

Modellierung und Echtzeitanalyse komplexer Wirkketten in Fahrerassistenzsystemen

Tapio Kramer, Dr. Ralf Münzenberger

INCHRON GmbH
August-Bebel-Str. 88
14482 Potsdam

kramer@inchron.com
muenzenberger@inchron.com

Veröffentlicht im Vortragsband zur 3. AutoTest, FKFS,
27.+28. Oktober 2010, Stuttgart

Kurzfassung: Vorgestellt wird eine neue Methode um die Integration der Soft- und Hardware auf Steuergeräte sowie deren Integration in Fahrzeuge früh entwicklungsbegleitend testen zu können. Ein Prozess zur Spezifikation, Austausch und Überprüfung von Echtzeitanforderungen zwischen OEM und Zulieferern wird durch diese Methode wesentlich unterstützt. Die vorgestellte Methode zusammen mit den jüngst in AUTOSAR 4.0 spezifizierten Timing-Parametern und Event-Chains ermöglichen es, basierend auf einem Standard eine virtuelle Integration von Komponenten, insbesondere von Fahrerassistenzsystemen, durchzuführen.

1 Ursachen und Auswirkungen von Echtzeitproblemen auf Fahrerassistenz- und Fahrzeugregelsysteme

Die Kombination von verschiedenen Sensoren, Aktoren und Reglern in Steuergeräten mittels Sensordatenfusion, Objektverifikation und komplexen Funktionen ermöglicht es leistungsfähige Fahrerassistenzsysteme zu realisieren. Das führt zwangsläufig zur Kombination von asynchronen, periodischen und ereignisgesteuerten Prozessen und Kommunikationen, die wiederum komplexe Wirkketten hervorrufen. Um in der Architekturphase und in späteren Tests das Zeitverhalten solcher End-to-End-Wirkketten sicher vorsagen und testen zu können, sind neben Funktionstests auch Analysen des Zeitverhaltens notwendig. Denn die Ausführungszeiten entlang der Wirkketten sind stark abhängig vom Scheduling, von der momentanen Auslastung der Target-Hardware und der Bussysteme.

Ein Abstandsregelsystem verwendet beispielsweise Kamera und Radar basierte Sensoren zur Objekterkennung. Beide Sensortypen liefern Ihre Daten unabhängig mit verschiedenen Abtastraten, die zudem asynchron zum Prozessortakt, und somit zum Takt der zeitgesteuerten Applikations-Tasks im Steuergerät, anfallen. Bei einer

Sensordatenfusion ist es wichtig, die Erfassungszeitpunkte der Daten und der darin erkannten Objekte genau zu berücksichtigen. Die Objektklassifikation und -verifikation kann nur korrekt erfolgen, wenn die Zeitbezüge sicher berücksichtigt werden können.

Entlang der Wirkketten durch das System werden die Daten erfasst, verarbeitet und ausgewertet, und die Aktuatoren angesteuert. Filter- und Reglerfunktionen arbeiten mit Annahmen bezüglich des Alters der Daten und der Präzision der Zeitangaben. In einem nicht synchronisierten, zeitgesteuerten, verteilten System können aufgrund von driftenden Uhren Schwebungseffekte auftreten, die zu Mehrfachverarbeitung aber auch Verlust von Daten führen können. Solche Echtzeitprobleme können nur durch eine Analyse des dynamischen Verhaltens erkannt werden. Eine rein funktionale Betrachtung ist nicht ausreichend.

Mit der zunehmenden Komplexität der implementierten Funktionen und der gleichzeitig steigenden Vernetzung nimmt der Aufwand zur Suche und Behebung von Echtzeitproblemen stark zu. Selbst wenn jedes Steuergerät für sich keine Echtzeitprobleme aufweist, können beim Zusammenspiel mehrerer Steuergeräte derartige Echtzeitfehler auftreten.

