# Echtzeitsimulation von Electronic-System-Level-Modellen

Florian Voit
florian.voit@siemens.com
Siemens AG, Industry Sector, Drive Technologies Division, Motion Control Systems
Frauenauracher Straße 80, 91056 Erlangen, Deutschland

## Kurzfassung

In einem modellbasierten Entwicklungsprozess entstehen funktionale Modelle des zu entwickelnden Produkts, zum Beispiel als ausführbare Spezifikation. Diese Modelle werden frühzeitig und fortlaufend auf Vollständigkeit und Konsistenz geprüft und unterliegen einer Reihe von qualitätssichernden Maßnahmen. Wegen ihrer hohen Qualität sollten diese Modelle möglichst vielseitig eingesetzt werden – auch nach Beendigung des eigentlichen Entwicklungsprojektes.

In [1] wurde die Verwendung von SystemC als Beschreibungssprache für ausführbare Spezifikationen dargestellt. Der vorliegende Beitrag beschreibt, wie diese SystemC-Modelle in Echtzeit simuliert werden – sei es im Rapid Prototyping zur Absicherung der Produktspezifikation oder in Hardware-in-the-Loop-Tests zur Nachbildung einer virtuellen Testumgebung.

Es wird darauf eingegangen, wie die Co-Simulation von SystemC-Modellen und Simulink-Modellen auf einer echtzeitfähigen Hardware-Plattform funktioniert und welche Voraussetzungen dafür zu schaffen sind. So gibt es mehrere Ansätze zur funktionalen Abstraktion und folglich unterschiedliche Mechanismen zur Einbindung von physikalischen I/O-Schnittstellen.

SystemC hat eine große Verbreitung im Hardware-Software-Co-Design digitaler Systeme gefunden. Entsprechend haben sich Standards und Modellierungsmethodiken ausgeprägt, die einen Zielkonflikt mit dem gängigen Vorgehen bei der Echtzeitsimulation analoger Systeme erzeugen. Dies wirkt sich vor allem auf den Rechenzeitbedarf und die erzielbaren Abtastzeiten aus. Am Beispiel eines Frequenzumrichters zum Betrieb von Elektromotoren werden die Erkenntnisse gezeigt und diskutiert.

## 1 Motivation

Zur Verkürzung der Markteinführungszeit neuer Produkte wird ein Entwicklungsprozess angestrebt, bei dem möglichst alle beteiligten technischen Disziplinen parallel arbeiten können. Bild 1 zeigt die parallele Struktur des Entwicklungsprozesses wie in [1] beschrieben. Dieser Ablauf erfordert eine gemeinsame funktionale Spezifikation, die zum Beginn der Designphase qualitätsgesichert vorliegen muss. Sie sollte als sogenannte ausführbare Spezifikation gestaltet, also ein Simulationsmodell auf geeignetem Abstraktionsniveau sein.

Nur dann ist eine tiefgreifende Prüfung auf Vollständigkeit und Konsistenz möglich. Reviews von statischen Spezifikationen auf der Basis von Fließtext, Tabellen, Zeichnungen und Diagrammen dokumentieren die gewünschte Systemdynamik meist nur

unzureichend und verursachen Missverständnisse durch unterschiedliche Interpretationen interdisziplinärer Sachverhalte.

Die ausführbare Spezifikation muss frühzeitig von allen beteiligten Parteien validiert werden. Dies geschieht in den jeweiligen Entwicklungsumgebungen unter geeigneten Rahmenbedingungen. Da die Spezifikationsprüfung auf dem kritischen Pfad eines Entwicklungsprojektes liegt, sollte sie ohne Modelländerung oder gar Neumodellierung in den Simulationswerkzeugen aller technischen Disziplinen sofort durchführbar sein.

