# Perfekt synchronisiert

## Das Echtzeit-Verhaltens von vernetzten Steuergeräten beherrschen

**Ein Steuergerät für ein stark vernetztes System soll alle Funktionen erfüllen sowie rechtzeitig fertig gestellt und kostengünstig produziert werden können. Aus Sicht der Entwickler und Kaufleute, Zulieferer und OEMs wird die Qualität des zu entwickelnden Steuergeräts jedoch durch eine Vielzahl von Faktoren mit sehr unterschiedlichem Einfluss definiert.**

Von Tapio Kramer

Immer bedeutsamer werden diejenigen Faktoren, die vom korrekten Echtzeit-Verhalten abhängen. Der geforderte Funktionsumfang ist nicht allein erfüllt, wenn das Steuergerät die richtige Reaktion zeigt. Sie muss auch zum richtigen Zeitpunkt, also rechtzeitig erfolgen. Je mehr Funktionen, Systeme und Kommunikationsstrecken an dieser Reaktion beteiligt sind, desto schwieriger ist es, diese Rechtzeitigkeit sicherzustellen. Aus der Komplexität eines verteilten Systems resultiert meist ein vielschichtiges Zeitverhalten. Dies verlängert meistens die Entwicklungszeit und auch die Produktkosten, da zur Absicherung mehr Rechenleistung eingesetzt wird als tatsächlich nötig.

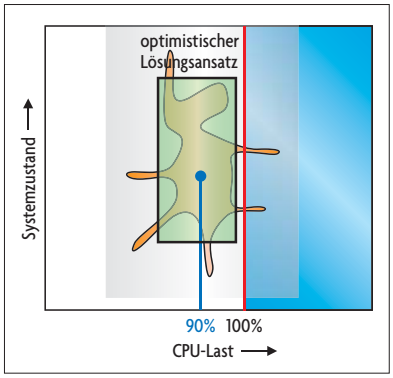


**Bild 1. Stochastische Ansätze decken zwar eventuell nicht alle Systemzustände ab, erlauben aber ein wesentlich effizienter ausgelastetes System.** (Quelle: Continental AG)

Die Produktkosten sinken, wenn alle Komponenten optimal, d.h. gleichmäßig hoch, ausgelastet und Lastspitzen vermieden werden. Eine gleichmäßige Verteilung von periodisch auftretenden Aufgaben mittels Phasenverschiebung, häufig durch Abstände der Aktivierungszeitpunkte gesteuert, bietet daher ein hohes Optimierungspotential. Die Vorhersage und exakte Planung der Ressourcen-Nutzung bekommt also eine bedeutende Rolle.

### Modellierung erlaubt bessere Vorhersagen als Prototyping

Um das Echtzeit-Verhalten von eingebetteten Systemen zu bestimmen, können verschiedene Ansätze gewählt werden. So können zunächst Prototypen entwickelt werden, die dann integriert und getestet werden. Dabei ist das Echtzeit-Verhalten jedoch erst überprüfbar, wenn die Software auf der Ziel-Hardware läuft und von der realen Umgebung stimuliert wird. Besser ist der Ansatz, das System zu modellieren, um es bereits in der frühen Architekturphase grob und im Verlauf der Entwicklung immer detaillierter auf sein Echtzeit-Verhalten hin analysieren zu können.

Zur Erstellung eines Echtzeit-Modells des Systems ermittelt man das Zeitverhalten der einzelnen Komponenten und verwendet anschließend Simulation und/oder Validierung, um das Echtzeit-Verhalten vorherzusagen (**Bild 1**). Beide Verfahren haben ihre Vor- und Nachteile, auf die im Weiteren eingegangen wird.

Die Simulation ist, vergleichbar mit Systemen im Hardware-in-the-Loop-Test, in der Lage, reale Situationen nachzustellen. Je besser das System modelliert und mit realen Stimuli angeregt wird, desto besser wird auch die Testabdeckung aller möglichen Betriebssituationen sein. Die mathematische Analyse mittels Validierung ermittelt die Best-Case- und Worst-Case-Fälle des Echtzeit-Verhaltens des Systems. Extremsituationen werden sicher erfasst und können mit den Echtzeit-Anforderungen und den Vorgaben hinsichtlich Leistungsfähigkeit verglichen werden.

