

Absicherung des Echtzeitverhaltens mittels virtueller Integration

Tapio Kramer, Dr. Ralf Münzenberger

Veröffentlicht im Vortragsband zur
4. Tagung - Simulation und Test für die Automobilelektronik, IAV, Mai/Juni 2010, Berlin

Abstract

One third of the effort spent for debugging the integration of components into the electrical/electronic (E/E) systems is caused by timing problems. These time and money consuming real-time issues can significantly be reduced by virtual integration. The early integration of models of uncompleted components into the overall system model is feasible on many levels, within control units, into systems of networked control units, and even into systems spanning multiple domains.

The paper describes a method of modeling the real-time behavior of E/E systems and its application. The models are subsequently simulated and analyzed for real-time requirements.

Kurzfassung

Ein Drittel des Aufwands zur Fehlersuche bei der Integration von E/E-Komponenten wird für Echtzeitprobleme aufgewendet. Diese zeit- und kostenaufwändigen Echtzeitfehler lassen sich wesentlich verringern, indem bereits vor der Fertigstellung der Komponenten eine virtuelle Integration des Gesamtsystems durchgeführt wird. Diese virtuelle Integration ist auf unterschiedlichen Ebenen, vom einzelnen Steuergerät über vernetzte Steuergeräte einer Domäne bis hin zu domänenübergreifend vernetzten Steuergeräten effizient einsetzbar.

Im vorliegenden Bericht wird eine Methode für die Modellierung des Echtzeitverhaltens von E/E-Systemen vorgestellt und exemplarisch angewendet. In der anschließenden Simulation wird das Echtzeitverhalten des modellierten Systems aufgezeigt und Echtzeitanforderungen überprüft.

1. Die Integrationsaufgabe

Das E/E-System eines Fahrzeuges erfüllt seine Anforderungen als ein Gesamtsystem, das aus vielen interagierenden Teilkomponenten integriert wird. Erst das korrekte Zusammenspiel verschiedener Einzelfunktionen, die auf den Steuergeräten verteilt ablaufen, ermöglicht die Erfüllung komplexer Aufgaben, wie z.B. die sichere Realisierung einer aktiven Hinterachslenkung über Drehmomentunterschiede.

Die Integration und das Testen der Teilkomponenten finden jedoch in einer späten Phase des Entwicklungsprozesses statt, wenn sowohl die Hardware verfügbar als auch die Implementierung der Software weitgehend abgeschlossen sind. Zudem ist die Integration der Komponenten in ein Fahrzeug ein mehrstufiger Prozess, bei dem die Verantwortung für die einzelnen Steuergeräte beim Zulieferer (Tier-1) liegt. Für die Vernetzung und Integration in das Fahrzeug ist hingegen der OEM zuständig.

Häufig muss der OEM bei der Erstellung der Systemarchitektur, genauso wie die Tier-1 bei der Erstellung der Steuergerätearchitekturen, mit unbekannten Anforderungen der Komponenten umgehen. Oftmals sind die Komponenten teils noch nicht genau spezifiziert und entwickelt, teils bleibt aus Gründen des IP-Schutzes das Verhalten dem Architekten unbekannt. Unter diesen Voraussetzungen sind Entscheidungen, die zu einer robusten, echtzeitfähigen Architektur führen sollen, nur bedingt möglich. Es gilt daher, möglichst früh die Anforderungen detailliert zu kennen, um sie in der Architektur und Implementierung zu berücksichtigen.

1.1 Echtzeitprobleme

Mit der zunehmenden Komplexität der einzelnen Steuergeräte und der gleichzeitig steigenden Vernetzung nimmt der Aufwand zur Suche und Behebung von Echtzeitproblemen sowohl bei Tier-1 als auch bei OEMs stark zu. Nach Aussagen von OEMs verursachen Echtzeitprobleme 20-40% des Aufwandes bei der Integration. Selbst wenn jedes Steuergerät für sich keine Echtzeitprobleme aufweist, können beim Zusammenspiel mehrerer Steuergeräte erhebliche Echtzeitfehler auftreten. Beispielsweise können Daten zu spät gesendet bzw. empfangen oder auch überschrieben werden.

