Simulation, Implementierung und Test vernetzter, zeitgesteuerter Fahrzeugsysteme

**Dipl.-Inform. Joachim Stroop, Dipl.-Ing. Susanne Köhl,
Dr.-Ing. Klaus Lamberg, Dr.-Ing. Rainer Otterbach
dSPACE GmbH**

Kurzfassung

Gerade die Entwicklung zeitgesteuerter Anwendungen bringt zusätzliche, über die heutige Situation hinausgehende Herausforderungen bezüglich der Entwicklungsmethoden und Entwicklungswerkzeuge mit sich. Insgesamt geht es darum, die heutigen Vorgehensweisen gemäß der Besonderheiten vernetzter, zeitgesteuerter und insbesondere sicherheitskritischer Anwendungen zu erweitern und zu systematisieren. Im vorliegenden Beitrag wird auf die existierende Unterstützung zur Entwicklung vernetzter Fahrzeugsysteme eingegangen. Exemplarisch wird dargestellt, wie diese Grundlage in Richtung auf zeitgesteuerte Fahrzeugsysteme weiterentwickelt werden kann.

1 Motivation

Wie viel Elektronik verträgt ein Automobil? Diese Frage mag sich angesichts des beeindruckenden Umfangs heutiger Steuergerätearchitekturen im Automobil stellen. Die zu beherrschende Komplexität derartiger Architekturen wird zunächst durch Anzahl und Umfang der realisierten Steuergerätefunktionen bestimmt. In zunehmendem Maße erweist sich ferner der Grad an Vernetzung als weiterer Komplexitätsparameter.

Informationsaustausch zwischen Steuergeräten und somit Reglersystemen erfolgt beispielsweise zur Weitergabe von Statusinformationen oder, um Sensorwerte gleichzeitig mehreren Empfängern zur Verfügung zu stellen. Er findet sich ebenso an den Übergängen (Gateways) zwischen verschiedenen Fahrzeugdomänen wie Antriebsstrang und Chassis (vgl. [3]). Für geplante steuergeräteübergreifende Regleranwendungen wie dem Global Chassis Control (vgl. [6]) ist dieser Informationsaustausch essentiell. Hier werden Reglerschleifen über die Grenzen von einzelnen Steuergeräten hinweg geschlossen.

Arbeitsgrundlage für vernetzte Steuer- und Regelungsanwendungen im Automobil ist der CAN-Bus. Für Entwicklung und Test solcher Anwendungen kann auf ein ausgereiftes Portefeuille von Entwicklungswerkzeugen zurückgegriffen werden. Allerdings werden mit dem wachsenden Grad an Vernetzung auch die Grenzen des CAN sichtbar. Das bereits heute vorhandene Kommunikationsaufkommen schöpft die tolerierbaren Bandbreiten des CAN oft aus. Das prioritätenbasierte, nichtdeterministische Arbitrierungsschema verschlechtert die Güte von globalen Regelkreisen, denn die Prioritätssteuerung führt je nach Auslastung des Busses zu zeitlichen Abhängigkeiten innerhalb eines Netzes. Und schließlich fehlen wirksame Fehlertoleranzmechanismen.

Als Alternative bzw. als Ergänzung werden zeitgesteuerte Bussysteme wie TTCAN, TTP und als de facto Industriestandard insbesondere FlexRay diskutiert. Obwohl oft ausschließlich im Zusammenhang mit sicherheitskritischen X-by-Wire-Anwendungen genannt, bieten zeitgesteuerte Bussysteme Potential für die gesamte Vernetzung im Automobil. Hohe Bandbreiten, deterministisches Kommu-

nikationsverhalten und Fehlertoleranzmechanismen prädestinieren beispielsweise FlexRay als zentrales Rückgrat zukünftiger Steuergerätearchitekturen. In Verbindung mit vereinheitlichten Services über diesen Bus und Gleichteilestrategien unter dem Aspekt der Wiederverwendung steht zu erwarten, dass damit langfristig auch Kostenvorteile darstellbar sind.

Die Eingangsfrage nach der beherrschbaren Komplexität muss schließlich vor dem Hintergrund der begleitenden Entwicklungswerkzeuge gesehen werden. Wachsende Vernetzung auf der Anwendungsseite wird zu verbesserter Unterstützung für vernetzte Bussysteme führen. Insbesondere zeitgesteuerte Systeme erfordern, dass Planungsaufgaben bereits in der Entwurfsphase vollständig durchgeführt werden.

In der Konsequenz dieser Entwicklung rückt das Gesamtsystem mehr denn je in den Fokus. Die einzelnen Steuergerätefunktionen werden zu Bausteinen einer übergreifenden Architektur – mit Konsequenzen für die Vorgaben, nach denen solche Bausteine erstellt werden.

2 FlexRay

FlexRay ist ein neues, zeitgesteuertes Kommunikationsprotokoll für sicherheitsrelevante Regelungen im Fahrzeug. Während ein ereignisgesteuerter Standardbus wie CAN bei niedriger Last zwar gute Übertragungsraten aufweist, kommt es bei vielen gleichzeitigen Busereignissen zu unkalkulierbaren Verzögerungen der Übertragungszeit. Bei sicherheitskritischen Echtzeitanwendungen jedoch müssen die Busereignisse stets zeitlich exakt vorhersagbar sein. Die vernetzten FlexRay-Controller erzeugen eine globale Zeitbasis, wodurch für fehlertolerante Anwendungen alle Signallaufzeiten klar definierbar und bestimmbar sind.