### Nicht zeitrelevante Faktoren werden gezielt abstrahiert

Bei allen Modellen von Systemen fokussiert man die Eigenschaften, die für das gewählte Analyseverfahren relevant sind. Das Echtzeit-Verhalten der Systemkomponenten wird im Modell abgebildet; komplexe, nicht timing-re-

levante Faktoren werden gezielt abstrahiert. Ein Echtzeit-Modell eines funktionsreichen Steuergeräts kann dennoch einfach und sehr präzise sein (**Bild 2**).

Ausgangsbasis sind die Hardware-Elemente des eingebetteten Systems (Prozessoren und Bussysteme) mit ihren Scheduling-Verfahren, die definieren, wie die Leistungsanforderungen der Funktionen auf diesen abgearbeitet werden. Der Bedarf an Rechenzeit oder Datenübertragungsrate einer Funktion wird als Ausführungszeit und Anforderungsschema — beispielsweise zyklische oder ereignisgesteuerte Ausführung — beschrieben. Im Fall eines OSEK-Steuergeräts werden die Tasks und Interrupts mit ihren Eigenschaften und Abhängigkeiten modelliert. Die Anregungen der Software über Interrupts von außen werden als Stimuli modelliert. Einige dieser Parameter können aus einem OIL-File (OSEK Implementation Language) eingelesen werden. Andere, wie beispielsweise die Ausführungszeiten, sind in der Oberfläche des Modellierungswerkzeugs leicht zu ergänzen.

Dass die Software-Routinen einer Task nicht nur konstante Ausführungszeiten haben, ist häufig vernachlässigbar. Denn die für die Echtzeit-Fähigkeit entscheidende Antwortzeit einer Software wird wesentlich vom Scheduling beeinflusst, das bei der Timing-Analyse korrekt berücksichtigt wird.

## Die Simulation des Echtzeit-Verhaltens

Zur Simulation wird das Timing-Modell mit Hilfe von Interrupts von Zeitgebern, externen Eingängen oder Bus-Nachrichten stimuliert und sein Echtzeit-Verhalten aufgezeichnet. Der Simulator chronSIM von Inchron steuert die Zuteilung der Zeitbedarfe der Software auf die modellierten Prozessoren

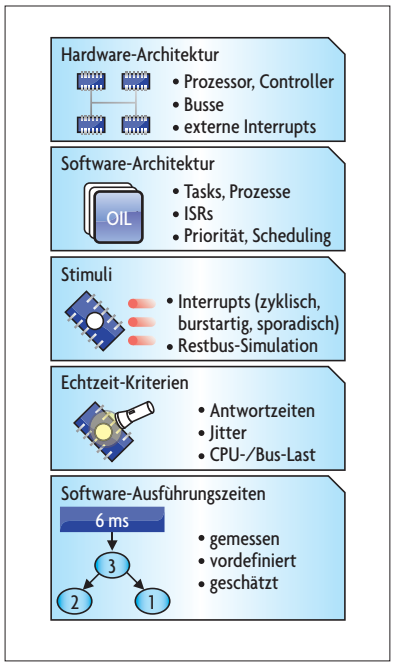


**Bild 2. Systeminformationen, die das Echtzeit-Verhalten eines Steuergeräts beeinflussen und im Timing-Modell abgebildet werden.**

und Busse entsprechend der gewählten Scheduling-Schemata. Jede Aktivierung, Verdrängung und Terminierung eines Prozesses (Task, ISR oder Bus-Nachricht) wird als Ereignis in einer Messspur aufgezeichnet. Die Prozessoren und Bussysteme haben häufig individuelle, asynchrone Taktgeber und externe Anregungen wie Kurbelwellen- oder Radsensoren. Sie werden mittels patentierter Methoden als driftende Uhren und zyklische und driftende Stimuli in der Simulation berücksichtigt, da sie wesentlichen Einfluss auf das Echtzeit-Verhalten haben (**Bild 3**).

Die Visualisierung stellt die Aktivierung, Blockierung und Verdrängung der Prozesse gemäß dem Scheduling dar. Damit kann leicht nachvollzogen werden, wie Prozesse niedriger Priorität von Prozessen hoher Priorität unterbrochen und gegebenenfalls gar nicht rechtzeitig abgeschlossen werden. In der Simulation können auf einfache Weise verschiedene Anregungsszenarien generiert werden, die extreme Auslastungen des Systems bewirken.