2 Frühzeitige Spezifikation und Berücksichtigung der Echtzeitkriterien bei der Architekturauswahl

Bereits im Vorfeld der Implementierung ist es somit unabdingbar, für die Architekturanalyse die Echtzeitkriterien und zeitkritischen Wirkketten durch das System zu kennen. Beim Design der Software- und Hardware-Architektur sind solche dynamischen Effekte zu berücksichtigen, um eine zuverlässige und rechtzeitige Ausführung der Funktionen sicherzustellen. Die detaillierte Spezifikation der Echtzeitanforderungen und -eigenschaften der Funktionen und Wirkketten ist also eine Voraussetzung für eine robuste Systemarchitektur. Die Spezifikation der Echtzeitanforderungen muss während der Anforderungsphase, d.h. noch vor der Architekturphase, erfolgen. Sie muss über das ganze System, und damit häufig auch über Domänen- und Unternehmensgrenzen hinweg erfolgen. [Aug10]

Das Verhalten der Schnittstellen von Software-Komponenten dürfen hierbei nicht nur statisch, sondern es müssen auch die dynamischen Anforderungen spezifiziert sein. Es ist erheblich, ob eine Schnittstelle Teil einer Wirkkette ist, wie alt die Daten sind oder ob die Daten ohne Datenverlust/-verdopplung hierüber transportiert werden müssen. Da die Teilkomponenten häufig von verschiedenen Zulieferern kommen und auch die Software-Komponenten innerhalb der Steuergeräte von unterschiedlichen Entwicklungsgruppen erstellt werden, kommen der Schnittstellendefinition und dem gemeinsamen Systemverständnis große Bedeutung zu. In [KrMü10] wird auf die virtuelle Integration und deren Vorteile näher eingegangen.

In dem hier vorgestellten modellbasierten Ansatz werden neben der Systemarchitektur die Echtzeiteigenschaften und -anforderungen einschließlich der Wirkketten in einem Task-Model modelliert. Hierbei werden die Software-Komponenten

auch von unterschiedlichen Zulieferern virtuell integriert, so dass in einer Echtzeit-simulation oder Validierung das dynamische Verhalten des Gesamtsystems überprüft werden kann. Dies ist beispielsweise mit dem Echtzeitsimulator chronSIM oder dem Validator chronVAL des Unternehmens INCHRON möglich. Es vermeidet kosten- und zeitaufwändige Re-Design Zyklen, da bereits in der Architekturphase eine optimierte Lösung gefunden wird. In [Wolf09] wird dieser Vorteil bei der Entwicklung eines Body-Steuergerätes aufgezeigt. Für die Abstimmung mit dem OEM über erweiterte Funktionalität konnte hier gezeigt werden, dass 2/3 der Zeit gegenüber den herkömmlichen Ansätzen gespart werden konnte.

3 Echtzeitprobleme und Echtzeitkriterien

Das korrekte Echtzeitverhalten von Systemen hängt je nach gewählter Software-Architektur von verschiedenen Kriterien ab. Für eine Analyse ist es daher notwendig, sowohl die relevanten Echtzeitkriterien der Teilkomponenten als auch die des Gesamtsystems zu kennen. Im oben genannten Beispiel mit Radar- und Kameradaten treten durch die Asynchronität der Sensoren zueinander und zur Datenverarbeitung typische Echtzeitprobleme durch Schwebungseffekte auf. Bus-Nachrichten verdrängen sich gegenseitig, wenn Phasenlage und Periode der Übertragungen ungünstig sind und daher die Daten manchmal gleichzeitig anliegen. Die Reihenfolge und Zykluszeit der Daten auf Empfangsseite variiert folglich. Die Datenfusion erfolgt nicht sicher zu dem Zeitpunkt, wenn beide Sensoren aktuelle Daten geliefert haben. Die Schwebungseffekte bewirken dann, dass evtl. einer der Sensoren zum Start der asynchronen Datenfusion noch keine oder bereits zwei aktualisierte Daten lieferte. Durch die geringe Uhrendrift kann dies in manchen Systemen sehr selten auftreten, vergleichbar mit Sonnenfinsternissen, die zwar periodisch auftreten, dann aber nicht immer am selben Ort auf der Erde zu beobachten sind.

Abbildung 1 zeigt in einem Task-Zustandsdiagramm die Datenverarbeitung entlang einer zeitkritischen Wirkkette. Das System besteht aus drei CPUs mit jeweils zeitgesteuerten Tasks. Da die Taktgeber der CPUs nicht synchronisiert sind, driften die Uhren zueinander. Die Verarbeitung, die auf CPU1 (Pre-Processing, links oben) beginnt, wird auf CPU2 (Function Logic) und CPU3 (Post-Processing) fortgesetzt. Driften die Zyklen auf CPU1 und CPU2 zueinander, kann der Verarbeitungsschritt auf CPU1 eventuell seine Daten erst im nächsten Zyklus der CPU2 weiter geben (im Bild oben rechts), so dass sich die Antwortzeit um die Periode der Task auf CPU2 verlängert. Dies führt dazu, dass trotz der korrekten Berechnung des Post-Processing-Algorithmus ein falscher Wert auf dem FlexRay-Bus gesendet wird, da die Berechnung auf der Basis von zu alten Daten erfolgt. Dieses sich manchmal sprunghaft ändernde Alter des Datums muss als Echtzeiteigenschaft der Architektur in der Datenverarbeitung berücksichtigt werden – oder mit geeigneten Maßnahmen verhindert werden.

