Entwurf echtzeitfähiger Steuergerätesoftware in FlexRay-Netzwerken

R. Münzenberger, M. Dörfel, INCHRON GmbH
C. Diederichs, Universität Oldenburg
U. Margull, 1 mal 1 Software GmbH
G. Wirrer, Siemens AG (Bereich SiemensVDO Automotive)

Ein Gesamtsystem von mehreren Steuergeräten, die über den FlexRay-Bus verbunden sind, verspricht Echtzeitfähigkeit. Aber, um diese Echtzeitfähigkeit zu realisieren, gilt es sowohl beim Entwurf der Softwarearchitektur eines neuen Steuergerätes als auch bei der Migration eines bestehenden Steuergerätes auf den FlexRay-Bus, das durch die FlexRay-Kommunikation vorgegebene zeitliche Verhalten zu berücksichtigen. In diesem Artikel werden dabei entstehende typische Problemstellungen beim Entwurf von FlexRay-fähigen Steuergeräten vorgestellt. Die Anwendungsstudie zeigt die Realisierung der Echtzeitfähigkeit einer verteilten Motorsteuerung auf.

Einleitung

In Automobilen nimmt die Vernetzung von Steuergeräten, den so genannten Electronic Control Units (ECU), erheblich zu. Mit der steigenden Vernetzung, etwa für X-by-wire-Systeme, sind ein deterministischer Buszugriff sowie eine garantierte maximale Latenzzeit bei der Übertragung notwendig: Zwar werden Fahrzeugfunktionen verteilt ausgeführt, aber die Echtzeitfähigkeit des Gesamtsystems muss auch über ECU-Grenzen gewährleistet werden. Das CAN-Bussystem ist durch den asynchronen Buszugriff nicht streng deterministisch und kann nur für die höchst priorisierte Botschaft eine maximale Latenzzeit garantieren. Das Bussystem FlexRay dagegen unterstützt beide Anforderungen, indem einer ECU ein periodisch wiederkehrender Zeitslot zum Senden von Nachrichten zugewiesen wird.

Die Vorteile von FlexRay wurden in der Serienproduktion erstmals im aktiven Dämpfersystem des BMW X5 SAV genutzt [1]. Darüber hinaus existieren Auftragsstudien, darunter ein Steer-by-Wire-System für den japanischen Automobilzulieferer DENSO [2] und die Fahrversuche mit der neuen elektronischen Brake-by-Wire-Keilbremse von Siemens VDO [3]. Auch die zentrale Chassis-Dynamik-Steuerung von Siemens VDO ist mit den anderen Chassis-Systemen durch einen oder mehrere FlexRay-Bus-Systeme vernetzt [4]. Jedes dieser genannten Beispiele enthält ein FlexRay-Cluster aus vier bis sechs Steuergeräten.

Bei der Fragestellung, ob ein FlexRay-Cluster Echtzeitfähigkeit mitbringt, beschränken sich viele Untersuchungen zunächst darauf, das Echtzeitverhalten jedes Steuergerätes einzeln zu betrachten. Dies aber führt nicht immer zu tragfähigen Ergebnissen, wie im folgenden Abschnitt anhand von typischen Echtzeitproblemen beim Entwurf von FlexRay-Clustern aufgezeigt wird. Ist die Analyse des Echtzeitverhaltens *einer* ECU bereits eine komplexe Aufgabe, gestaltet sich das Auffinden von

Echtzeitfehlern in einem *FlexRay-Cluster* noch erheblich aufwändiger. Dann nämlich muss das dynamische Verhalten der ECU-Software gemeinsam mit dem zeitlichen Verhalten des FlexRay-Busses betrachtet werden. Dies ist mit dem Echtzeitsimulator chronSim [5] möglich. Eine solche Echtzeitanalyse ist stets unabhängig davon notwendig, ob es sich um eine Neuentwicklung handelt oder eine Migration eines bestehenden Systems nach FlexRay vorgenommen wird.

Die Problemstellungen

Echtzeitfähigkeit einzelner Steuergeräte

Das dynamische Verhalten von Steuergeräten und damit die Echtzeitfähigkeit werden maßgeblich durch die Softwarearchitektur beeinflusst. Dazu zählen beispielsweise die Anzahl von Tasks und Interrupt-Service-Routinen (ISR), die Zuordnung von Funktionalität auf Tasks und Interrupt-Service-Routinen, das Schedulingverfahren (präemptiv, kooperativ etc.) und die Scheduling-Strategie (priorisiert, round-robin etc.). Die Entwickler müssen die optimale Systemarchitektur finden, die eine Echtzeitfähigkeit des Systems gewährleistet.

