personen an einem Projekt formalisiert werden. Basierend auf dieser Methodik können umfangreiche, wiederverwendbare Baukastensysteme auf beliebigen Abstraktionsniveaus aufgebaut werden. In derselben Denkweise implementiert, sind viele Standardfunktionen bereits durch kommerziell verfügbare „Blocksets“ abgedeckt, die einen zusätzlichen Effizienzvorteil entfalten. Typisch für

den Ansatz der modellbasierten Entwicklung ist der späte Zeitpunkt der Spezifizierung einer Zielplattform. Erst am Schluss der Entwicklung muss, z. B. nach Abschätzung der notwendigen Rechenkapazitäten und des Speicherbedarfs, die konkrete Festlegung erfolgen. Dann ersetzen im Modell Referenzen auf Funktionen, beispielsweise für den Datenaustausch auf der Zielplattform, die Abbildungen der Umgebungseigenschaften. Ist in der Darstellung modularisiert gearbeitet worden, ist es auch unproblematisch, die Entscheidung während der Entwicklung zu revidieren oder die Implementierung eines Prototyps auf einer anderen Hardware vorzunehmen und später zu wechseln bzw. hin und her zu wechseln. Die Auswirkungen von Veränderungen, wie zum Beispiel die Änderun-

gen der eingesetzten Zahlenformate, in der geschlossenen Umgebung können dabei analysiert und gezielt optimiert werden.

## Funktion

Die Basis-Funktion der C-Code-Generierung ist in dem Modul „Realtime-Workshop“ zusammengefasst, das standardmäßig Ansi-C-Code erzeugt. Mit dem so genannten Target-Language-Compiler kann der erzeugte C-Code durch Beschreibung der jeweiligen Besonderheiten an jede Zielhardware angepasst werden. Dafür können z. B. mit der Hardware gelieferte Bibliotheken in den Build-Vorgang eingebunden werden. Auf der Simulink-Oberfläche wird die Funktionalität der Plattform abstrahiert, indem Platzhalterblöcke Funktionen aus solchen mitgelieferten Bibliotheken repräsentieren.

Für die Erzeugung von PLD-Anwendungen, bzw. insbesondere FPGA-Implementierungen stehen spezielle Entwicklungswerkzeuge der entsprechenden Baustein-Hersteller zur Verfügung. Dabei werden Logik-Blöcke mit hinterlegtem VHDL-Code in Form von Simulink-Blöcken bereit gehalten, die eine direkte Abbildung, Simulation und anschließende Systemgenerierung des Logiksystems in bzw. von Simulink aus ermöglichen. Außerdem stehen die Werkzeuge am Anfang einer Entwicklung, an deren Ende VHDL-Code direkt aus Simulink-Modellen generiert werden kann. Realität ist heute schon die direkte Verifikation von MATLAB/Simulink-Modellen gegen manuell implementierten VHDL- bzw. Verilog-Code durch Co-Simulation mit dem Design-Werkzeug ModelSim.

## Zusammenfassung

Der modellbasierte Entwicklungsansatz etabliert sich als Antwort auf die zunehmende Komplexität bei der Entwicklung von signalverarbeitenden Systemen. Der Top-down-Ansatz mit schrittweiser Detaillierung ermöglicht dabei eine frühzeitige Erkennung von Fehlern im Designentwurf. Durch die kontinuierliche Verifikationsmöglichkeit und die anschließende automatische Codegenerierung werden dabei typische Fehlerquellen eingedämmt. Die grafische Beschreibung macht außerdem nicht nur eine Darstellung der Algorithmen möglich, sondern visualisiert auch die Systemarchitektur. Durch die Entwicklung des Verifikations- und Code-Generierungspfades in Richtung auf Hardwarebeschreibungssprachen werden auch Problemstellungen adressiert, die erst während des Entwicklungsprozesses in Software bzw. Hardwareimplementierung partizioniert werden. ■

Beitrag als PDF auf [www.duv24.net](http://www.duv24.net)

more @ click DV044253 >

Anzeige