Ein weiterer Effekt sind Datenverluste und ungewollte Mehrfachverarbeitung. Selbst wenn Messwerte genau so oft verarbeitet wie erzeugt und übertragen werden, kann

eine Asynchronität an den Schnittstellen dazu führen, dass ein Datum verpasst oder doppelt verwertet wird (Siehe Kapitel 6 Abbildung 4). Einfach nur die Daten mehrfach abzutasten, zu übertragen und zu verarbeiten schafft i.d.R. keine Abhilfe, da teilweise mit alten Daten gearbeitet und das System unnötig zusätzlich belastet wird – und ein Datenverlust dennoch nicht garantiert vermieden wird. Daher muss auch hier mit geeigneten Echtzeitkriterien im Vorfeld sichergestellt werden, dass die Architektur die Anforderungen der Datenverarbeitung erfüllen können wird.

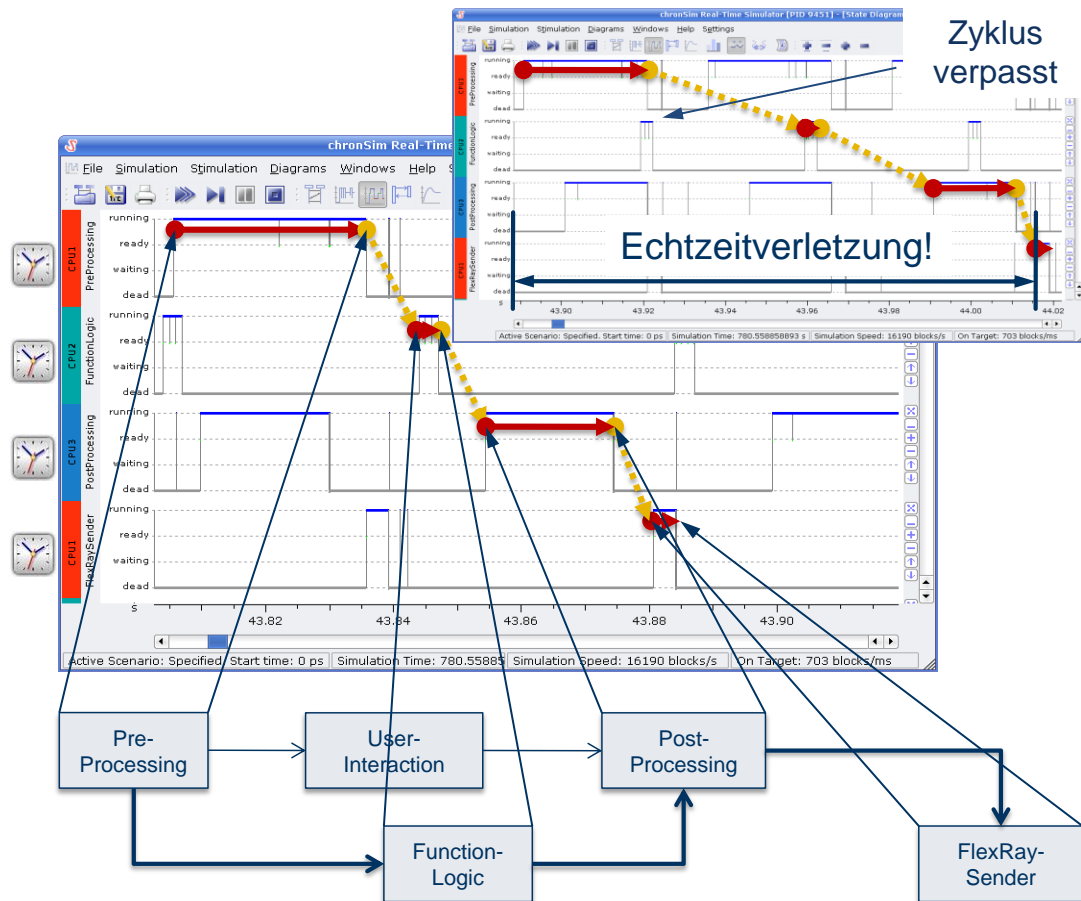


Abbildung 1: Task-Zustandsdiagramm mit Wirkkette. Antwortzeit wird links eingehalten, oben nicht.