Häufig wird in den ECUs eine Multitasking-Architektur eingesetzt. Diese besteht aus zeitgesteuerten Tasks (Time-Triggered-Tasks), die periodisch aufgerufen werden (beispielsweise alle 1 ms, 5 ms, 10 ms etc.), sowie aus Interrupt-Service-Routinen und Tasks (Event-Triggered-Tasks), die asynchron in Abhängigkeit von Ereignissen der Systemumgebung aktiviert werden (beispielsweise motordrehzahlabhängig Sensorsignale). Selbst wenn die Aktivierungen der Time-Triggered-Tasks und der Event-Triggered-Tasks in einem ungünstigen Verhältnis zueinander stehen, muss das Steuergerät stets echtzeitfähig sein. Dennoch können Echtzeitfehler auftreten: Beispielsweise dann, wenn eine asynchron aufgerufene Interrupt-Service-Routine eine zeitkritische Time-Triggered-Task so häufig unterbricht, dass die Deadline der Time-Triggered-Task überschritten wird.

Bereits kleine Änderungen der Software können zum Verlust der Echtzeitfähigkeit des Gesamtsystems führen. Wird beispielsweise eine zuvor echtzeitfehlerfreie Softwarearchitektur um eine Funktionalität erweitert und dadurch die Ausführungszeit einzelner Module verlängert, kann dies bereits zu einem Verlust der Echtzeitfähigkeit führen. Derartige Echtzeitfehler treten häufig nur sporadisch auf und sind deshalb schwer auffindbar. Deshalb ist ein systematisches Vorgehen bei der Analyse der Echtzeitfähigkeit notwendig, wie es der Echtzeitsimulator chronSim [6] [7] ermöglicht.

Echtzeitfähigkeit vernetzter Steuergeräte

Gestaltet sich der Entwurf von einzelnen Steuergeräten bereits als ein komplexer Vorgang, ist der Entwurf vernetzter Steuergeräte noch einmal erheblich aufwändiger. Besonderes Augenmerk bei der Konzeption von vernetzten echtzeitfähigen ECUs gilt dem verwendeten Bussystem. Der FlexRay-Bus [8] verspricht Echtzeitfähigkeit, da er einen Buszyklus in ein statisches und ein dynamisches Segment unterteilt. Das **statische Segment** besteht aus Zeitslots fester Größe, in denen jeweils eine ECU *eine* Nachricht mit *einer* garantierten Latenzzeit übertragen kann. In welchem Zeitslot eine ECU eine Nachricht übertragen darf, wird vom OEM vorgegeben. Im **dynamischen Segment** erfolgt der Zugriff auf den FlexRay-Bus prioritätsgesteuert: Die ECU, welche mit der höchsten Priorität senden möchte, bekommt den Bus zuerst, sodass keine maximale Latenzzeit bei der Übertragung für niedriger priorisierte ECUs garantiert werden kann.

Jede ECU hat eine eigene Zeitbasis durch ihre lokale Uhr. Diese Uhren der vernetzten ECUs können aber voneinander abweichen: Erstens sind die Uhren nicht zueinander synchronisiert und können daher unterschiedliche Zeiten anzeigen. Zweitens driften die Uhren der ECUs wegen natürlicher Schwankungen im Quarz, sodass die Anzeigen von einst synchronisierten Uhren auseinanderlaufen.

Der FlexRay-Bus legt für die Kommunikation eine globale Zeitbasis fest, damit der Start eines Zeitslots auf allen FlexRay-Controllern zum gleichen Zeitpunkt stattfindet. Dies geschieht durch die Zeitsynchronisation im FlexRay-Controller. Jedoch sind die Uhren der ECUs in Relation zur globalen Zeitbasis des FlexRay-Busses nicht synchronisiert. Die Uhren driften in Verhältnis zueinander, wie in Abb. 1. angedeutet.

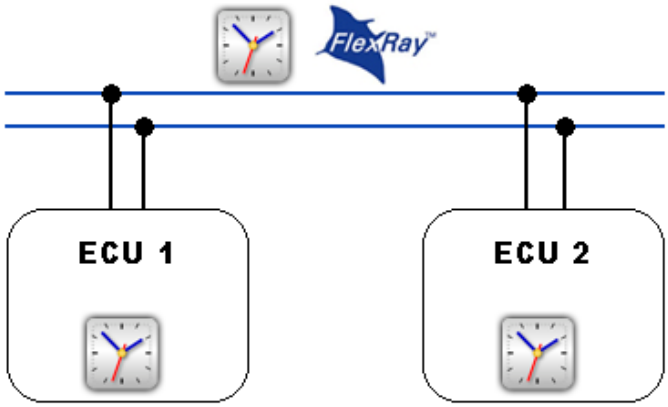

Abb. 1: Zeitbasen der ECUs und FlexRay

Ein häufiges Ziel bei der Übertragung von Nachrichten besteht darin, die Latenzzeit zwischen dem Bereitstellen der Daten auf der Sende-ECU und dem Verarbeiten der Daten auf der Empfangs-ECU zu minimieren. Hierzu ist es notwendig, die Daten rechzeitig – aber nicht zu früh – an den FlexRay-Controller zu übermitteln. Der Zeitpunkt zum Bereitstellen der Sendedaten wird von der Sende-ECU festgelegt. Es ist aber für die Sende-ECU unmöglich, nur anhand ihrer lokalen Uhr den Startzeitpunkt eines Zeitslots zum Senden einer Nachricht im statischen Segment zu bestimmen, denn die Zeit auf der lokalen Uhr der Sende-ECU weicht von den anderen im System vorhandenen Zeitbasen ab.