Steuergeräte in modernen Fahrzeugen vereinen mehrere Funktionen in einem Gerät und arbeiten stark vernetzt. Mehr und mehr Funktionalität wird von verteilten Teilkomponenten realisiert. Hierbei wird eine verteilte Funktion mit anderen Funktionen um die Systemressourcen konkurrieren. Dazu gehören auch die Buskommunikation und die Ausführung auf anderen Prozessoren, deren Ressourcen jeweils nach unterschiedlichen Scheduling-Strategien zugeteilt werden. Das Echtzeitverhalten der gesamten, verteilten Funktion ist daher von einer Vielzahl von Zuständen, Ereignissen, Ausführungszeiten und Phasenlagen periodischer Vorgänge abhängig.

Viele Funktionen können mittels Algorithmen und Simulation (beispielsweise Simulink) ohne Berücksichtigung der Hardware spezifiziert, implementiert und getestet werden. Die Frage des ‚Was‘ und ‚Wie‘ einer Teilfunktion kann recht gut frühzeitig beantwortet werden. Das ‚Wann‘ ist bei Embedded Systemen mit herkömmlicher Entwicklungsmethodik aber erst erlebbar, nachdem die Software implementiert und auf der Ziel-Hardware integriert und ausgeführt wurde. Wann eine Reaktion des Systems auf eine Anregung erfolgt, hängt stark davon ab, wie schnell einzelne Teilfunktionen nacheinander ausgeführt werden können und wie viele andere Funktionen die knappen Systemressourcen zur gleichen Zeit ebenfalls beanspruchen. Die Definition des ‚Wann‘ in Form von Echtzeitanforderungen erfolgt im Entwicklungsprozess folglich häufig zu spät und oftmals mit mangelnder Präzision.

1.2 Herausforderung AUTOSAR

Um das enorme Einsparpotential der Wiederverwendung zu nutzen, hat sich die Automobilindustrie zum AUTOSAR (AUTomotive Open System ARchitecture) Konsortium zusammengefunden. Die Software wird nicht mehr individuell für ein Steuergerät eines Fahrzeugmodells von einem Zulieferer erstellt. Vielmehr werden plattformunabhängige Software-Komponenten (SWC) definiert, die, von verschiedenen Zulieferern stammend, von Integratoren in das Gesamtsystem integriert werden. Für die Entwicklung der Steuergeräte ergibt sich daraus ein sehr anspruchsvolles Entwicklungsszenario.

Der notwendige Prozess, der die Abhängigkeiten minimieren und die Module entkoppeln soll, sieht vor, dass mit einer Gesamtsystemkonfiguration begonnen wird. Aus dieser leiten sich dann die Konfiguration der Kommunikation und Steuergeräte ab. Letztere werden dann gemäß den darauf verteilten Funktionen konfiguriert.

Aktuell wird der Prozess so noch nicht bei allen Beteiligten gelebt. Die Tier-1 sind noch sehr auf das einzelne Steuergerät, das von ihnen entworfen und größtenteils implementiert wird, konzentriert. Und bei den OEMs ist das Know-How und die Informationsbasis für eine detaillierte Architekturentscheidung noch nicht aufgebaut. Zudem ist häufig die Kommunikationsmatrix noch Ausgangspunkt und nicht Artefakt des Architekturentwurfsprozesses.

Folglich werden heute Komponenten im Gesamtsystem verteilt, deren Architektur schon vor der Verteilung festgelegt wurde. SWC, die für eine Architektur entworfen wurden, werden in ein Steuergerät mit anderer Architektur integriert. In der Natur der Sache liegt dabei die Problematik, dass hierbei viele Informationen von den verschiedenen SWC und Architekturen unbekannt, undefiniert und unvollständig sind – beispielsweise aus IP-Schutz-Gründen und der ‚Henne-Ei-Problematik‘. Das ergibt technisch und auch organisatorisch eine enorme Herausforderung. Denn um die geforderte hohe Qualität in immer kürzerer Zeit zu erreichen, sind lange, teure Iterationszyklen zur Architekturoptimierung über alle Entwicklungsphasen hinweg nicht mehr möglich.

1.3 Entwicklungsrisiko minimieren

Aus dem Blickwinkel der Projektleitung sind Echtzeitanforderungen nicht nur Mittel zum Zweck, um das richtige Produkt zu entwickeln. Sie ergeben auch ein wichtiges Indiz für das Projektrisiko.