Allgemein formulierte, typische Echtzeitkriterien sind:

- Antwortzeiten von Funktionen, Tasks, Wirkketten und Systemen
- Start-zu-Start Jitter von Tasks und Software-Komponenten
- Mehrfachaktivierung von Tasks
- Ausführungsreihenfolge von Software-Komponenten
- Latenzen von Wirkketten (Event Chains)
- Alter und Konsistenz der Daten

- Verlust oder Wiederverwendung von Daten (gewollt oder unbeabsichtigt)
- Verlust oder Blockierung von Interrupts

4 Bedeutung der AUTOSAR Methodik für Entwicklung und Test komplexer, echtzeitkritischer Systeme

Die Tendenz zu komplexeren Systemen führte zur Entwicklung von AUTOSAR [AR]. Es stellt eine modulare und konfigurierbare Entwicklungsplattform mit standardisierten Basisfunktionen bereit, die die Entwicklung von Applikationen im Fahrzeug insgesamt erleichtern und die Qualität der E/E Systeme verbessern soll. Die strikte Trennung von Applikations- und Basissoftware, die einheitlichen (standardisierten) Kommunikationsschnittstellen sowie die Definition standardisierter Dienste (Services) und Basisfunktionen ermöglichen dies.

AUTOSAR erleichtert die Austauschbarkeit von Software-Komponenten (SW-C) verschiedener Zulieferer, was wiederum den Bedarf an genauer Spezifikation des Echtzeitverhaltens und der Verbesserung des gemeinsamen Systemverständnisses erhöht. Dem wird nun mit den Timing Extensions von AUTOSAR Release 4.0 teilweise Rechnung getragen.

Wurde eine SW-C zuvor entwickelt um auf einer Software-Architektur ohne prioritätenbasierte Verdrängung zu laufen, muss sie in einer Umgebung mit Verdrängungen die Datenkonsistenz bei Unterbrechungen selbst sicher stellen oder als Anforderung an die Architektur bekannt geben. Kommunikation, die innerhalb eines Steuergerätes sehr schnell abläuft, wird bei Verteilung der kommunizierenden Komponenten auf mehrere Steuergeräte deutlich mehr Latenz und Jitter zeigen. Diese dynamischen Echtzeiteigenschaften der neuen Architektur und Echtzeitanforderungen der Komponenten müssen beschrieben und miteinander abgeglichen werden.

Mit den Timing Extensions wurden weitere Möglichkeiten geschaffen das Zeitverhalten der SW-C zu spezifizieren und auch Wirkketten (Event Chains) zu definieren [AR TimExt]. Es können Anforderungen bezüglich Jitter, Latenz, Synchronisation usw. an Funktionen und Event Chains formuliert werden. Event Chains beschreiben eine kausale Verkettung von zeitlichen Systemereignissen. Jede Event Chain hat eine klar definierte Anregung und Reaktionen, die Anfang und Ende festlegen und sie ist hierarchisch aus Event Chain Segmenten zusammengesetzt. So können nun Anforderungen formuliert werden, welche Zeit maximal vergehen darf, bis ein Datum verarbeitet wurde, beispielsweise gemessen ab dem Zeitpunkt der Datenerfassung. Bezogen auf das oben genannte Beispiel Abstandsregelung könnte hiermit das maximale Alter eines Kamerabildes definiert werden, dass mit einem Radarbild zur Objektverifikation korreliert wird. Daraus leiten sich dann weitere Echtzeitanforderungen an die Radardatenerfassung oder Nachrichtenübertragung ab.

5 Ansatz zur Absicherung des Echtzeitverhaltens mittels frühzeitiger, virtueller Integration

Zur virtuellen Integration der Teilkomponenten in das Gesamtsystem wird ein Systemmodell benötigt, das das Echtzeitverhalten abbildet. Das dynamische Zusammenspiel der Komponenten wird in diesem Systemmodell beschrieben (siehe Abbildung 2). Die Software-Architektur (Scheduling-Verfahren, Prioritäten, Anzahl von Tasks, Task-Aktivierungen, etc.) und Hardware (Ressourcen: CPUs und Bussysteme) sowie deren Stimuli (Interrupts, Busnachrichten, Aktivierungen) gehen mit in das Modell ein. Das Systemmodell wird anschließend mittels Simulation und Validierung analysiert und abgesichert.