In der Realität haben die Übertragung in Zeitslots sowie die unsynchronisierten Uhren und die driftenden Zeitbasen sowohl der ECUs und auch des FlexRay-Busses mehrere Auswirkungen:

- *Erhöhung der Sendelatenz durch Warten auf den nächsten Zeitslot*

  Sendedaten werden in Event-Triggered-Tasks in Abhängigkeit von asynchron auftretenden externen Ereignissen berechnet. Diese Sendedaten sollen sofort übertragen werden. Hierzu muss jedoch der nächste der ECU zugeordnete Zeitslot abgewartet werden. Die Folge: Die Sendelatenz erhöht sich.

- *Übertragungslatenz durch unsynchronisierte Uhren*

  Oftmals bestehen Abhängigkeiten der Daten zwischen verschiedenen Tasks. Daten werden in Event-Triggered-Tasks vorverarbeitet und anschließend in Time-Triggered-Tasks berechnet, bevor sie über den FlexRay-Bus übertragen werden. Damit die Übertragungslatenz minimiert wird, sollten diese Time-Triggered-Tasks mit der gleichen Periodendauer wie die Zeitslots des FlexRay-Busses aktiviert werden. Zusätzlich sollte die Aktivierung so spät erfolgen, dass die

Bereitstellung der Daten gerade noch rechtzeitig vor dem Sendeslot auf dem FlexRay-Bus erfolgt. Sind die Uhren der Sende-ECU (ECU 1) und des FlexRay nicht synchronisiert, kann der in Abb. 2 dargestellte Effekt auftreten.

Die Time-Triggered-Task zum Berechnen der Sendedaten wird auf der ECU 1 periodisch alle 4 ms berechnet. Diese 4 ms ensprechen dem Zyklus des FlexRay-Busses. Da jedoch die Uhr der ECU 1 im Vergleich zur FlexRay-Uhr nachgeht, werden die Sendedaten an den FlexRay-Controller zu spät übermittelt, sodass der dem Steuergerät zugeordnete Zeitslot gerade verpasst wird. Dies führt zu einer Erhöhung der Kommunikationslatenz um die Zeitdauer bis zum nächsten Zeitslot.

Wie in Abb. 2 ebenfalls dargestellt, kann dieses Szenario noch eine weitere Folge haben: Bei der Empfänger-ECU (ECU 2) wird aufgrund der aufgrund der unsynchronisierten Uhren die periodische Weiterverarbeitung der Sendedaten in einer Time-Triggered-Task gerade verpasst. Die Latenz erhöht sich weiter. Im skizzierten Beispiel beträgt aufgrund der nicht synchronisierten Uhren zwischen dem Berechnen der Daten auf ECU 1 und dem Verarbeiten der Daten auf ECU 2 eine Latenz von 7 ms.