Für weitergehende Darstellungen zur FlexRay-Technologie sei auf [5] verwiesen. Im vorliegenden Beitrag wird insbesondere auf die nachfolgenden Eigenschaften Bezug genommen:

- FlexRay ist ein deterministisches Protokoll mit Datenraten bis zu 10 Mbit/s.
- Die FlexRay-Controller bestimmen mittels fehlertoleranter Uhrensynchronisation eine globale Zeit. Sie können ein oder zwei Kommunikationskanäle bedienen.
- Der Buszugriff erfolgt kollisionsfrei. Nachrichten auf einem FlexRay-Bus werden innerhalb eines periodischen Kommunikationszyklus in vordefinierten Zeitfenstern („Slots“) übertragen.
- Für die Anwendungen wird analog ein periodischer Anwendungszyklus definiert, der aus einem oder mehreren Kommunikationszyklen bestehen kann.

3 Systementwurf

Der Entwurf vernetzter Systeme erfolgt auf einer Abstraktionsebene oberhalb des Funktionsentwurfs. In einem zugrundeliegenden Systemmodell werden Eigenschaften der zu realisierenden Systemarchitektur erfasst. Das Systemmodell unterscheidet zwischen Eigenschaften der Funktionen und der Signale einerseits sowie der Rechnerhardware und des Kommunikationsmediums andererseits. Weiterhin beinhaltet das Modell Abbildungsvorschriften, z.B. Funktionen zu Rechnerknoten bzw. von Signalen zu Botschaften auf einem Kommunikationsmedium. Abbildung 1 zeigt in vereinfachender und nur symbolisch zu verstehender Darstellung ein solches Systemmodell mit beispielhaften Eigenschaften und zugehörigen Werten – hier mit einem XML-Editor erstellt.

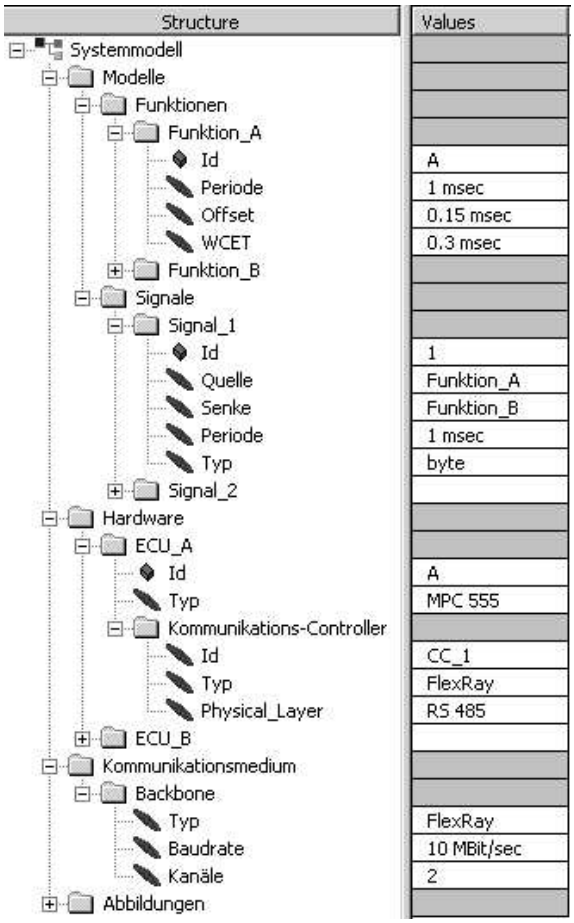

Abbildung 1: Beispielhafte Darstellung eines vereinfachten Systemmodells

Die im Systemmodell verankerte Trennung von Aspekten sollte sich sinnvoll auch in einem modellbasierten Systementwurf – basierend auf MATLAB/Simulink – widerspiegeln. Dazu sind Lösungsansätze im Sinne von Style-Guides empfehlenswert. Diese sollten beispielsweise die Verwendung expliziter „Signal Layer“ zur Signalauf- und -nachbereitung beinhalten (vgl. die gleichnamigen Simulink-Subsysteme in Abbildung 2). Signale werden somit nicht mehr direkt angeschlossen, sondern indirekt über die „Signal Layer“. Die eigentliche Reglerfunktion wird gekapselt und bleibt somit portabel, so dass sie zu einem geeigneten Zeitpunkt einer ECU zugeordnet werden kann.

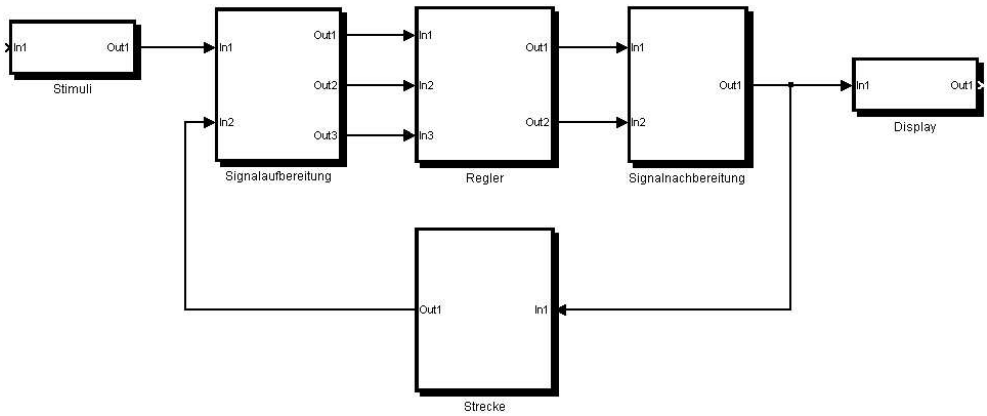

Abbildung 2: „Signal Layer“ im Systementwurf

Der Entwurf und die Darstellung der Kommunikationsmatrix stellen eine weitere, explizite Aufgabe im Systementwurf dar. Im Fall eines zeitgesteuerten Kommunikationsprotokolls wie FlexRay kommt zusätzlich die zeitliche Einplanung der Nachrichtenübertragungen hinzu. Für dieses a priori Scheduling in Abhängigkeit von den vorhandenen Kommunikationsbeziehungen und Ausführungsmustern der Regler stehen bereits heute entsprechende Werkzeuge zur Verfügung [11]. Bei einem vollständig auf MATLAB/Simulink begründeten Systementwurf kann anschließend eine Gesamtsystemsimulation durchgeführt werden, die zugleich mögliche Quantisierungseffekte bedingt durch die Buskommunikation berücksichtigt.