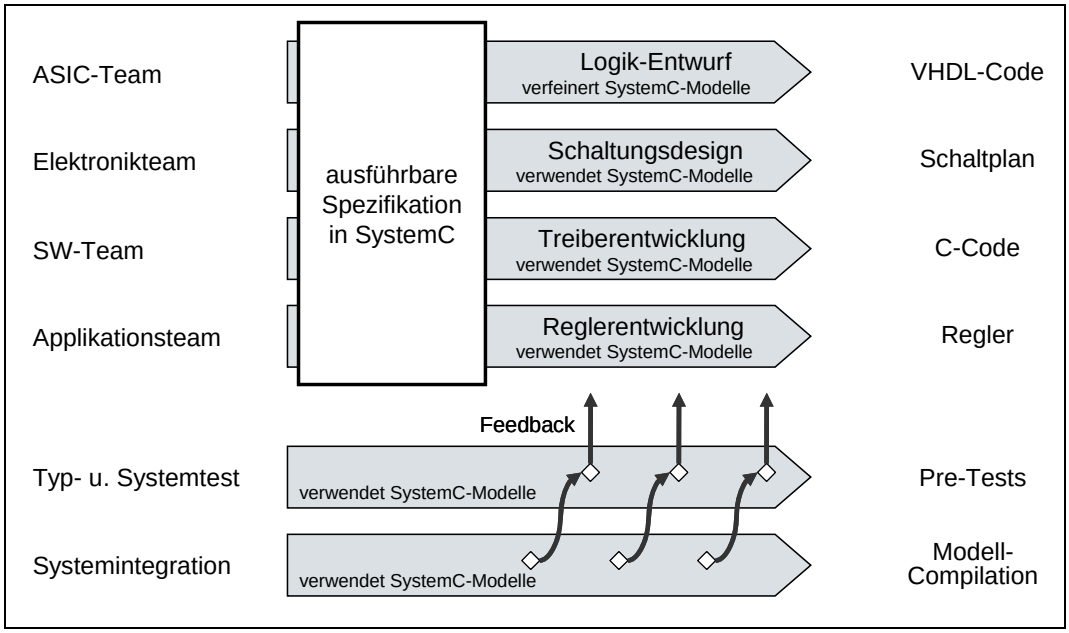


**Bild 1:** Paralleler Entwicklungsprozess

Echtzeitsimulationen werden ausgeführt, wenn die Kopplung eines Simulationsmodells mit realen physikalischen Komponenten notwendig ist. Typische Anwendungsfälle sind das Rapid Prototyping, bei dem ein simuliertes Produkt in realer Umgebung gestestet wird, sowie die Hardware-in-the-loop-Simulation (HIL), die ein reales Produkt in eine simulierte Umgebung bettet. Natürlich sind andere Kombinationen und Konstellationen ebenfalls möglich.

# 2 Modellbeschreibungssprache SystemC

SystemC ist eine C++-Klassenbibliothek und wurde als IEEE 1666 standardisiert. Traditionell wird SystemC für den Systementwurf digitaler Systeme und für Hardware-Software-Codesign eingesetzt. Mittlerweile haben sich ausgefeilte Modellierungsmethoden und -konventionen gebildet, die von freien wie auch von kommerziellen Werkzeugen und Herstellern unterstützt werden. SystemC-Modelle, wie auch der von der Open SystemC Initiative (OSCI) zur Verfügung gestellte Referenz-Simulator, sind in diverse Simulationswerkzeuge mit C/C++-Schnittstelle integrierbar. Auf diesem Weg ist auch

eine Kopplung mit Analogsimulatoren wie Simulink möglich, um Mixed-Signal-Systeme zu modellieren und zu simulieren.

Eine Stärke von SystemC ist die Abdeckung einer großen Bandbreite von Abstraktionsgraden. Sie reicht von zyklenakkurater Darstellung digitaler Signale bis zu zeitlich lose gekoppelten Transaktionskonzepten. Daher ist SystemC eine geeignete Wahl zur Modellierung von ausführbaren Spezifikationen. Das sind abstrakte Modelle der angestrebten Produkt- oder Systemfunktionalität, die schrittweise verfeinert werden, beziehungsweise als goldene Referenz für das detaillierte Design dienen.