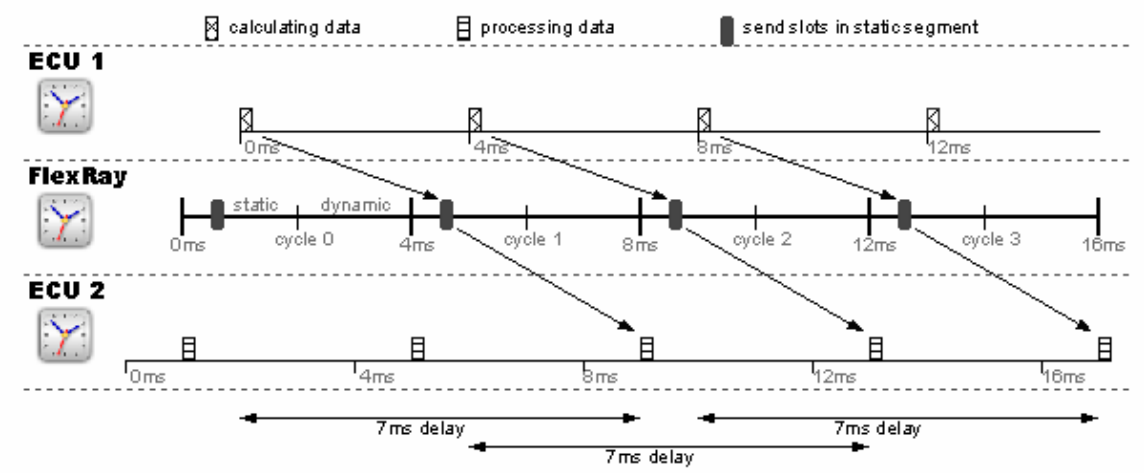

Abb. 2: Übertragungslatenz durch unsynchronisierte Uhren

- *Jitter der Übertragungslatenz durch driftende Uhren*

  Die auseinander driftenden Uhren führen zu einem dritten Problem, dem Jitter der Latenz bei der Übertragung: Eine Nachricht der ECU 1 wird gerade noch in dem ihr zugewiesenen Buszyklus übertragen, wie in Abbildung 3 dargestellt. Dann beträgt die Latenz bei der Übertragung in diesem Beispiel 0,2 ms. In folgenden Buszyklus erreicht die ECU-Nachricht den ihr zugewiesenen Zeitslot aufgrund einer im Vergleich zur FlexRay-Uhr kleineren Drift gerade nicht. Die Folge: Die Latenz bei der Übertragung auf der Sendeseite vergrößert sich erheblich. Der gleiche Effekt kann auf der Empfangs-Seite auftreten, wenn die Weiterverarbeitung der Sendedaten in einer Time-Triggered-Task erfolgt. Dies führt zu einer zusätzlichen Latenz auf der Empfangsseite, so dass die Gesamtlatenz 8 ms beträgt: Aufgrund des Jitters der Latenz werden Berechnungen auf der ECU 2 in der Time-Triggered-Task zu den Aktivierungszeitpunkten 5 ms und 9 ms mit alten Daten durchgeführt und erst zum Zeitpunkt 13 ms stehen neue Daten zur Verfügung.

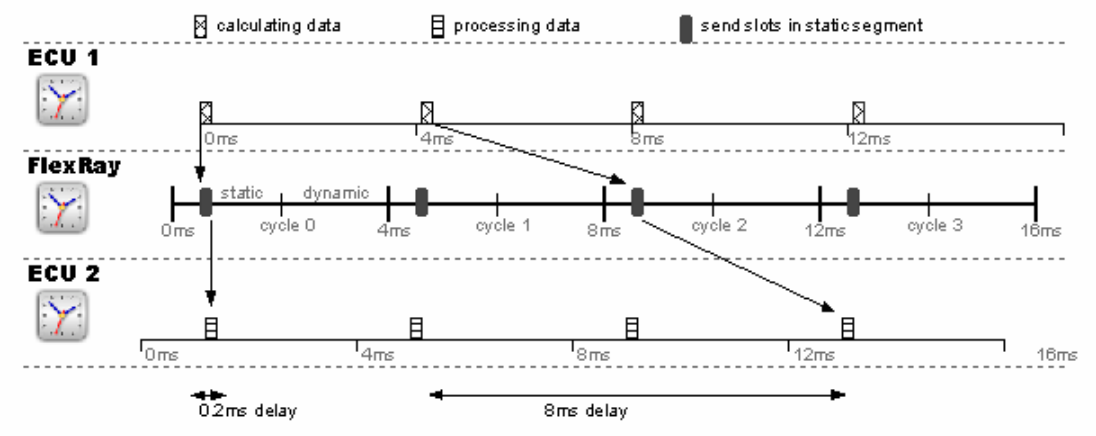

Abb. 3: Jitter der Übertragungslatenz durch driftende Uhren

- *Synchrone Berechnung von Daten auf mehreren ECUs trotz Jitter und nicht synchronisierter Uhren*

  Das vierte Problem entsteht, wenn zwei ECUs eine Berechnung synchron ausführen müssen. So kann der Fall auftreten, dass eine ECU den Impuls zu einer solchen Berechnung über den FlexRay-Bus an die zweite ECU überträgt. Um eine synchrone Ausführung zu ermöglichen, muss die Latenzzeit in etwa bekannt sein. Demzufolge ist der Jitter der Latenzzeit so klein wie möglich zu halten, da andernfalls eine synchrone Ausführung der Berechnung der ECUs kaum zu erreichen ist. Diese Funktionalität wird bei bestimmten Regelfunktionen im Automobil (z. B. Abgasreinigung mittels Lambdasondenmessung und Motorsteuerung) benötigt.

Die Optimierung von Latenz und Jitter stehen gegenseitig im Konflikt: Um das Latenzproblem zu optimieren, muss der Sendezeitpunkt der ECU-Nachricht nah an dem Startpunkt des FlexRay-Buszyklus liegen. Dann aber steigt die Gefahr, dass Nachrichten durch die Drift der unterschiedlichen Zeitbasen von ECU und FlexRay-Bus nicht weiterverarbeitet werden. Wenn das Jitterproblem optimiert werden soll, muss der Sendezeitpunkt möglichst früh vor dem Start des FlexRay-Zyklus liegen. Damit verschlechtert sich die Latenzzeit.