Echtzeitfehler sind schwer zu finden und oftmals sehr aufwändig zu beheben. Viele dieser Fehler treten erst in der Integration auf und konnten bis dato auch erst dort gefunden werden. Je robuster eine Systemarchitektur auf dynamisches Zeitverhalten reagiert, desto sicherer werden potentielle Echtzeitfehler nicht auftreten oder zumindest keine fatalen Folgen haben. Können also die Echtzeitanforderungen an die Architektur früh definiert und überprüft werden, lässt sich die Wahrscheinlichkeit von späteren Integrationsproblemen wesentlich reduzieren. Frühe, detaillierte Echtzeitanforderungen und –analysen senken also das Entwicklungsrisiko und eignen sich gut zur Projektsteuerung. Dies ermöglicht die virtuelle Integration.

2. Virtuelle Integration

Die virtuelle Integration der Komponenten in das Gesamtsystem erfolgt modellbasiert in mehreren Phasen. Ausgangsbasis ist ein grobes Modell des Gesamtsystems aus einer frühen Architekturphase und dessen Echtzeitanforderungen. Die Modelle der Komponenten aus der Spezifikation oder aus bereits vorliegenden Fragmenten der Implementierung werden in dieses Systemmodell integriert und die Echtzeitanforderungen werden gemeinsam analysiert. Iterativ wird die Systemarchitektur analysiert und verfeinert, um die Anforderungen zu erfüllen.

Aus den Echtzeitanforderungen an das Gesamtsystem ergeben sich implizit Anforderungen an die einzelnen Komponenten. Und umgekehrt haben Echtzeitanforderungen einzelner Komponenten eine Einschränkung der Gesamtarchitektur zur Folge.

Die Echtzeitanforderungen sind jedoch oft nur global und unscharf definiert. Viele Details der Echtzeitanforderungen ergeben sich aus der Implementierung oder erst bei Integrationstests. Ziel der virtuellen Integration ist es daher, die spärlichen, global definierten Echtzeitanforderungen mit der Architektur früh zu kombinieren und zu verfeinern. Der Integrator kann hier bereits Risikobetrachtungen durchführen und Aussagen treffen, ob spezifizierte Komponenten in das Gesamtsystem integriert werden können oder eventuell angepasst werden müssen. In [1] werden die Vorteile der virtuellen Integration anhand einer Anwendungsstudie im Bereich Body-Controller beschrieben.

2.1 Echtzeit-Systemmodell

Zur virtuellen Integration der Teilkomponenten in das Gesamtsystem wird ein Systemmodell benötigt, das das Echtzeitverhalten abbildet. Mit den Task-Models bietet die INCHRON Tool-Suite die Möglichkeit, das Echtzeitverhalten eines Gesamtsystems mit verteilten Ressourcen zu modellieren, zu simulieren und zu validieren. Hierfür wird das Zeitverhalten der Steuergeräte und Bussysteme in der Architekturphase zunächst mit einer Granularität von Tasks und Busnachrichten modelliert. Annahmen über das Zeitverhalten der Tasks werden als Zeitbudgets zusammen mit den Scheduling-Informationen in die Task-Models eingetragen (siehe Bild 1 links). In das Gesamtsystemmodell werden die Teilmodelle der Steuergeräte und Bussysteme integriert. Die Modellierung kann alternativ in UML (Unified Modeling Language) erfolgen [2].

<pre>ISR(ISR_Rotation) { DELAY(30,unit_us); ActivateTask(Task_Rot); }</pre>	<pre>TASK(TASK_TT_5ms) { DELAY(300, unit_us); ① Schedule(); ② while(!finished()) { ③ DELAY(100, unit_us);} exectime = 10*data_size; ④ DELAY(exectime,unit_us); DELAY(gaussian(500, 10),unit_us); ⑤ TerminateTask(); }</pre>
<pre>TASK(Task_Rot) { DELAY(150,unit_us); TerminateTask(); }</pre>	<p>Beispiel eines komplexeren Task-Models</p> <ul style="list-style-type: none"> ① : konstante Ausführungszeit ② : Aufruf des Schedulers ③ : ablaufabhängige Ausführungszeit ④ : datenabhängige Ausführungszeit ⑤ : wahrscheinlichkeitsverteilte Ausführungszeit

Beispiel des einfachsten Task-Models für OSEK.
 ISR_Rotation wird in 30 µs und Task_Rot in 150 µs ausgeführt.