# 3 ESL-Modellierungstechniken

Die traditionelle Domäne von SystemC liegt im HW-SW-Codesign, insbesondere bei der Entwicklung von ASICs, Mikroprozessoren und -controllern sowie System-on-Chip (SoC) der unterschiedlichsten Branchen. Dabei wird die HW als virtuelle Plattform modelliert und erlaubt dem SW-Entwickler die gleichen SW-Entwicklungsphasen wie ein FPGA-basierter HW-Prototyp, ohne dass diese HW bereits existiert.

Die HW-inhärente Parallelität von Funktionen und Abläufen, wie sie unter anderem in Form von Zustandsmaschinen und Logikgattern auftritt, wird in SystemC durch parallele Threads abgebildet. Die Threads laufen unter Kontrolle eines im SystemC-Simulatorkern integrieren Schedulers. Dieser Scheduler benötigt dazu den Zugriff auf eine Multi-Threading-Schnittstelle wie POSIX-Threads, Windows-Fibers oder QuickThreads.

Typische SystemC-Modelle erzeugen je nach Modellgröße und -architektur eine beachtliche Anzahl Threads. Deren Verwaltung und vor allem die erzwungenen Threadwechsel kosten Rechenzeit, die man zur Beschleunigung der Simulationsläufe jedoch gerne einsparen möchte.

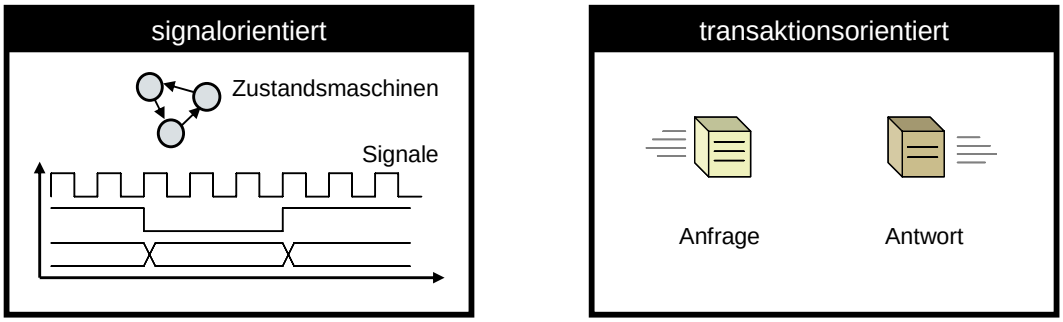


**Bild 2:** Signalorientierte versus transaktionsorientierte Kommunikation

Das erreicht man durch die Vermeidung von Signalflanken und Ereignissen. So wird die Kommunikation zwischen Modellkomponenten möglichst nicht signalorientiert, sondern transaktionsorientiert modelliert. SystemC unterstützt dieses Vorgehen durch das Konzept des Transaction Level Modelling (TLM) und die standardisierte TLM-Bibliothek.

Wie in Bild 2 dargestellt, wird bei der transaktionsorientierten Kommunikation nur ein Datenpaket in jede Kommunikationsrichtung ausgetauscht. Die SW-Implementierung dieser Kommunikationsart besteht aus wenigen Funktionsaufrufen innerhalb eines

Threads. Das ist erheblich aufwandsärmer als die signalorientierte Kommunikation, bei der diverse Signalfolgen von Zustandsmaschinen im interaktiven Ablauf von verschiedenen Threads erzeugt werden müssen.

Ein weiteres Konzept zur Rechenzeitoptimierung ist die in Bild 3 gezeigte vorübergehende zeitliche Entkopplung (Temporal Time Decoupling) von Modellkomponenten, beziehungsweise ihren Threads. Bei diesem Ansatz werden Threads in ihrer Reihenfolge neu sortiert, so dass unter Wahrung der Modellkausalitäten einzelne Threads möglichst lange und ohne Unterbrechung durch andere Threads ablaufen.