Eine Lösung der oben beschriebenen Problemstellungen bei der Sicherstellung der Echtzeitfähigkeit von vernetzten Steuergeräten könnte darin bestehen, die Time-Triggered-Tasks nicht in Abhängigkeit der ECU-Uhr zu aktivieren, sondern synchron zur FlexRay-Uhr. Hierbei sind folgende Aspekte zu beachten:

- Beim Systemstart verstreicht eine gewisse Zeit, bis die vollständige Funktionalität eines FlexRay-Controllers zur Verfügung steht. Dies betrifft insbesondere die Synchronisation der FlexRay-Controller zur Bestimmung der globalen Zeitbasis. Oftmals muss während dieser Startphase die ECU bereits Prozesse ausführen, so dass eine Aktivierung der Time-Triggered-Tasks synchron zum FlexRay-Bus in dieser Phase nicht möglich ist, sondern zunächst in Abhängigkeit der lokalen ECU-Uhr erfolgen muss.
- Ähnlich verhält es sich, wenn sich der FlexRay-Bus während des Betriebs neu synchronisiert. Steht der FlexRay-Bus zur Aktivierung der Time-Triggered-Tasks nicht mehr zur Verfügung, werden Alternativmechanismen benötigt.

In beiden Fällen führt ein hartes Umschalten bei der Aktivierung der Time-Triggered-Tasks zwischen der ECU-Uhr und FlexRay zu Zeitsprüngen, so dass die Tasks nicht mehr periodisch aktiviert werden. Ist dies aufgrund der zu regelnden physikalischen Prozesse nicht erlaubt, sind hierfür Alternativmechanismen notwendig, die einen weichen Übergang ermöglichen. Dies ist beispielsweise durch die Implementierung einer digitalen PLL (Phase-locked loop) möglich.

Da die in diesem Abschnitt dargestellten Effekte systembedingt auftreten, muss bei der Festlegung der Systemarchitektur der ECU demnach das durch die FlexRay-Kommunikation vorgegebene zeitliche Verhalten berücksichtigt werden, um optimale Ergebnisse zu erzielen. In der im Folgenden vorgestellten Motorsteuerung wurde systematisch mit Hilfe des Echtzeitsimulators chronSim eine Lösung erarbeitet, die eine Echtzeitfähigkeit der Gesamtsystems gewährleistet.

Die Anwendungsstudie

In der im Folgenden vorgestellten Studie wurde die Echtzeitproblematik zweier über FlexRay gekoppelter ECUs untersucht. Die beiden ECUs kommen in der Motorsteuerung zum Einsatz. Jedes der Steuergeräte regelt eine Zylinderbank. Ziel der Studie war die systematische Untersuchung des Datenaustausches der Steuergeräte über den FlexRay-Bus, um die Echtzeitfähigkeit des Systems zu gewährleisten.

Die ECUs regeln verschiedene Parameter für die Motorsteuerung nach fest definierten Kriterien. Die dafür notwendigen Tasks der ECUs können durch drei Arten der Aktivierung angestoßen werden: Durch zeitgesteuerte Aktivierung, durch asynchrone externe Ereignisse wie z. B. die Motordrehzahl und durch den FlexRay-Bus. Zeitgesteuerte Aktivierungen von Tasks folgen einem festen zeitlichen Raster, im vorliegenden Fall im Millisekundenbereich. Motorgesteuerte Aktivierungen erfolgen in Abhängigkeit eines speziellen Zahnrads auf der Kurbelwelle; jeder Zahn des Zahnrades löst einen Interrupt in der ECU aus. Die Aktivierungen durch den FlexRay-Bus werden im Takt der Zeiten des Buszyklus ausgelöst.

Einige Eingangssignale stehen beiden ECUs gleichermaßen zur Verfügung, andere nicht. In der Versuchsanordnung gab es eine Master-ECU, die mehr Aufgaben hatte als eine Slave-ECU. Die Slave-ECU erhält über den FlexRay-Bus vorverarbeitete Signale der Master-ECU. Die Nachbildung der Hardware-Architektur der Motorsteuerung im Echtzeitsimulator chronSim ist in Abb. 4 zu sehen: Sie beinhaltet u.a. die bidirektionale Kommunikation zwischen der Master- und der Slave-ECU über den FlexRay-Bus sowie Interrupteingänge, mit deren Hilfe die Sensoren ihre Signale an die Steuergeräte übergeben.