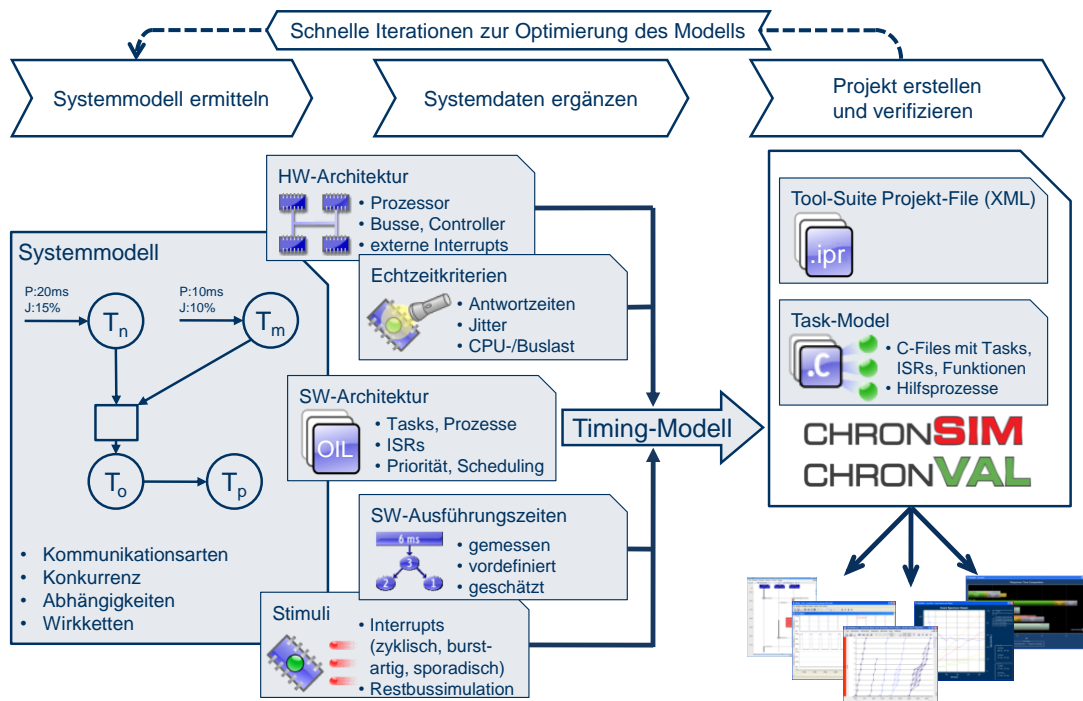


Abbildung 2: Vom Systemmodell zum Timing-Modell und der Analyse des Echtzeitverhaltens

Mit den Task-Models bietet die INCHRON Tool-Suite die Möglichkeit, das Echtzeitverhalten eines Gesamtsystems und der Teilsysteme mit verteilten Ressourcen zu modellieren, zu simulieren (chronSIM) und zu validieren (chronVAL). 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 Abbildung 3 links). In das Gesamtsystemmodell werden die Teilmodelle der Steuergeräte und Bussysteme integriert. Die Modellierung kann alternativ in UML (Unified Modeling Language) erfolgen [Nick10].

Im Verlauf der Entwicklung werden die Annahmen und unvollständig bekannten Anforderungen aus der Architekturphase mit Eigenschaften aus der Implementierungsphase ersetzt. Die Zeitbudgets können feingranular mit detaillierten Verhaltensmodellen (siehe Abbildung 3 rechts) auch 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. Als Prozess begleitende Analyse der Echtzeiteigenschaften mittels Simulation wird dieses Vorgehen in [Stük09] aufgezeigt.

Durch diese modellbasierte, virtuelle Integration der Komponenten in das Systemmodell wird die Integration in frühe Phasen der Entwicklung vorverlegt. Modelle auf verschiedenen Abstraktionsebenen werden dazu verwendet das System und seine Komponenten auf ihre Integrierbarkeit und Echtzeitfähigkeit zu überprüfen. Modellkomponenten aus der Spezifikation und aus bereits erfolgter Implementation können kombiniert und ihr dynamisches Verhalten gemeinsam simuliert und validiert werden. So wird noch vor der Fertigstellung der Hardware- und Software-Komponenten, der Test der Einhaltung von Echtzeitkriterien ermöglicht.