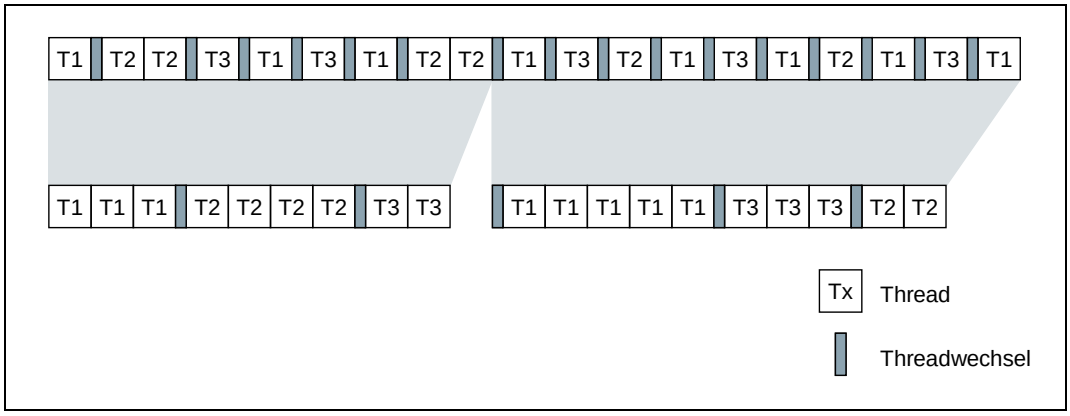


**Bild 3:** Vorübergehende zeitliche Entkopplung von Threads

Dabei werden bei Bedarf die entstehenden Daten, Zustände und Transaktionen mit Zeitstempeln versehen, die den Beginn ihrer Gültigkeit festlegen. Es wird also zu möglichst wenigen Zeitpunkten in konzentrierter Form alles berechnet, was „in der Nähe dieses Zeitpunktes“ passiert.

# 4 SystemC-Portierung auf eine Echtzeitplattform

Für die meisten Echtzeitanwendungen ist es essentiell, über I/O-Schnittstellen mit der physikalisch realen Umwelt zu interagieren. Ferner kann es notwendig sein, das digitale ESL-Modell zusätzlich mit analogen Modellkomponenten zu koppeln und gemeinsam zu simulieren. Eine weit verbreitete Echtzeitplattform ist das dSPACE-System, das Simulink-basierte Modelle und diverse dSPACE-spezifische I/O-Karten unterstützt. Aufgabe war nun, den SystemC-Referenzsimulator der OSCI auf die dSPACE-Plattform zu portieren und einen Datenaustausch zu Simulink-Modellen zu ermöglichen.

Leider ist der OSCI-Simulator V2.2.0 im Originalzustand nicht in Simulink einbettungsfähig, da er stets als eigener Prozess läuft, innerhalb eines Prozesskontextes nicht multi-Instanz-fähig ist und auch nicht von außerhalb des SystemC-Modells gesteuert werden kann. Diese Schwierigkeiten lassen sich jedoch durch Eingriffe in den Quelltext des Simulatorkerns überwinden. Zusätzlich waren umfangreiche SW-Arbeiten notwendig, um den C++-Code des SystemC-Simulators für die Echtzeitplattform zu kompilieren, eine SystemC-verständliche Multi-Threading-Schnittstelle auf der Echtzeitplattform zu

implementieren und einen Adapter für den Datenaustausch und für die Synchronisation zwischen SystemC-Modell und Simulink-Modell zu entwickeln.

# 5 Beispielanwendung Frequenzumrichter

Frequenzumrichter der Produktfamilie Sinamics G120 bestehen im Wesentlichen aus einer Control Unit (CU) und einem Power Module (PM), die über eine serielle Schnittstelle miteinander kommunizieren. In typischen Applikationen wie die in Bild 4 wird das PM mit dem Stromnetz und einem Motor verbunden, wobei letzterer das Drehmoment für die eigentliche Applikation erzeugt. Im Inneren des PM befinden sich neben Gleichrichter, Zwischenkreis und Wechselrichter auch Komponenten zur analogen und digitalen Signalverarbeitung sowie die Einrichtungen zur Kommunikation mit der CU.