Wenn nun wichtige Signale vom Master zum Slave fließen, treten Latenzzeit- und Jitterprobleme bei der Übertragung auf. Im skizzierten Versuchsaufbau galt es daher, folgende Zielstellungen miteinander in Einklang zu bringen:

- *Optimierung der Latenzzeit*
- *Optimierung des Jitter*
- *Optimale Funktionsfähigkeit des Systems auch bei FlexRay-Resynchronisation*

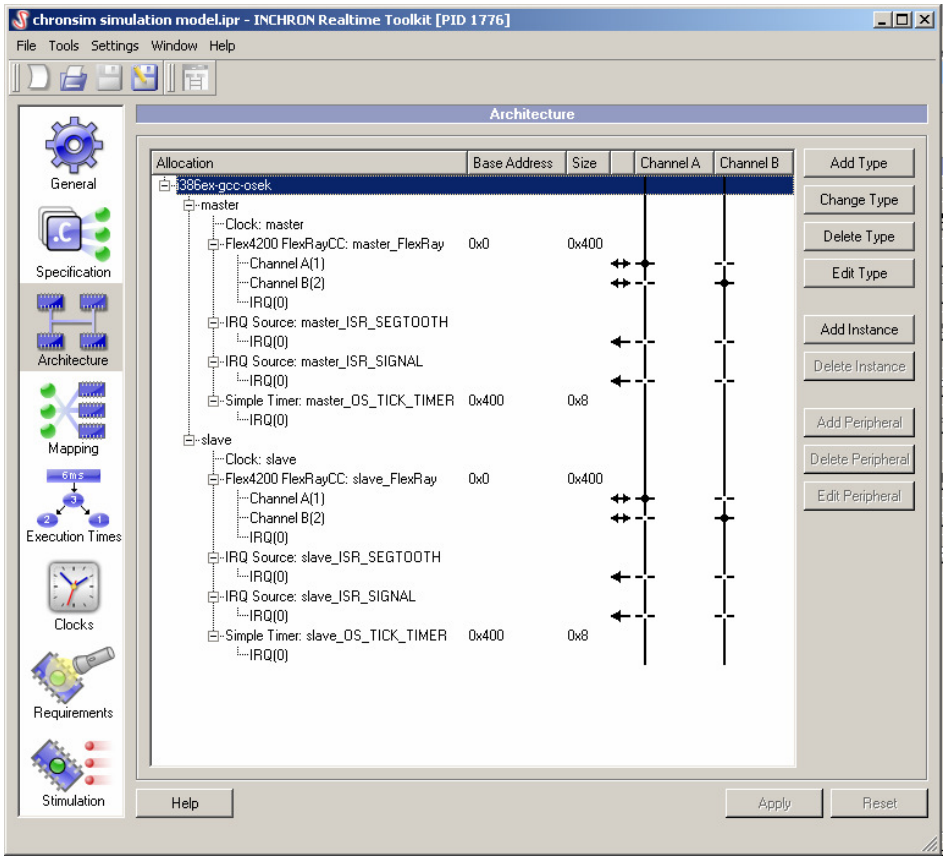

Abb. 4: Architektur der Motorsteuerung bestehend aus zwei ECUs und einen FlexRay-Bus

Wie bereits oben angedeutet, konkurrieren diese Anforderungen miteinander. Eine Optimierung der Latenzzeit resultiert meistens in einer Verschlechterung des Jitter – und umgekehrt. Durch die weiter oben skizzierten spezifischen systembedingten FlexRay-Probleme muss auch die Kommunikation der beiden über FlexRay verbundenen ECUs (Master-Slave, Slave-Master) bestimmten Zeitanforderungen genügen. Auch hier bedingt die Optimierung eines Parameters die Verschlechterung des anderen. Dies ist besonders dann unerwünscht, wenn die zeitgleiche Ausführung einer vom Master-ECU ausgelösten Aufgabe auf beiden ECUs gefordert ist: Dies ist nur dann möglich, wenn die Latenzzeit zum einen sehr genau bekannt und zum zweiten stabil ist.

Um die Teststellung auf die geforderten Parameter hin systematisch optimieren zu können, kam das Softwaretool chronSim zum Einsatz. chronSim ermöglicht die vollständige Echtzeitsimulation, Analyse und detaillierte Vorhersage des dynamischen Verhaltens von softwaregestützten Steuergeräten. Mittels virtueller Prototypen können mehrere Steuergeräte gleichzeitig simuliert werden. Auch die Simulation des Zusammenspiels der virtuellen ECU mit dem FlexRay-Bus ist möglich.