Das Systemmodell wird im arbeitsteiligen Zusammenspiel zwischen Fahrzeughersteller und Zulieferern erstellt. Hier lassen sich unterschiedliche Vorgehensweisen begründen. Der Fahrzeughersteller kann globale Vorgaben für alle beteiligten Zulieferer festlegen, z.B. in Form einer vollständig definierten Kommunikationsmatrix als Resultat einer Kommunikationsplanung. Er kann aber auch partielle Vorgaben machen, z.B. in Form von Korridoren für eine zeitgesteuerte Kommunikation, die vom Fahrzeughersteller global eingeplant werden. Diese werden vom Zulieferer im Rahmen des Rapid Control Prototyping (siehe Abschnitt 4) verfeinert und individuell gefüllt, so dass sie anschließend ins globale Systemmodell zurückgespeist werden können. Für den Informationsaustausch sind zwingend standardisierte Beschreibungsformate erforderlich. Neben den bekannten Formaten z.B. für die CAN-Kommunikationsmatrix werden übergreifende Ansätze wie FIBEX (Field Bus Exchange Format) [4] an Bedeutung gewinnen. Zugleich sind eigenständige Architekturbeschreibungssprachen in der Diskussion.

4 Rapid Control Prototyping

Rapid Control Prototyping (RCP) ist als Methode der Erprobung und Validierung neuer Reglerkonzepte im modellbasierten Entwicklungsprozess weithin etabliert. Der gesamte Entwurfsprozess vom frühen Reglerentwurf über erste Tests in Labor und Fahrzeug, späterem Prototyping auf targetnahen Plattformen sowie Flottentest bis hin zur Implementierung auf dem Seriensteuergerät erfordert eine durchgängige Hardware- und Software-Umgebung. dSPACE wird die Besonderheiten des Entwurfs vernetzter, zeitgesteuerter Systeme sukzessive auf allen Ebenen dieser RCP-Werkzeugkette berücksichtigen, so dass eine vollständige und durchgängige Integration beispielsweise des FlexRay-Protokolls gegeben ist. Im vorliegenden Abschnitt wird dazu ein erstes Integrationsbeispiel für ein klassisches High-End Prototyping-System beschrieben.

Aufgabe des Rapid Control Prototyping ist die frühzeitige Überprüfung eines Reglerkonzepts. Mit komfortablen Werkzeugen, die schnelle Experimentierzyklen unterstützen, wird ein Reglerentwurf im Kontext einer realen Strecke und unter Echtzeitbedingungen getestet. Damit werden Effekte einbezogen, die in einer reinen Offline-Simulation zunächst nicht berücksichtigt werden können. Die daraus gewonnenen Erkenntnisse führen zu einem abgesicherten Entwurf des Reglers bzw. der Architektur, in die der Regler eingebettet wird. Risiken in der nachfolgenden, kostspieligen Realisierung des Reglers werden minimiert.

Das Rapid Control Prototyping ist jedoch nicht auf die Überprüfung einzelner, isolierter Reglerfunktionen beschränkt. Es wird sich immer stärker auch auf die Überprüfung von vernetzten Funktionen, die in einem Steuergeräteverbund realisiert werden sollen, erstrecken. Abbildung 3 zeigt den prototypischen Aufbau einer elektromechanischen Bremse unter Verwendung von CAN als Netzwerktechnologie sowie von dSPACE AutoBox und MicroAutoBox als RCP-Systeme. Die Ablösung von CAN durch FlexRay könnte zu einem vergleichbaren Aufbau führen. Die redundant ausgeleg-

ten CAN-Busse würden durch die beiden Kommunikationskanäle eines FlexRay-Busses ersetzt werden. Auch steht zu erwarten, dass weitgehend auf die verwendeten Reglermodelle zurückgegriffen werden könnte. Zu berücksichtigen wären jedoch neue „Signal Layer“ einschließlich darin realisierter Fehlertoleranzverfahren sowie generell die durch die Zeitsteuerung erforderliche, veränderte Entwurfsmethodik, auf die im Folgenden näher eingegangen werden soll.