| | | |
|---|--|---|
| <pre>ISR(ISR_Rotation) { DELAY(30,unit_us); ActivateTask(Task_Rot); }</pre> | <pre>TASK(Task_Rot) { DELAY(150,unit_us); TerminateTask(); }</pre> | <pre>TASK(TASK_TT_5ms) { DELAY(300, unit_us); ① Schedule(); ② while(!finished()) { DELAY(100, unit_us); } ③ exectime = 10*data_size; DELAY(exectime,unit_us); ④ DELAY(gaussian(500, 10),unit_us); ⑤ TerminateTask(); }</pre> |
| <p>Beispiel des einfachsten Task-Modells für OSEK.</p> <p>ISR_Rotation wird in 30 µs und Task_Rot in 150 µs ausgeführt.</p> | | <p>Beispiel eines komplexeren Task-Modells</p> <p>① : konstante Ausführungszeit ② : Aufruf des Schedulers ③ : ablaufabhängige Ausführungszeit ④ : datenabhängige Ausführungszeit ⑤ : wahrscheinlichkeitsverteilte Ausführungszeit</p> |

Abbildung 3: Modellierung der Zeitbudgets im Task-Model

6 Definition und Analyse von zeitkritischen Wirkketten

Die Qualität der Integration und Systemarchitektur wird an den Wirkketten deutlich. Häufig ist die Fahrzeugfunktion (,Bremseneingriff, da Objekt auf Fahrbahn‘ oder ,Wankregelung mittels aktiver Dämpfer‘) über eine kritische Wirkkette definiert. So ist der rechtzeitige Bremseneingriff eines Fahrerassistenzsystems abhängig davon, wie schnell der kritische Pfad von der Bild-/Radardatenerfassung zum Bremsen durchlaufen wird. Oder die Wankregelung kann nur so schnell und fein erfolgen, wie der

Regelkreis aus Einfederungsmessung, Fahrsituationsbeurteilung bis Dämpferhärteverstellung durchlaufen werden kann.

Eine große Bedeutung kommt daher der Modellierung und Analyse der Wirkketten zu. Wirkketten beschreiben den Datenpfad durch das System, wo die Datenweitergabe (auch über Bussysteme) und die Datenverarbeitung stattfinden. Nur die Aktivierungskanten der Tasks miteinander zu verbinden ist im Allgemeinen nicht ausreichend, da eventuell auch mehrere Datenverarbeitungs- und -weitergabepunkte innerhalb einer Task liegen können oder die Daten schon vor dem Ende der Task in einer anderen Task weiterverarbeitet werden.

Die gemeinsam definierten Wirkketten und kritischen Pfade durch das System, von Sensoren zu Aktoren, über Bussysteme und durch Software-Schichten, helfen den Beteiligten (OEMs, Zulieferer, Dienstleister) ein gemeinsames Gesamtsystemverständnis zu erhalten [Aug10]. Die resultierende, verbesserte Kollaboration ermöglicht Optimierungen über das Gesamtsystem hinweg, deutlich exaktere Vorhersage des Projektverlaufs und Systemverhaltens und senkt somit das Projektrisiko erheblich [KraMü10].