Erster Schritt der Analyse war der Nachbau der Teststellung in chronSim. Dazu wurde ein Task-Modell automatisiert aus der bereits vorhandenen Implementierung der beiden Steuergeräte erstellt. Task-Modelle bilden das Grundgerüst für die Spezifikation und somit für die Analyse des Systemverhaltens jeder ECU. Die Task-Modelle beinhalten eine Menge von nebenläufigen Betriebssystem-Tasks und Interrupt-Service-Routinen einschließlich des Schedulingverfahrens, der

Task-Prioritäten und der FlexRay-Kommunikation. Für den Einsatz von Task-Modellen sprechen mehrere Gründe:

- Task-Modelle lassen sich sehr einfach und schnell erstellen, da sie in der gleichen Sprache wie die Implementierung – in C – geschrieben werden. Das Erlernen einer zusätzlichen Spezifikationstechnik entfällt.
- Durch die Verwendung von C als Programmiersprache für die Erstellung von Task-Modellen ist ein Einbinden und Analysieren von bereits implementierten Modulen direkt möglich. Dies erlaubt Entwicklern einen fließenden Übergang von der Modellebene während des Systemdesigns in die anschließende Implementierung, da die gleiche Programmiersprache zum Einsatz kommt.
- Die Ausführungszeiten können als Zeit-Budgets entsprechend dem erforderlichen Abstraktionsniveau auf (Betriebssystem-)Task-, Funktions- oder Anweisungsebene vorgegeben werden. Dies erleichtert auch während der anschließenden Implementierung und dem Integrationstest ein Überprüfen der Echtzeitfähigkeit des Steuergerätes, da das geforderte Zeitverhalten in einer Zeitspezifikation bereits formalisiert wurde.

Im konkreten Fall wurden bereits bekannte Abläufe in den ECU in chronSim nachgestellt, sodass die Abläufe in den virtuellen Prototypen belastbare Rückschlüsse auf die Abläufe in den realen Prototypen zulassen. Zusätzlich wurden die bereits gemessenen Laufzeiten als Zeitbudgets in das Task-Modell eingebunden. Der FlexRay-Controller wurde in chronSim als eine Komponente des virtuellen Prototypen definiert. Im virtuellen Prototypen des FlexRay-Controllers sind die Informationen zur zeitlich richtigen Übertragung von Daten im statischen oder dynamischen Segment eines FlexRay-Zyklus, die Nachbildung der API und die Synchronisation der lokalen FlexRay-Controller-Uhren zur Gewährleistung einer globalen Zeitbasis im FlexRay-Netzwerk implementiert. Damit war der Nachbau der Teststellung abgeschlossen, die Bedingungen in dem virtuellen System innerhalb chronSim glichen denen in einem realen Versuchsaufbau.

Durch die Verwendung virtueller Prototypen konnten mittels chronSim verschiedene Szenarien analysiert werden, etwa die Größe des Jitters bei der Aktivierung von zeitgesteuerten Tasks, oder die Optimierung der Latenz von ECU-Nachrichten beim Versand über den FlexRay-Bus. Dabei war nicht nur die Untersuchung des Gesamtsystems mehrerer über FlexRay gekoppelter ECUs von Interesse, sondern auch die internen Abläufe innerhalb der Steuergeräte. Gerade die Möglichkeit der isolierten Betrachtung des dynamischen Verhaltens einzelner ECU im Wechsel mit der Betrachtung innerhalb des Gesamtsystems half nachdrücklich, die unterschiedlichen Ergebnisse auf deren Ursachen zurückzuführen: Resultiert eine festgestellte Latenzproblematik in fehlerhafter Programmierung des ECU, in systembedingten Verzögerungen des FlexRay-Busses oder in dem Zusammenspiel aller Komponenten? Welche Auswirkungen zeigt die Modifikation eines bestimmten Parameters auf das einzelne Gerät sowie auf das Gesamtsystem? Hierbei war es beispielsweise sehr hilfreich, im Task-Zustands-Diagramm (siehe Abb. 5) den zeitlichen Ablauf der Betriebssystemzustände der Tasks zu sehen und hierdurch die Auswirkungen von Task- und ISR-Verdrängungen zu analysieren.