Bild 1: Einfaches und komplexeres Task-Model

Im Verlauf der Entwicklung werden die Annahmen aus der Architekturphase mit Eigenschaften aus der Implementierungsphase ersetzt. Die Zeitbudgets können feingranular mit detaillierten Verhaltensmodellen (siehe Bild 1 rechts) auf Ebene von Runnables beschrieben werden. Komponenten mit unterschiedlich fortgeschrittener Implementierung werden auf verschiedenen Abstraktionsebenen modelliert und in

das Gesamtsystem integriert. Das Echtzeitmodell des Systems wird so kontinuierlich verfeinert.

2.2 Typische Echtzeitkriterien

Das korrekte Echtzeitverhalten von Systemen kann nach verschiedenen Kriterien beurteilt werden. Abhängig von der gewählten Software-Architektur und –Implementierung sind unterschiedliche Kriterien entscheidend für die korrekte Funktion. Für eine Analyse ist es notwendig, sowohl die Echtzeitkriterien der Teilkomponenten als auch die des Gesamtsystems zu kennen. Typische Echtzeitkriterien sind:

- Antwortzeiten
- Start-zu-Start Jitter von Tasks und Software-Komponenten
- Mehrfachaktivierung von Tasks
- Ausführungsreihenfolge von Software-Komponenten
- Latenzen von Wirkketten (Event Chains)
- Verlust von Daten
- Alter der Daten
- Datenkonsistenz
- Wiederverwendung von Daten (gewollt oder unbeabsichtigt)
- Verlust von Interrupts
- Blockierung von Interrupts

2.3 Virtuelle Integration mit Task-Models

Die beteiligten Parteien (der OEM und die Zulieferer) können mit Hilfe von Task-Models das Echtzeitverhalten und die Echtzeitanforderungen beschreiben und austauschen. Ohne zu viel IP offengelegt zu bekommen oder auf die Implementierung warten zu müssen, können die Integratoren frühzeitig das Systemverhalten analysieren.

AUTOSAR definiert seit Release 4.0, wie der Performance-Bedarf der SWC beschrieben wird und wie Wirkketten (siehe Kapitel 3.1) zu beschreiben sind. Somit kann eine Integration von SWC verschiedener Zulieferer als Task-Models, die mit diesen Angaben erstellt wurden, erfolgen.

Ein häufiges Problem bei herkömmlich durchgeführten Integrationen ist die Datenkonsistenz. Wurde beispielsweise eine SWC für eine Architektur mit kooperativem Scheduling entwickelt, kann sie bei der Integration in ein präemptives System inkonsistente Daten erhalten. Der Integrator kann den Datenfluss der SWC in der Simulation des Task-Models beobachten und schnell verschiedene Alternativen untersuchen, um das Problem zu vermeiden. Datenpufferung und verschiedene Verteilungen der Runnables der SWC auf die Tasks der neuen Software-Architektur werden Einfluss auf das Alter der Daten und den Jitter der Funktionen haben.

2.4 Risikobewertung von Echtzeitkriterien während des Entwicklungsprozesses

Für das Gesamtsystem und die Komponenten werden die Echtzeitanforderungen zusammen mit den Daten für die Modelle erfasst. Mit Hilfe der virtuellen Integration werden die bekannten Echtzeitanforderungen überprüft und es wird eine Risikobewertung durchgeführt. Echtzeitanforderungen, die nach herkömmlicher Methodik erst spät detailliert bekannt waren, werden rechtzeitig ermittelt und überprüft. Mit der frühen Bewertung werden potentielle Risiken und konkrete Probleme adressiert und in der Projektleitung von Anfang an berücksichtigt. Eine Übersicht des Risiko-Levels gibt der Projektleitung die Möglichkeit, die Risiken im Fokus zu behalten und entlang des Entwicklungsprozesses (siehe Bild 3) immer wieder zu adressieren.