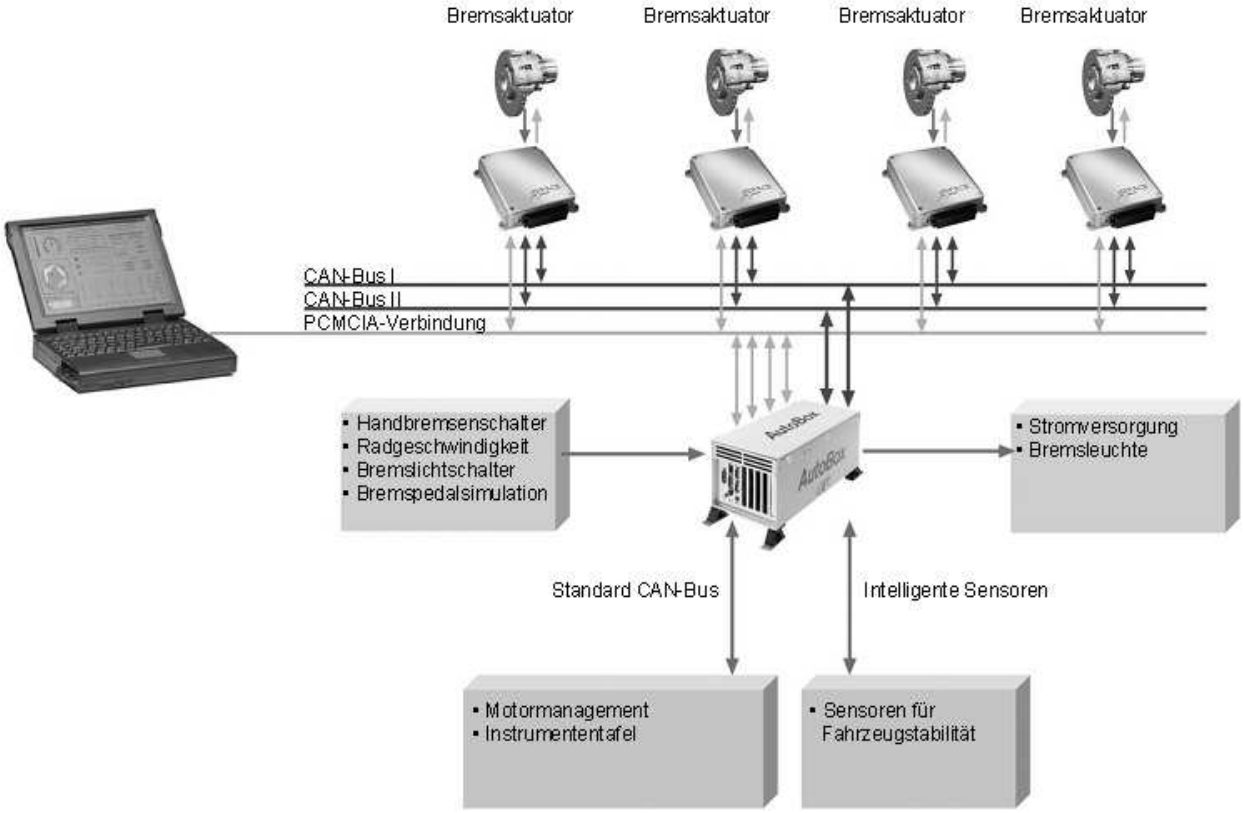

Abbildung 3: Brake-by-Wire Aufbau [2]

Bereits heute existieren leistungsfähige Prototyping-Systeme, die der Vielzahl von Bussystemen im Automobil Rechnung tragen. Sowohl die modularen Systeme als auch die MicroAutoBox (vgl. Abbildung 4) von dSPACE sind mit Schnittstellen für CAN, LIN und insbesondere FlexRay verfügbar. Die Systeme schreiben die bewährten Prototyping-Konzepte für zeitgesteuerte Bussysteme fort. Ihre Bauformen sind für den Einsatz im Fahrzeug vorbereitet.

Abbildung 4: Fahrzeugtaugliches RCP-System von dSPACE mit Schnittstellen zu CAN, LIN und FlexRay

Durch den zugrunde liegenden, zeitgesteuerten Ansatz von FlexRay sind während der Reglerentwurfsphase zusätzliche planerische Schritte im Vergleich zur bisher gewohnten Arbeitsweise notwendig. Hierzu sind u.a. die folgenden Aspekte zu beachten.

Task-Konfiguration: In der klassischen Regelungstechnik und den daraus abgeleiteten Ansätzen werden ein oder mehrere Subsysteme in MATLAB/Simulink zu einer Task zusammengefasst und auf einem RCP-System gemäß ihrer Abtastrate ausgeführt. Zur Aktivierung der Tasks zur Laufzeit werden Scheduling-Verfahren wie Rate Monotonic Scheduling herangezogen. Bei einer Zeitsteuerung erfolgt jedoch kein derartiges Scheduling zur Ausführungszeit. Die notwendigen Ablaufpläne werden schon im Entwurf bestimmt. Dazu müssen die Tasks mit weiteren zeitlichen Annotationen versehen werden. Eine FlexRay-Task ist zusätzlich durch den Startzeitpunkt innerhalb eines FlexRay-Anwendungszyklus sowie die maximale Ausführungszeit zu charakterisieren. Stand der Technik ist, dass diese Annotationen vom Anwender vorgegeben werden, die zugleich den Ablaufplan der Tasks festlegen.

Einplanung der Nachrichten: In einer auf dem CAN-Bus basierenden Fahrzeugkommunikation werden Nachrichten insofern eingeplant, dass ihnen implizit eine Priorität mit Hilfe des Message-Identifiers zugewiesen wird. Je höher die Priorität ist, um so wahrscheinlicher setzt sich die Nachricht am Bus durch. Die tatsächliche Nachrichtenübertragung erfolgt jedoch ereignisgesteuert und ist somit grundsätzlich nicht deterministisch. Für die Verwendung von FlexRay als Kommunikationssystem müssen die Nachrichten jedoch in Bezug auf den Ablaufplan der Anwendungs-Tasks zeitlich eingeplant werden. Jeder Nachricht wird somit exklusiv ein Übertragungs-Slot innerhalb des sich wiederholenden FlexRay-Kommunikationszyklus zugeordnet. Das Resultat ist ein statisches und deterministisches, offline berechnetes Schedule für Nachrichten und Tasks. Im Detail müssen zur Festlegung dieser Schedules weitere Aspekte berücksichtigt werden, z.B.:

- die Zusammenfassung von Signalen zu Nachrichten,
- die zeitliche Einplanung von Routinen für das Versenden und Empfangen von Nachrichten,
- verschiedene Redundanz-Strategien, wenn z.B. gefordert wird, dass Nachrichten identisch mehrfach innerhalb eines Kommunikationszyklus übertragen werden sollen.