Messungen von Zeitabständen, z.B. Task-zu-Task-Aktivierungen oder Anfang-zu-Ende-Antwortzeiten von Wirkketten und statistische Analysen geben ein gutes Bild von der zu erwar-

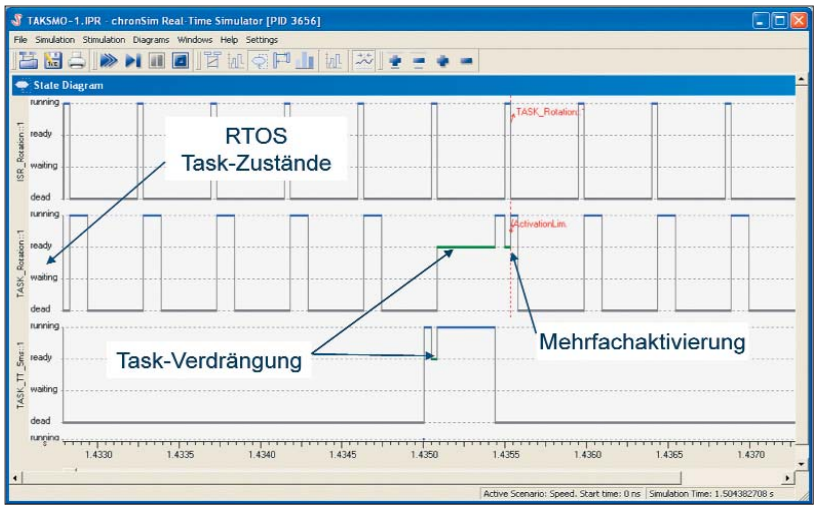


**Bild 3. Zyklische Tasks verschiedener Perioden können Mehrfachaktivierungen auslösen und zu Timing-Problemen führen.**

x= 0
Y= 239.5

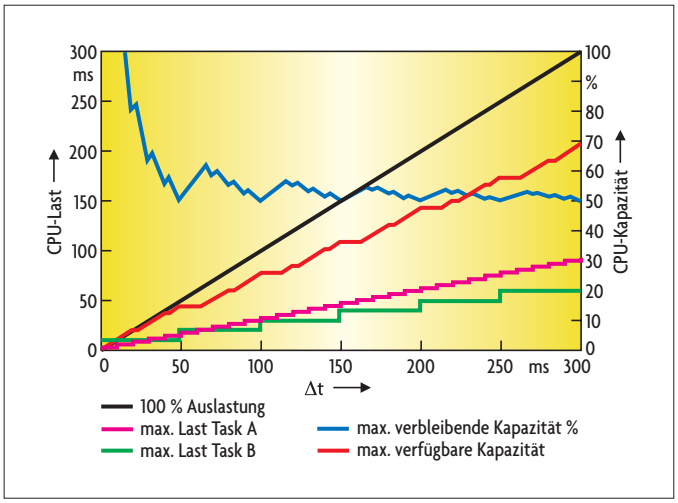


**Bild 4. Im Ereignisspektrum wird die CPU-Last der Prozesse über die Zeitintervalle Δt dargestellt; dies erlaubt detaillierte Analysen der Worst-Case-Fälle.**

tenden Echtzeit-Fähigkeit des modellierten Systems. Die Prozessor- und Busauslastung kann über längere Zeiträume beobachtet sowie unerwünschte Lastspitzen genauer analysiert und in ihrer Ursache verstanden werden.

## Validierung des Systems mittels Ereignisspektralanalyse

Für die Validierung des Echtzeit-Verhaltens wird das Timing-Modell durch chronVAL mit Hilfe mathematischer Algorithmen untersucht. Aus allen möglichen Betriebssituationen werden die Fälle ermittelt, in denen die Ausführungszeiten und Anregungen des Systems zu best- und schlechtestmöglichen Antwortzeiten führen. Mathematisch gesehen ist diese Analyse eine Faltung, ähnlich der Fourier-Analyse. Die im Zeitablauf beschriebenen Stimuli des Systems werden zu Ereignisspektren umgerechnet und die Antwortzeiten der Software-Prozesse und Wirkketten daraus ermittelt (**Bild 4**).