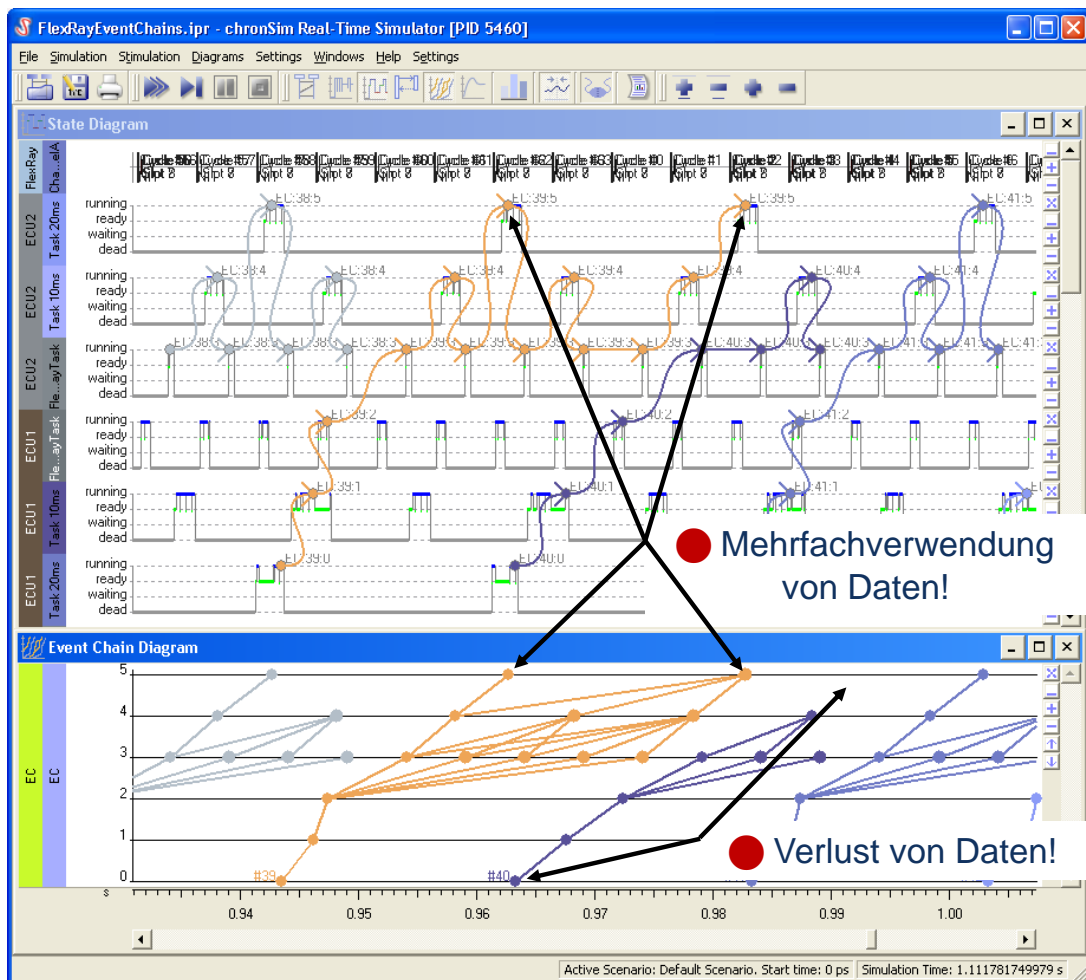


Abbildung 4: Task-Zustandsdiagramm und Wirkkettendiagramm zur Darstellung des Datenverlustes wegen der asynchronen Kommunikation zweier Steuergeräte über FlexRay

Für die Spezifikation des Datenflusses und der Aktivierungskanten sind die in der AUTOSAR Timing Extension beschriebenen Event Chains hinreichend. Für die genaue zeitliche Analyse mittels Echtzeitsimulation wird in der INCHRON Tool-Suite auch noch ein Token für jede Instanz einer Wirkkette eingeführt und analysiert. Somit ist man in der Lage auch zu detektieren, wann welche Daten überschrieben werden oder wie abgebrochene Wirkketten den Datenverlust auf der Empfangsseite hervorrufen.

In Abbildung 4 ist die asynchrone Kommunikation zweier Steuergeräte über FlexRay dargestellt. Die Daten der Applikation in der 20ms-Task auf ECU1 (im Bild im oberen Diagramm unten) werden an die 10ms- und anschließend an die zum FlexRay synchrone 5ms-Task übermittelt. Von ECU2 werden die Daten empfangen und in einer 10ms- und 20ms-Task weiter verarbeitet. Da zum einen die Applikationen asynchron zum FlexRay laufen und zudem noch Mehrfachabtastungen der Daten bei der Vorverarbeitung durch die 10ms-Task ECU2 in der Architektur vorgesehen

wurde, kommt es zu Echtzeitproblemen. Im Bild ist dargestellt, wie in einem Zyklus dieselben Ausgangsdaten mehrfach von der Applikation (oben) verarbeitet werden. Darauf folgt aber gleich auch ein Datenverlust. Das Wirkkettendiagramm in der unteren Bildhälfte zeigt dieselben Wirkketten in übersichtlicher Darstellung und erleichtert das Auffinden von Datenverlusten oder besonders hervortretenden Antwortzeiten der Wirkketten. Eine detaillierte Beschreibung zu den Timing-Effekten mit asynchronen FlexRay-Architekturen findet sich auch in [Münz08], wo von Siemens VDO eine Motorsteuerung mit FlexRay-Kommunikation untersucht wurde.