Betriebssystem-Unterstützung: Die vorbestimmten Ablaufpläne für die Task-Ausführungen sowie die Nachrichtenübertragungen müssen zur Laufzeit durch eine geeignete Infrastruktur gekoppelt werden. Für die dazu erforderlichen Dienste ist mit OSEKtime [12] eine zeitgesteuerte Erweiterung der OSEK/VDX-Spezifikation festgelegt worden. Zusammen mit einer fehlertoleranten Kommunikationsschicht bietet eine Implementierung von OSEKtime eine verlässliche Ausführungsgrundlage für ein FlexRay-System. Für die Prototyping-Systeme von dSPACE ist der vorhandene Echtzeit-Kernel um erforderliche Bestandteile von OSEKtime erweitert worden. Eine volle Unterstützung von OSEK bzw. OSEKtime ist i.d.R. in den frühen Prototyping-Phasen noch nicht gefordert, womit sich die RCP-Systemkosten begrenzen lassen. Insgesamt werden die folgenden Dienste angeboten:

- Durch das FlexRay-Kommunikationssystem wird eine globale Zeitbasis vorgehalten. Abweichungen der lokalen Zeit eines angeschlossenen Hardware-Knotens vor dieser globalen Vorgabe müssen korrigiert werden – entweder „hart“ durch unmittelbare Korrektur oder „weich“ durch inkrementelle Anpassung über mehrere Schritte hinweg. Auf diese Weise wird der Startzeitpunkt des folgenden Anwendungszyklus bestimmt.
- Auf der Grundlage der lokalen Zeitbasis eines RCP-Systems muss der Echtzeit-Kernel die zeitlich korrekte Ausführung der Tasks gewährleisten – anstelle einer schnellen Verarbeitung von

Interrupts. Aktivitäten, die der Zeitsteuerung unterliegen, erhalten daher eine entsprechend hohe Priorität, so dass sie zu den vorbestimmten Zeitpunkten lauffähig werden. Die Aktivierung erfolgt relativ zum Startzeitpunkt des aktuellen Anwendungszyklus.

- Es ist ein Leerlaufverhalten möglich, d.h. eine Ausführung der Tasks ohne Synchronisation mit einem FlexRay-System. Damit können z.B. Laufzeiten der Tasks gemessen werden. Zugleich ist eine Re-Synchronisation mit dem Bus aus einem Leerlaufverhalten heraus möglich.
- Auch im Kontext eines zeitgesteuerten Bussystems wie FlexRay müssen die klassischen regelungstechnischen Funktionen in Verbindung mit der peripheren I/O sowie weiteren Bussystemen berücksichtigt werden. Der von dSPACE gewählte Ansatz, auf das etablierte Prototyping aufzusetzen und dieses mit einer Zeitsteuerung zu verknüpfen, ist prinzipiell geeignet, diese Aspekte modellbasiert zu berücksichtigen. Damit lässt sich auch ein Gateway-Knoten realisieren, welcher Nachrichten zwischen einem CAN- und einem FlexRay-Bus transferiert. Eine derartige Gateway-Funktion muss jedoch berücksichtigen, dass sowohl zeit- als auch ereignissynchrone Nachrichten verarbeitet werden müssen.

Die genannten Aspekte werden im Rahmen einer Werkzeugintegration realisiert, die von den Firmen DECOMSYS und dSPACE anteilig als Produkt angeboten wird. Abbildung 5 visualisiert die an der Integration beteiligten Werkzeuge und Komponenten, deren primären Aufgaben sowie den Datenfluss zwischen ihnen. Es ist geplant, diese Integration entsprechend der technologischen Entwicklungen im Umfeld von FlexRay anzupassen und fortzuschreiben.

Die in der Integration verwendeten Werkzeuge von DECOMSYS zielen vorrangig auf die Planung der Kommunikation sowie die Generierung des Codes für die fehlertolerante Kommunikationsschicht (FTCom). Dagegen werden mit Hilfe des RTI FlexRay Blockset von dSPACE automatisch der Anwendungs-Code generiert sowie alle Bestandteile zur Ausführung auf einem dSPACE Prototyping-System verknüpft. Die Ausführung und damit Simulation erfolgt mit Hilfe des dSPACE Echtzeit-Kernels gemäß der oben genannten Prinzipien von OSEKtime. Wie gewohnt können bestehende Produkte wie ControlDesk zur Durchführung und Visualisierung eines Experiments eingesetzt werden.