Real-Time Criterion	Risk	Observe
Response times 1	●	max rate IRQ (CAN, sensor 3)
Response times 2	●	WCET of component 4
Response times 3	●	preemption through 10ms Task
Response times 4	●	execution time SW component 15
Start-to-Start Jitter SW component 1	●	jitter of SW component 4
Start-to-Start Jitter SW component 2	●	asynchronous activation of sensor 7 task
Latencies of critical event chain 1	●	sending data on SPI
Latencies of critical event chain 2	●	burst length chassis CAN for ID 596
Data loss	●	multi activation of 10ms Task
Data consistency 1	●	Wrapper for CAN messages
Data consistency 2	●	asynchronous (ECU-1) – synchronous (FlexRay) – asynchronous ECU-2
Multiple usage of data 1	●	preemption through 5ms task
Loss of IRQs	●	Σ WCET (blocking IRQs + execution time ISR)

Bild 2: Echtzeitkriterien mit Risikobewertung

Eine Kategorisierung des Risikos in Grün (konzeptionell gelöst), Gelb (muss beobachtet werden) und Rot (Kriterium wird nicht erfüllt) erleichtert die Projektsteuerung. Insbesondere in der Anfangsphase sind viele Details der Komponenten noch nicht bekannt. Daher kann es sich anbieten, eine vierte Kategorie für ‚kann noch nicht bewertet werden‘ einzuführen.



Bild 3: Risikobewertung entlang des Entwicklungsprozesses

3 Analyse des Echtzeitverhaltens

Zur Analyse des Echtzeitverhaltens wird das Task-Model des Gesamtsystems mit der INCHRON Tool-Suite simuliert (chronSIM) und validiert (chronVAL). Das Task-Model wird angeregt durch Stimuli wie Interrupts oder Restbussimulation. Durch die dynamische Reaktion des Systemmodells auf die Stimuli werden die Tasks aktiviert und verdrängt, die Runnables ausgeführt und Daten übertragen. Das Echtzeitverhalten der Komponenten kann beobachtet und analysiert werden. Die Echtzeitanforderungen des Gesamtsystems werden dabei automatisch überprüft. (Siehe Bild 4) [3]

Eine erfolgreiche Anwendung dieser Methodik, vor einer Implementation und ohne Target-Hardware, ist bereits mehrfach erfolgreich durchgeführt worden. [4]