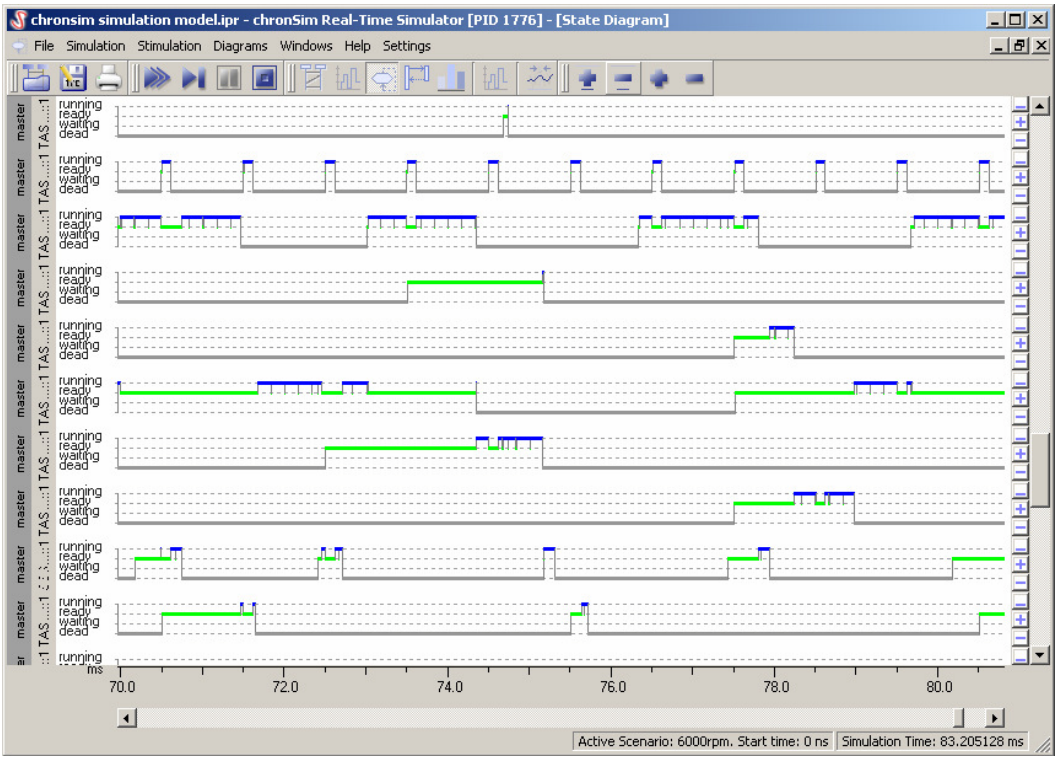

Abb. 5: Task-Zustands-Diagramm zur Visualisierung der Task- und ISR-Verdrängungen

Die Simulation der Teststellung in chronSim hatte gleich mehrere Vorteile: Zum einen war die Vorher-Nachher-Überprüfung von Änderungen sehr schnell möglich. Eine vergleichbare Analyse mittels Prototyping wäre weder zeit- noch kostenmäßig abbildbar gewesen. Zudem erlaubt es das virtuelle System, gezielt zeitkritische Stellen anzusteuern und zu untersuchen. Weiterhin erschloss die Analyse mit chronSim ein tieferes Verständnis, wie die einzelnen Komponenten im Versuchsaufbau dynamisch und auch im Gesamtsystem unter Einbeziehung des FlexRay-Busses reagieren.

Mit diesem Wissen konnten Komponenten und Gesamtsystem im Versuchsaufbau zielgenau optimiert werden. Das Echtzeitverhalten des untersuchten Systems verbesserte sich durch die gezielten Maßnahmen signifikant. Die Latenzzeiten innerhalb des Systems wurden in kontrolliertem Maße verbessert und dadurch auch das Jitterverhalten „besser“ im Sinne von „im jeweiligen Testumfeld noch gut kontrollierbar“. Insgesamt profitierte das Gesamtsystem von einem besser vorhersagbaren Verhalten bei den untersuchten Parametern.

Zusammenfassung

Zwei über den FlexRay-Bus verbundene Steuergeräte wurden hinsichtlich ihres Echtzeitverhaltens untersucht. Durch die Analyse des Systems mit dem Echtzeitsimulator chronSim konnten Echtzeitfehler des Gesamtsystems sowie jede einzelne ECU verstanden und daraus eine optimierte Lösung erarbeitet werden. Die Auswirkungen von Alternativen in der Softwarearchitektur der beiden ECUs im Zusammenspiel mit dem zeitlichen Verhalten des FlexRay-Busses konnten schnell und effizient analysiert werden. Durch den Einsatz von chronSim wurde eine Lösung zwei Monate früher als prognostiziert gefunden.

Literatur

[1] Freescale Silicon Enables World's First Vehicle with FlexRay(TM) Technology, News Release, (Freescale Semiconductor, Inc. 2006, http://media.freescale.com

[2] Software-Engineering for Powertrain-Control, Ingenieurbüro Barheine, Ettlingen © 2005, http://www.barheine.de/Powertrain.pdf

[3] A. Farrenkopf, P. Schneider, A. Ismailov, O. Bremicker: Brake-by-FlexRay. Hanser automotive, spezial edition FlexRay, 2006

[4] Zentrale Chassis-Dynamik-Steuerung und FlexRay Bus-System, http://www.siemensvdo.de

[5] www.inchron.de

[6] T. Komarek, R. Münzenberger: Detecting timing violations and tracking lost tasks in an RTOS. ECE Magazine, October 2006

[7] T. Komarek, M. Dörfel, R. Münzenberger: Developing real-time constrained embedded software using task models. In proceedings of the Advanced Automotiv Electronics (AAE 2007), Januar 2007

[8] FlexRay Communications System Protocol Specification Version 2.1, http://www.flexray.com