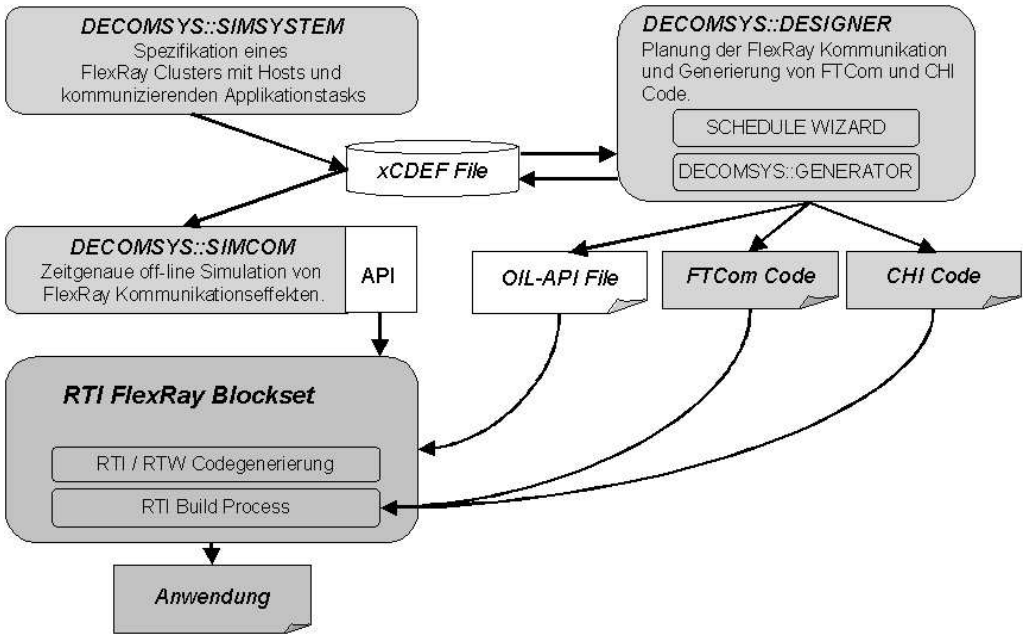

Abbildung 5: Werkzeugkopplung für das RTI FlexRay Blockset

Die nachfolgende Abbildung 6 veranschaulicht beispielhaft die Verwendung der Integrationslösung. Das Simulink-Fenster oben links zeigt die Modellierung einer Task einschließlich der Attribute Offset und Periode mit Hilfe von Blöcken aus dem SIMSYSTEM Blockset von DECOMSYS. Weiterhin enthält das Fenster einen Setup-Block für den genutzten FlexRay-Controller. Mit Hilfe der Eigenschaften dieses Blocks wird der Controller einem RCP-Knoten von dSPACE zugeordnet. Die Anwendungs-Task wird aus einem vereinfachten Subsystem gebildet. Dieses ist unten rechts abgebildet. Die Abbildung verdeutlich zugleich, dass die Integrationslösung konsequent auf MATLAB/-Simulink basiert.

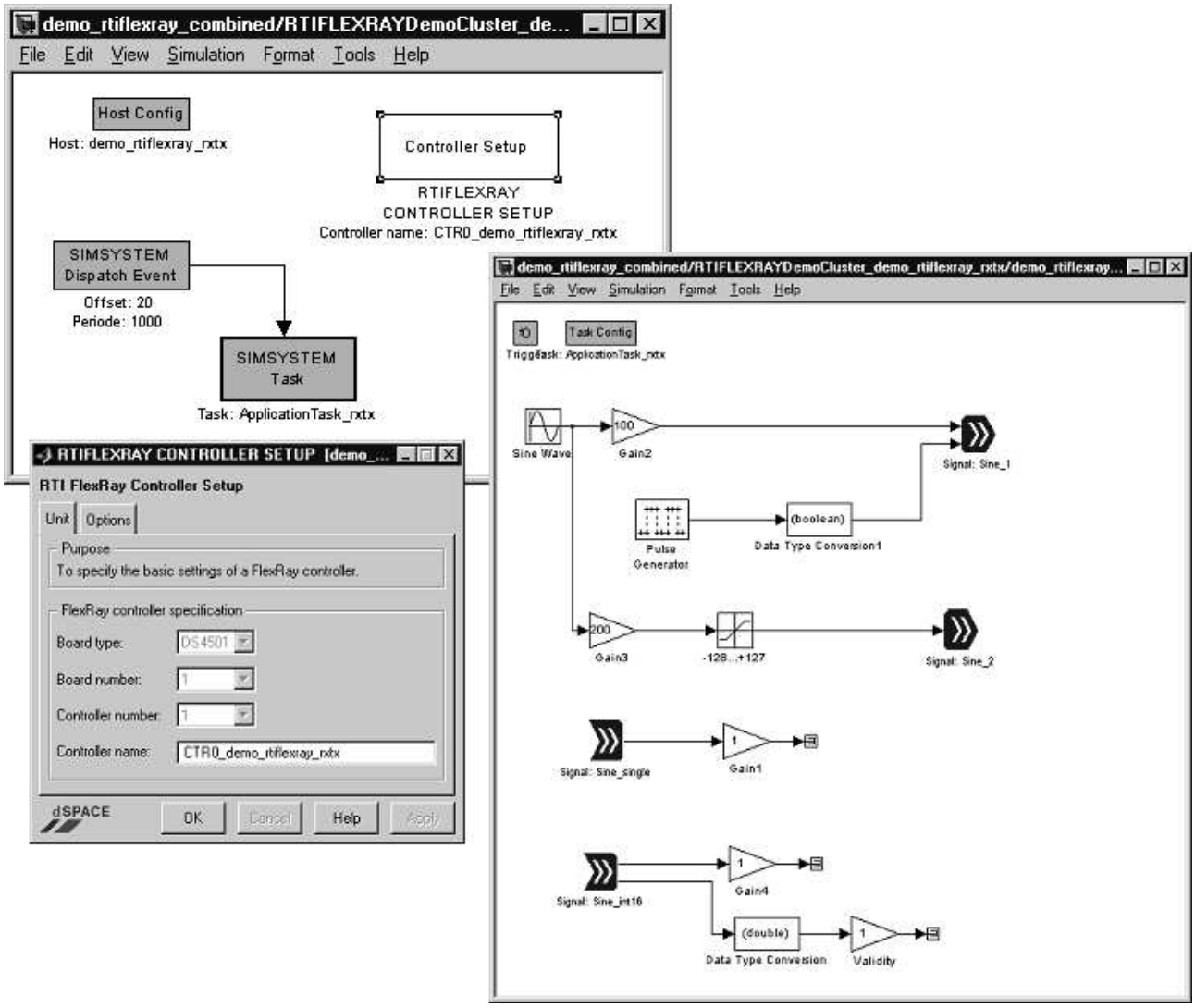

Abbildung 6: Beispielhafte Nutzung des RTI FlexRay Blocksets

5 Seriencode-Generierung und Software-Integration

Die Implementierung eines validierten Reglermodells im Seriensteuergerät mit Hilfe automatischer Code-Generierung ist ein wesentliches Element im modellbasierten Entwicklungsprozess. Hierbei sind die folgenden Aspekte zu berücksichtigen.

Die Generierung hoch-effizienten Seriencodes basiert auf einem adäquaten Modellierungsstil. Dazu gehört die Beschränkung auf solche Modellierungselemente (Simulink-Blöcke), die vom Code-Generator in effizienten Code umgesetzt werden können und mit den begrenzten Ressourcen des Seriensteuergeräts auskommen. Auch hier wird man typischerweise die Verwendung von Style-Guides empfehlen.