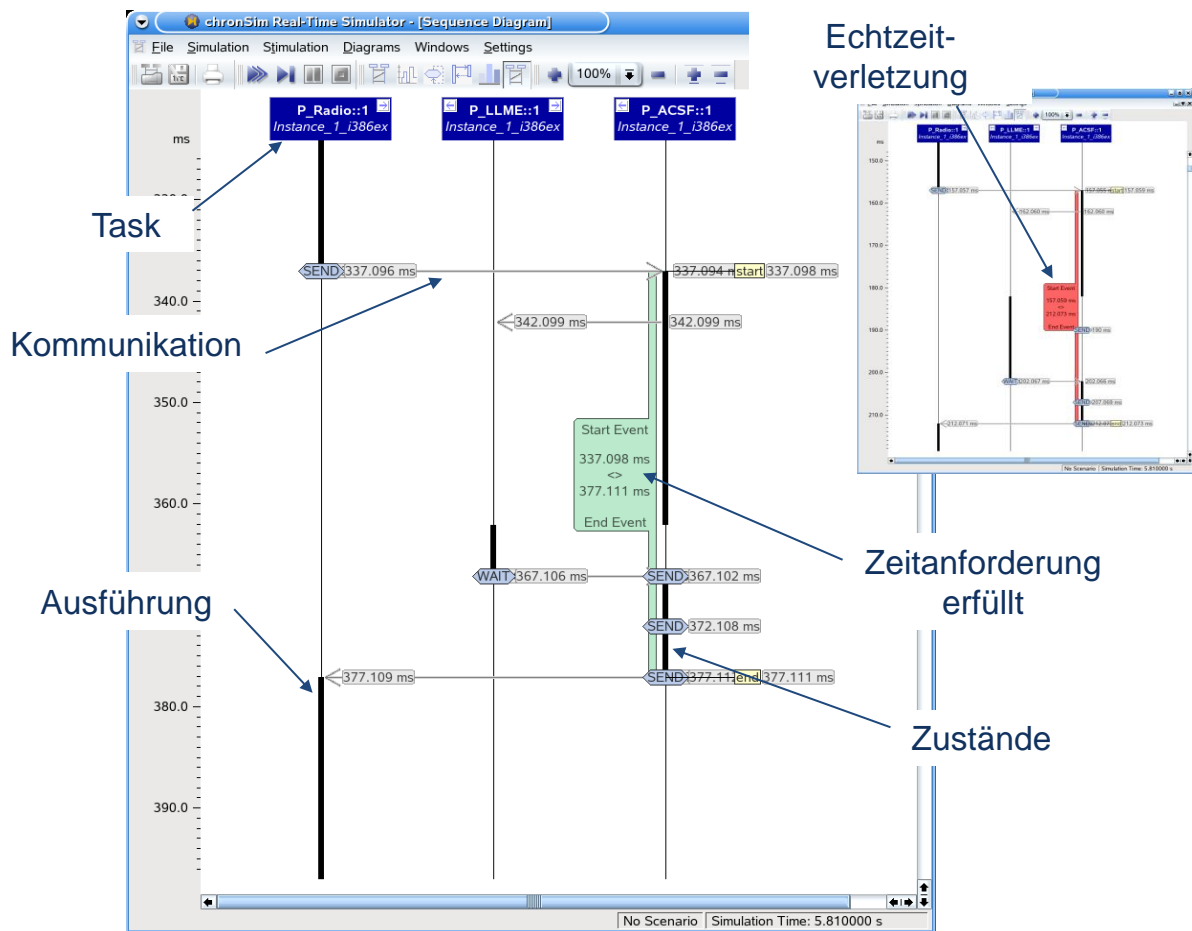


Bild 4: Sequenzdiagramm mit Echtzeitanforderungen

3.1 Wirkketten

Eine Funktion kann auch mit Hilfe von Wirkketten (Event Chains) [5] beschrieben werden. Sie verlaufen von Sensoren und Eingängen über verschiedene Hardware- und Software-Komponenten (Mechatronik, Bussysteme, Treiber-Software etc.) zur Applikations-Software, wo aus den Eingangsdaten und Systemzuständen eine Reaktion des Systems ermittelt wird. Dies wird wieder über verschiedene Komponenten zu den Aktoren übertragen. Die Reaktion hängt dabei von vielen Faktoren ab. Zustandsautomaten müssen in definierten Zuständen sein, Eingangswerte in vorgegebenen Wertebereichen und all dies zum richtigen Zeitpunkt. Da viele Wirkketten für verschiedene Funktionen gemeinsam ablaufen, interagieren und interferieren sie häufig. Diese Zusammenhänge statisch zu beschreiben, ist bereits eine komplexe Aufgabe. Im dynamischen, zeitlichen Zusammenhang betrachtet, ist eine Beherrschung des Systems nur mit Hilfe von Echtzeit-Simulationsmodellen möglich [6].