Die Ereignisspektralanalyse stellt den Ressourcen-Bedarf der Prozesse über Zeitintervalle dar. Aus den Diagrammen kann abgelesen werden, in welchen Zeitintervallen beispielsweise der Prozessor oder der Bus von einem Prozess angefordert wird. Durch Überlagerung der Prozesse wird deutlich, ob in einem Intervall noch Ressourcen frei bleiben oder ob die Prozesse das System vollständig auslasten. Nur den Worst- und Best-Case aufzulisten, ohne die Aussage über die Zeitintervalle darzustellen, gäbe dem Entwickler wenig Möglichkeiten zu verstehen, wie diese Fälle zu vermeiden wären.

Mit einfachen Eingaben in der Benutzeroberfläche kann eine Sensitivitätsanalyse zeigen, wie sich eine geänderte Anregung, beispielsweise eine Absenkung der Task-Ausführungshäufigkeit von alle 5 ms auf alle 10 ms, auf die Gesamtleistung und einzelne Antwortzeiten auswirkt. Für den Systemarchitekten bedeutet dies, er kann das Abbilden der Funktionen auf Ressourcen und der Runnables auf Tasks hinsichtlich ihrer Auswirkung auf die Systemleistung schnell untersuchen und optimieren.

## Simulation und Validierung können sich ergänzen

Die Simulation ist prinzipiell, genauso wie Tests, optimistisch, da nur die Fälle untersucht werden, die in der Simulation auch angeregt wurden. Eine Validierung hingegen ermittelt die Extremfälle des Systemverhaltens, wird aber aus den Annahmen im Modell das System pessimistisch überschätzen und Fälle aufzeigen, die in der Realität nicht vorkommen.

Daher ist es vorteilhaft, ein System mit beiden Mitteln zu untersuchen. Mit Hilfe der Validierung werden alle Extremsituationen berücksichtigt. Diese pessimistische Analyse kann garantieren, dass das System geforderte Antwortzeiten einhalten wird. Werkzeuge wie chronVAL können helfen, das System auszulegen und später abzusichern. In der Simulation mit chronSIM werden die Abläufe im System nachvollziehbar. Es kann genau beobachtet werden, welche Situationen das System zu einem bestimmten Verhalten gebracht haben. Über Monte-Carlo-Simulationsläufe bekommt der Entwickler auch die Information, mit welcher Wahrscheinlichkeit bestimmte Situationen eintreten werden.

Beispielsweise führen bei vielen Echtzeit-Systemen Mehrfachaktivierungen von Tasks — häufig sind das Schwebungseffekte zyklischer Prozesse — zu Problemen. Die Validierung zeigt zuverlässig auf, ob diese vorkommen können. Mit Hilfe der Simulation sind die Ursachen für die Mehrfachaktivierung leicht nachvollziehbar. In der Kombination beider Verfahren können daher Extremsituationen sicher erfasst und auf ihre Relevanz für das System untersucht werden.

Betrachtet man die aktuellen Trends der Elektrik/Elektronik der Automobilindustrie, wird schnell deutlich, dass es immer anspruchsvoller wird, ein Steuergerät zu entwickeln. Durch die Konzentration von immer mehr Funktionen auf wenigen Domänen-Controllern sowie stärkere Vernetzung und Verteilung der Funktionen, sind die Systeme und ihr Echtzeit-Verhalten immer schwieriger zu beherrschen.

Timing-Modelle und deren Analyse mit geeigneten, leistungsfähigen Werkzeugen ermöglichen es, die Echtzeit-Probleme der Systeme frühzeitig im Entwicklungsprozess zu adressieren. Wirft man einen Blick in Richtung AUTOSAR mit seinen wiederverwendbaren Software-Modulen verschiedener Zulieferer, wird es unumgänglich, das Zeitverhalten mittels Timing-Modellen zu spezifizieren. *sj*

**Dipl.-Ing. Tapio Kramer**
studierte Elektrotechnik an der TU Braunschweig. Danach arbeitete er in verschiedenen Positionen bei Nokia, National Instruments, Wind River und jetzt Inchron. Dort ist er für das Marketing und das Produkt-Management zuständig. In der Betreuung von Partner- und Kundenprojekten hat er sich intensiv mit der Echtzeit-Thematik auseinandergesetzt.
kramer@inchron.com