Zur Generierung des Seriencodes muss der Software-Entwickler auf Basis seines Implementierungswissens zusätzliche Informationen in das Modell einbringen. Dazu zählen beispielsweise die Skalierungseigenschaften der Signale und die Deklaration der Sichtbarkeit (Scopes) von Variablen. Insbesondere die Vorgabe der Skalierungseigenschaften ist ein aufwendiger und fehleranfälliger Prozess, der vom Code-Generator gut unterstützt werden muss. Der Seriencode-Generator TargetLink von dSPACE bietet deshalb Möglichkeiten für die automatische Skalierung und den Test des Festkomma-Codes in einer simulierten Umgebung [1]. Unter Verwendung eines seriennahen Prototyping-Systems oder des Seriensteuergeräts selbst ist sogar ein entsprechender Test in der realen Umgebung am Prüfstand oder im Fahrzeug möglich (vgl. auch Abschnitt 6).

Beim Übergang auf das Seriensteuergerät benötigt der Entwickler volle Kontrolle über die Partitionierung des Codes in Funktionen und Tasks. Dies wird durch entsprechende TargetLink-Blöcke erreicht, die in einem Simulink-Subsystem platziert werden. Über die Konfigurationsdialoge dieser Blöcke wird die Generierung von C-Funktionen und Betriebssystem-Tasks gesteuert. Die Unterstützung OSEK/VDX-konformer Betriebssysteme ist an dieser Stelle essentiell. Auch hier bietet die TargetLink Simulations- und Testumgebung dem Anwender umfangreiche Möglichkeiten, die Funktions- und Task-Partitionierung und die Parametrierung des Betriebssystems zu erproben und zu optimieren.

Von dieser Ausgangslage kann die Verwendung von TargetLink für ein vernetztes, zeitgesteuertes System auf Basis von FlexRay gestaltet werden. Eine konzeptionelle Trennung von Funktionsmodell (Regler) und Kommunikationsmodell (in einem „Signal Layer“) erweist sich hierbei als ausgesprochen hilfreich für eine separate Generierung der entsprechenden Code-Anteile. Zum einen beinhaltet dies die Umsetzung der einzelnen Funktionsmodelle in effizienten Seriencode für die jeweilige Zielplattform durch TargetLink. Zum anderen ist es erforderlich, den Kommunikations-Code (FTCom) für FlexRay aufgrund der vorgenommenen Ablaufplanung zu generieren – entsprechend Abbildung 5.

Zusätzlich gilt es dann, die Laufzeitumgebung für OSEK bzw. OSEKtime durch einen Systemgenerator auf Basis der in TargetLink spezifizierten Betriebssystemkonfiguration zu erzeugen. In der Regel liegen darüber hinaus weitere Software-Komponenten bereits im Quellcode vor oder werden aus anderen Werkzeugen generiert (z.B. Legacy Code, Diagnose- und Servicefunktionen, Startup-Code).

Für das abgestimmte Wechselspiel aller beteiligten Werkzeuge wird ein definiertes Austauschformat benötigt. Es steht zu erwarten, dass sich dieses mit FIBEX [4] etablieren kann. Die Integration aller Software-Komponenten im Steuergerät erfordert ferner eine Produktionsumgebung, die die Generierungswerkzeuge startet und die entstehenden Code-Anteile aus verschiedenen Quellen zusammenführt. Projekt-, Versions- und Konfigurationsmanagement sowie ein zentrales Data Dictionary sind essentielle Bestandteile der Produktionsumgebung.

6 Steuergerätetest

Der methodische Test ist ein wichtiges qualitätssicherndes Element im Steuergeräteentwicklungsprozess. Die modellbasierte Entwicklung ermöglicht und erfordert hier zugleich neue Herangehensweisen an den Test.