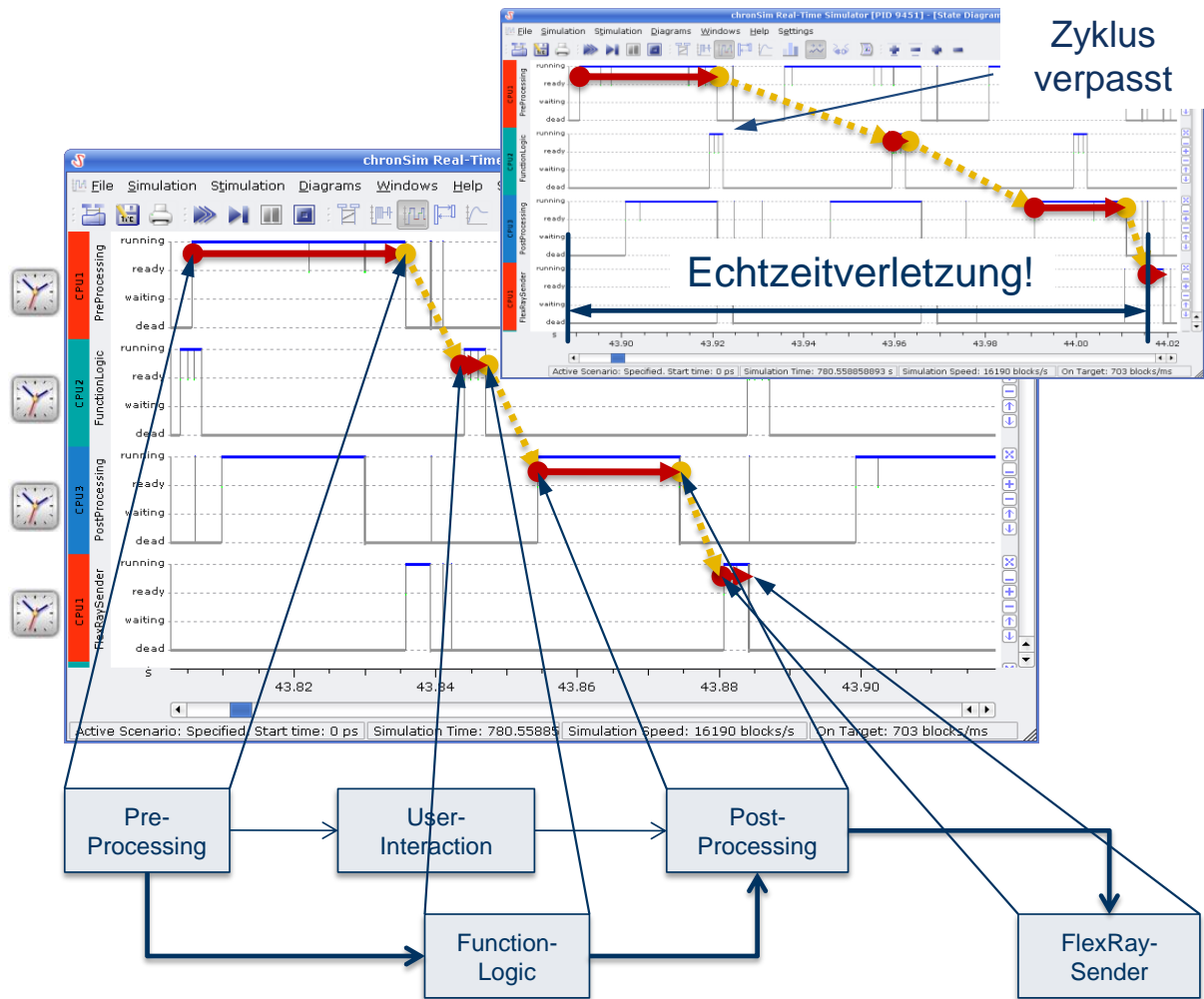


Bild 5: Wirkketten in Task-Zustandsdiagrammen

In einem Task-Model wird neben dem Rechenzeitbedarf der Funktionen auch die Wirkkette markiert. In Bild 5 sind vier Tasks modelliert und im Task-Zustandsdiagramm vom chronSIM in der Simulation dargestellt. Abhängig von der Phasenlage der periodischen Tasks auf verschiedenen CPUs zueinander, wird die Wirkkette evtl. stark verzögert abgearbeitet.

Mit Hilfe der Task-Models und Wirkketten können die Ablaufdetails der Einzelkomponenten (und damit die verteilte Detailkenntnis jeder Entwicklergruppe) in einem gemeinsamen Simulationsmodell erfasst werden. Die Simulation der Wirkketten hilft allen Beteiligten, die Abläufe im Ganzen besser zu verstehen und diejenigen Pfade durch das System zu identifizieren, deren Verhalten echtzeitkritisch sind [7].