Die Vorteile bereits frühzeitig die Optimierung einer Software-Architektur gemäß diesem Ansatz durchzuführen, wurde in [Kess10] bei der Applikation eines Body-Steuergerätes von Lear für BMW beschrieben. Es konnten „erste brauchbare Ergebnisse bereits nach zwei Tagen“ ermittelt werden. Auf Grund der sehr kurzen Turnaround-Zeiten konnte mit einem Gesamtaufwand von zwei Wochen (inklusive Systemanalyse und Laufzeitmessung) die Optimierung abgeschlossen werden.

7 Zusammenfassung

Insbesondere in Fahrerassistenzsystemen hat sich gezeigt, dass Echtzeitfehler häufig erst spät auftreten und der Aufwand zur Suche groß ist. Die Komplexität dieser integrierten Systeme ist teilweise nicht mehr beherrschbar. Deshalb ist es notwendig die Echtzeitanforderungen bereits in der Anforderungsphase für jede Software-Komponente zu definieren. So kann bereits in der Architektur-Phase das System auf das Echtzeitverhalten und die Performance hin optimiert werden - lange bevor die fortgeschrittene Implementation und Integration die Redesigns teuer und aufwändig machen. Diese frühe Echtzeitanalyse wird ermöglicht durch den beschriebenen modellbasierten Ansatz, in dem eine virtuelle Integration des Systems mit anschließender Simulation und Test durchgeführt wird. Das Modell wird projektbegleitend verfeinert und verringert das Projektrisiko erheblich, da zu jeder Zeit die Integration überprüft und Echtzeitprobleme erkannt werden.

Die aufgezeigten Probleme treten häufig nicht nur in Fahrerassistenzsystemen auf sondern zunehmend auch in Body-Steuergeräten, die eine Vielzahl von Kundenfunktionen und häufig auch Gateway-Aufgaben beinhalten. Vermehrt anspruchsvoll werden aber auch Safety-Steuergeräte, in die mehr und mehr Fahrerassistenzfunktionen mit kritischen Sensordatenfusionen integriert werden.

Die Methode eignet sich sehr gut für Zulieferer mit einem Steuergerät, dass im Fahrzeug mit anderen Komponenten vernetzt wird. In Kollaborationsprojekten mit mehreren Zulieferern und IP-Schutz-Anforderungen bietet die Abstraktion der Modelle ideale Voraussetzungen für ein gemeinsames, gutes Systemverständnis. Sie wurde von mehreren Unternehmen in unterschiedlichen Domänen erfolgreich eingesetzt.

8 Literaturverzeichnis

- [Aug10] B. Augustin; Integration von bisher eigenständigen Steuergeräten und Funktionen – Herausforderungen eines Kollaborationsprojektes; 2. Fachkongress Echtzeitentwicklung 2010; www.echtzeitkongress.de
- [Wolf09] A. Wolfram, M. Makarov, T.Kramer, W. Ramisch, R. Münzenberger; Design of Robust System Architectures for Automotive ECUs; In proceedings of Conquest 2009, Nuremberg; www.isqi.org/konferenzen/conquest/2009/
- [AR] AUTOSAR, AUTomotive Open System ARchitecture, www.autosar.org
- [AR TimExt] AUTOSAR 4.0 Timing Extensions; www.autosar.org, AUTOSAR_RS_TimingExtensions.pdf.
- [Nick10] U. Nickel, J. Meyer; Wie hoch ist die Performance?; Automobil-Elektronik, Ausgabe Juni 2010; www.automobil-elektronik.de
- [Stük09] Dr. Dirk Stüker, Gökhan Tabanoglu; Herausforderungen bei der Entwicklung multifunktionaler Steuergeräte aus OEM-Sicht; Fachkongress Echtzeitentwicklung 2009, München; www.echtzeitkongress.de
- [KrMü10] T. Kramer, Dr. R. Münzenberger; Absicherung des Echtzeitverhaltens mittels virtueller Integration; 4. Tagung - Simulation und Test für die Automobilelektronik, IAV, Mai/Juni 2010, Berlin; ISBN 978-3-8169-3023-5
- [Münz07] R. Münzenberger, M. Dörfel, C. Diederichs, U. Margull, G. Wirrer; Entwurf echtzeitfähiger Steuergerätesoftware in FlexRay-Netzwerken; In proceedings of the Kfz-Entwicklerforum 2007, Ludwigsburg; www.elektroniknet.de/index.php?id=2404
- [Kess10] M. Kessler; Absicherung der Echtzeitanforderungen in einem komplexen Steuergerät; BICC: Innovation Forum Embedded Systems 2010; www.ifes2010.de