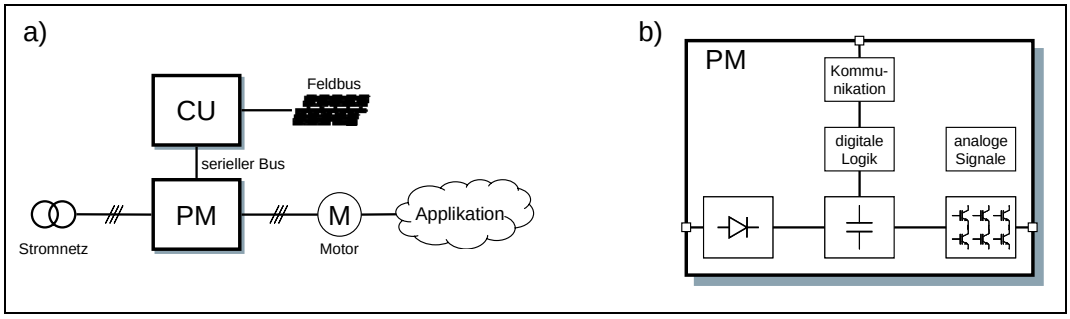


**Bild 4:** Aufbau eines Antriebs (a) und eines Power Module (b)

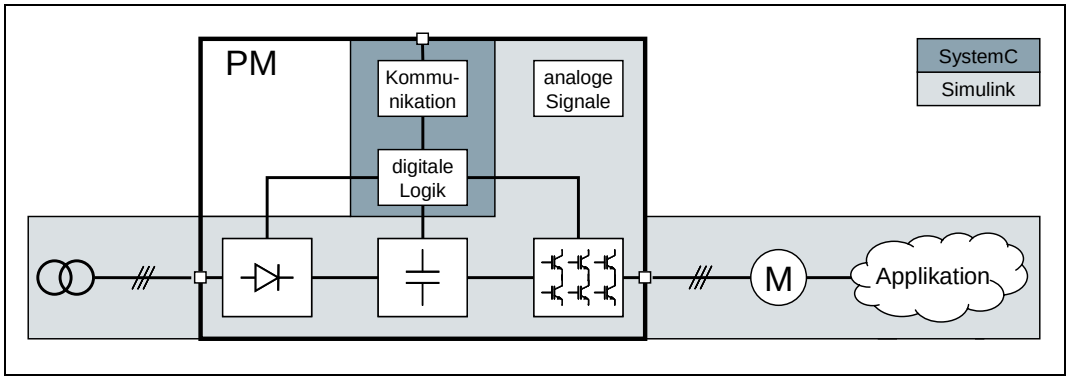


**Bild 5:** Modellierung eines Frequenzumrichters mit Beschaltung

Bild 5 veranschaulicht die Aufteilung des Modells in einen SystemC-Anteil und einen Simulink-Anteil. Die Kommunikationsschnittstelle und die digitalen Logikelemente werden als ESL-Modell in SystemC beschrieben. Das beinhaltet die Steuerung der internen Zustände, die diskreten Algorithmen zur Pulsmustergenerierung für den Wechselrichter, aber auch Überwachungsfunktionen zum Schutz der Hardware. Die analoge Signalverarbeitung und die leistungselektronischen Anteile werden als Simulink-Modell nachge-

bildet. Schließlich gibt es noch Funktionen wie die Analog-Digital-Wandler, die sich auf beide Modellarten aufteilen.

Das Gesamtmodell, bestehend aus Simulink- und SystemC-Anteilen ist in der Simulink-Workbench offline simulierbar. Per Knopfdruck lässt es sich für die Echtzeitplattform übersetzen und anschließend dort ausführen.