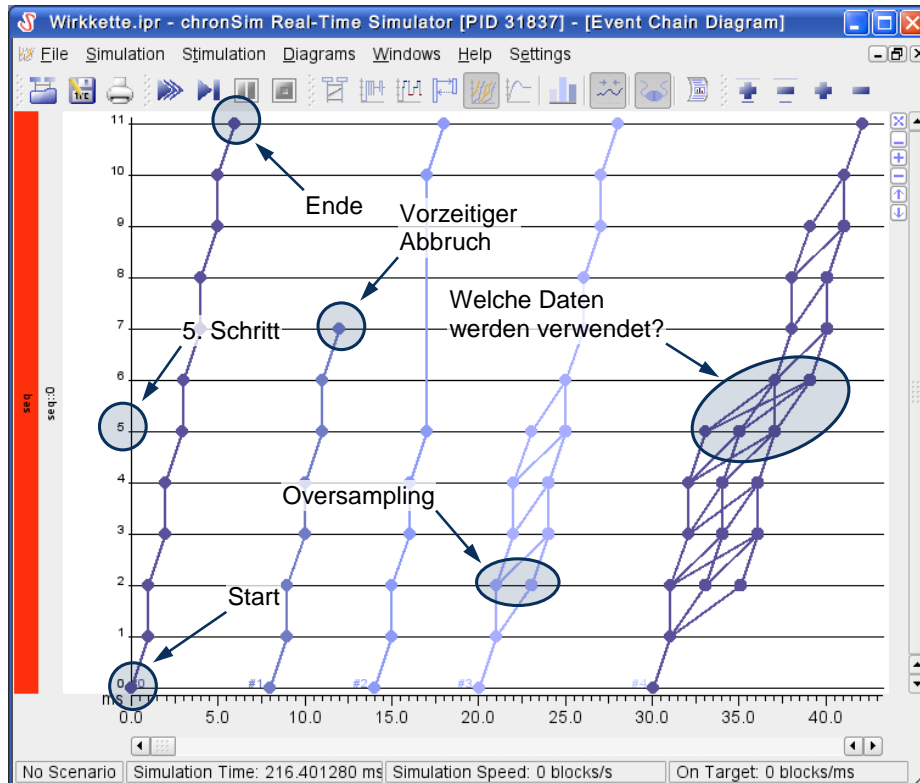


Bild 6: Wirkkettendiagramm

Bild 6 zeigt exemplarisch verschiedene Wirkketten. Horizontal ist die Zeit aufgetragen. In der Vertikalen sind die einzelnen Ereignisse/Funktionen aufgetragen, entlang derer die Wirkkette verläuft. Beispielsweise sind das der Empfang einer Busnachricht, die Übertragung des Signals an die Applikation oder das Zusammenfassen zu einer Nachricht. Wird ein Datum mehrfach verwendet (oversampling) ist das häufig unkritisch. Werden Informationen aber ungewollt verworfen oder das Alter der verwendeten Daten ist der Funktion unbekannt, kann eine kritische Wirkkette ihre Echtzeitanforderung ggf. nicht mehr erfüllen.

4. Zusammenfassung

Das Echtzeitverhalten von Software auf verteilten Steuergeräten frühzeitig zu beherrschen, wird immer wichtiger. Die steigende Komplexität der Systeme und der Entwicklung machen einen Wechsel zu einem modellbasierten Vorgehen zur frühen Architekturoptimierung unumgänglich. Die hier vorgestellte Methode zur virtuellen Integration ist in der Lage, den wesentlichen Aspekt, das korrekte Echtzeitverhalten, entlang des Entwicklungsprozesses sicherzustellen. Sie wurde bereits bei mehreren OEMs und Tier-1 erfolgreich eingesetzt. Eine Risikobewertung der Echtzeitkriterien mit Hilfe der virtuellen Integration sollte jeder Projektleiter auf seinem Dashboard zur Projektsteuerung haben.

Literatur

- [1] A. Wolfram, M. Makarov, T. Kramer, W. Ramisch, R. Münzenberger; Design of Robust System Architectures for Automotive ECUs; Conquest 2009, Nürnberg; www.isqi.org/conferences/conquest
- [2] M. Lokietsch; Performance-Analyse von UML-Systemen; Fachkongress Echtzeitentwicklung 2009, München; www.echtzeitkongress.de
- [3] T. Komarek, R. Münzenberger: Detecting timing violations and tracking lost tasks in an RTOS, ECE Magazine, Oktober 2006
- [4] M. Kessler; Absicherung der Echtzeitanforderungen in einem komplexen Steuergerät; BICC: Innovation Forum Embedded Systems 2010; www.ifes2010.de
- [5] AUTOSAR Release 4.0, AUTOSAR_TPS_TimingExtensions.pdf , www.autosar.org
- [6] T. Kramer, R. Münzenberger; New Functions, New Sensors, New Architectures – How to Cope with the Real-Time Requirements; Advanced Microsystems for Automotive Applications 2009, Berlin; www.amaa.de; ISBN 978-3-642-00744-6
- [7] R. Münzenberger, T. Kramer; Collaboration Support to Master Real-Time Challenges; embedded world congress 2009, Nuremberg