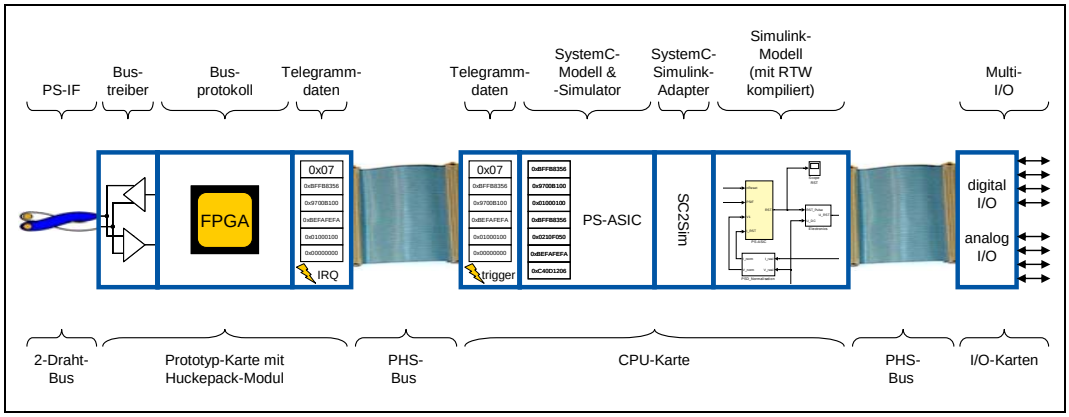


**Bild 6:** Modell-Allokation auf der Echtzeitplattform

Die Allokation der Modellteile auf der Echtzeitplattform erfolgt wie in Bild 6 abgebildet: Das SystemC-Modell, der SystemC-Simulatorkern, das Simulink-Modell sowie die Kopplung beider Modelle laufen auf der CPU-Karte. Das Protokoll der Kommunikation mit der CU wird von einer proprietären Spezialkarte abgewickelt, die gegenüber dem Simulationsmodell als abstrakte Telegrammschnittstelle in Erscheinung tritt. Schließlich werden mit Hilfe kommerzieller I/O-Karten die Analog- und Digitalsignale erzeugt, die üblicherweise bei einem Antrieb auf dem Testplatz erwartet und gemessen werden. Dazu gehören vor allem Tachosignale und Bremssteuersignale, aber auch Analogwerte der Zwischenkreisspannung und Motorströme zur Auswertung in einem Testsystem oder Oszilloskop. Die I/O-Karten werden vom Simulink-Modell angesteuert unter Zuhilfenahme der Bibliotheksblöcke, wie sie der Hersteller der Echtzeitplattform für Simulink zur Verfügung stellt.

In dieser Beispielanwendung liegt die Abtastzeit bei $T_s = 50$ µs. Zeichnet man wie in Bild 7 den Rechenzeitbedarf der Simulation über die Zeit auf, erkennt man eine Grundlast von etwa 30 µs bis 35 µs und Rechenzeitspitzen von etwa 45 µs. Die Spitzen tauchen periodisch alle 250 µs auf. Sie werden von dem SystemC-Modell verursacht, das seine Berechnungen wegen oben genannter Überlegungen auf den Beginn einer PWM-Periode konzentriert. Die PWM-Frequenz ist auf 4 kHz eingestellt, was einer Periodendauer von 250 µs entspricht. Die Rechengrundlast entsteht durch das Simulink-Modell, das gleichförmig bei jedem Abtastschritt denselben Algorithmus abarbeitet.

Diese ungleiche Verteilung der Rechenlast ist für Echtzeitanwendungen ungünstig, weil die Abtastzeit des gesamten Simulationssystems nicht kürzer sein darf, als die periodischen Rechenzeitspitzen es zulassen. Der Modellierer steht in einem Zielkonflikt: Einerseits sollte für die Echtzeitsimulation die Rechenzeitverteilung möglichst gleichför-

mig sein. Das SystemC-Modell sollte also die internen Berechnungen möglichst kleinteilig und zeitnah durchführen. Möchte man andererseits dasselbe SystemC-Modell auch für weitere Aufgaben wie HW-SW-Codesign einsetzen, wird die ungleichförmige, konzentrierte Berechnung angestrebt, um Threadwechsel einzusparen und die Gesamtsimulationszeit zu minimieren.

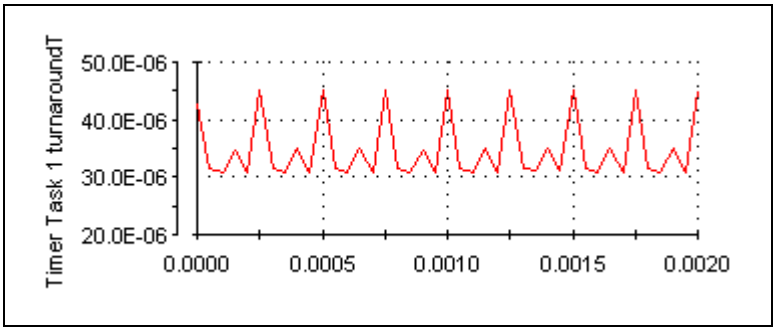


**Bild 7:** Rechenzeitbedarf

Es sind verschiedene Lösungsansätze für dieses Problem denkbar. Erstens könnte man eine Priorität entweder für die Echtzeitsimulation oder für das HW-SW-Codesign festlegen und das SystemC-Modell mit entsprechender Granularität erstellen. Das bedeutet, dass nur ein Einsatzzweck optimal befriedigt würde. Zweitens wäre es denkbar, die Granularität des Modells konfigurierbar zu gestalten. Jedoch resultiert daraus ein erheblich größerer Modellierungsaufwand, der außerdem die Gefahr von Inkonsistenzen des Modellverhaltens in sich birgt.

Drittens ließe sich das SystemC-Modell soweit abstrahieren, dass insgesamt weniger berechnet werden muss und die Rechenzeitspitzen somit kleiner werden. In diesem Fall ist aber trotzdem sicherzustellen, dass die Modellgenauigkeit adäquat zum gewünschten Anwendungsfall ist. Als vierten Weg könnte das SystemC-Modell in mehrere kleinere Modelle aufgeteilt und unabhängig voneinander ko-simuliert werden. Durch die stochastische Verteilung der Rechenzeitbedarfe stellt sich hoffentlich ein gleichmäßigerer Verlauf ein. Jedoch können sich die Rechenzeitbedarfe in ungünstigen Situationen auch überlagern und zwar seltene, aber dennoch höhere Spitzen verursachen.

Die Auswahl eines geeigneten Lösungsansatzes hängt stark von der Applikation und den angestrebten Einsatzzwecken ab. Eine Kombination mehrerer Ansätze ist möglich.

# 6 Zusammenfassung

Ein digitales ESL-Modell wurde zusammen mit analogen Komponenten als Mixed-Signal-System auf einer Echtzeitplattform implementiert. Dabei wurden sowohl signal- als auch transaktionsorientierte Modellierungstechniken eingesetzt. Die Interaktion mit der Außenwelt wurde über I/O-Karten ermöglicht. Während der Simulationsdurchführung war eine ausgeprägte Ungleichverteilung der Rechenzeitbedarfe pro Abtastschritt zu beobachten, was die erzielbaren Abtastraten des Gesamtsystems beeinflusste.

Es zeigt sich, dass das als ausführbare Spezifikation erzeugte ESL-Modell aus der Produktentwicklung für die Echtzeitsimulation verwendet werden kann. Damit steht der

Weg offen für die Wiederverwendung solcher Modelle zu Zwecken des Rapid-Prototyping wie auch zur Gestaltung von physikalisch realen Testständen.

# 7 Literatur


[1] *Voit, F.*: Mehrfachverwendung von Simulationsmodellen im multi-disziplinären Entwicklungsprozess. ASIM 2009, B. Luther (Hrsg.). Aachen: Shaker Verlag, 2009

[2] *Open SystemC Initiative*: http://www.systemc.org