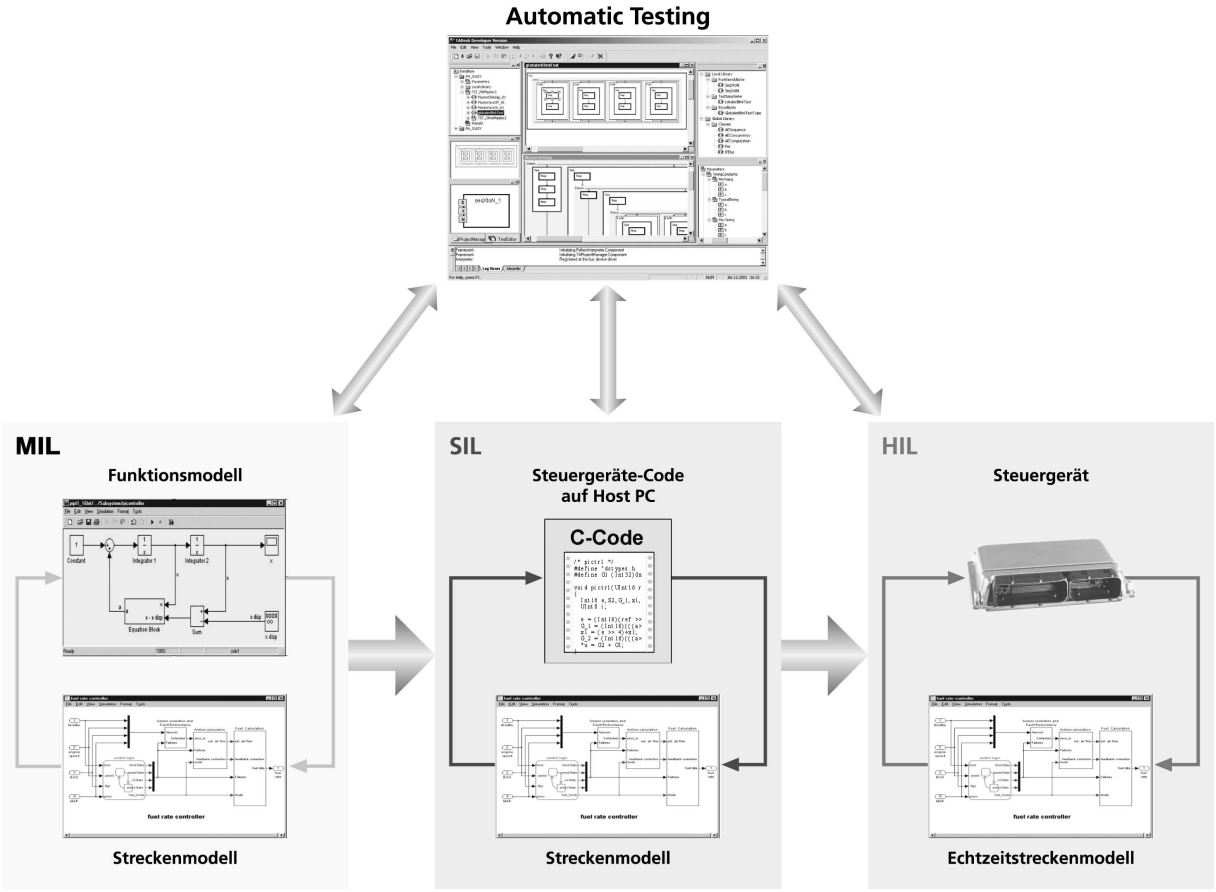

Abbildung 7: Automatische, modellbasierte Tests in verschiedenen Phasen des Steuergeräte-Entwicklungsprozesses

In frühen Entwicklungsphasen kann bereits eine Vielzahl von Tests ausgeführt werden, sobald eine Steuergerätefunktion als Simulationsmodell vorhanden ist (zum Beispiel in Simulink/Stateflow). Das Funktionsmodell, die „Unit-under-Test“ (UUT), und ein Streckenmodell als virtuelle Umgebung bilden einen geschlossenen Regelkreis, weswegen dieser Schritt auch Model-in-the-Loop (MIL) genannt wird.

Wie in Abschnitt 5 beschrieben wird in einem Software-in-the-Loop- (SIL-)System die UUT im Gegensatz zu MIL bereits in Form des Produktions-Codes simuliert, der später im Steuergerät integriert wird. Dabei kann das Verhalten des Steuergeräte-Codes mit dem des Funktionsmodells verglichen werden. Abweichungen ergeben sich z.B. durch Quantisierungseffekte bei der Verwendung von Fixed-Point-Code im Seriensteuergerät gegenüber der Floating-Point-Simulation des Funktionsmodells.

Den letzten Schritt bildet die Hardware-in-the-Loop-Simulation (HIL). Hier wird das reale Steuergerät in eine Simulationsumgebung eingebunden, die nicht nur das Streckenmodell, sondern auch elektrische Simulation beinhaltet. Eine realistische Simulation erfordert ein Echtzeitstreckenmodell.

Für einen effektiven Entwicklungsprozess ist die Wiederverwendung von Funktionstests aus früheren Phasen wichtig. Hinzu kommen bei der HIL-Simulation weitere Tests, wie zum Beispiel Diagnosetests oder der Test der Kommunikation in vernetzten Systemen.

Testen in frühen Phasen, Wiederverwendbarkeit von Tests sowie Testerstellung und -verwaltung und Automatisierung von Tests werden durch das neue integrierte Werkzeug dSPACE AutomationDesk unterstützt, das seit Mitte 2003 produktiv in Projekten in der Automobilindustrie eingesetzt wird. Es erlaubt eine schnelle und effiziente Testerstellung, stellt Basistestfunktionalitäten in einer Bibliothek zur Verfügung und bietet eine Projektmanager-Komponente für das Speichern und Verwalten großer Mengen von Tests, Testdaten und Testergebnissen [10].

HIL-Simulatoren von dSPACE werden nicht nur für den Test einzelner Steuergeräte, sondern insbesondere beim Automobilhersteller vielfach zum Test kompletter Steuergeräteverbunde eingesetzt [8]. Die Einsatzgebiete eines HIL-Simulators für den Test vernetzter Systeme sind vielfältig:

- Überprüfung des Zusammenspiels aller (bereits vorhandenen) Komponenten und ihrer Funktionen im Verbund. Da die Komponenten oftmals von unterschiedlichen Zulieferern bereitgestellt werden, kann dieser Test letztendlich erst beim Automobilhersteller erfolgen.
- Überprüfung von Teilfunktionen (ggf. erst nach der Identifizierung eines möglichen Problems zum Finden des Verursachers).
- Elektrische Fehlersimulation.
- Überwachung aller CAN-Signale, insb. auf ihr korrektes zeitliches Verhalten („30 msec nach Ereignis A muss das Steuergerät die Botschaft B erzeugen“).
- Simulation von im Verbund (noch) nicht vorhandenen Steuergeräten mittels Restbussimulation.
- Systematische Erzeugung von elektrischen und logischen Fehlern - einschließlich logischer Fehler im Netzwerk, z.B. im CAN-Bus.

Grundsätzlich kann sich der HIL-Simulator in den meisten Fällen nicht auf die Kommunikationsschnittstellen des Steuergeräts allein abstützen, sondern muss in der Lage sein, die gesamte analoge und digitale I/O des Steuergeräts zu bedienen.

Eine Besonderheit stellt sicherlich der Test des Verhaltens der Steuergeräte bei Fehlern in der CAN-Kommunikation dar. Wie sich das Steuergerät oder die verteilte Funktion verhält, wenn eine erwartete CAN-Botschaft ausbleibt oder unplausible Signale enthält, testet man im dSPACE Simulator mit Hilfe eines Fehler-Gateways [7]. Dabei werden einzelne oder mehrere CAN-Botschaften eines beliebigen Steuergeräts gezielt unterdrückt und/oder manipuliert, siehe Abbildung 8.

Hierfür werden zwei CAN-Controller pro Bus verwendet, mit denen jedes Steuergerät wahlweise und separat verbunden werden kann. Eine Softwarekomponente dient als Fehler-Gateway zwischen den beiden Controllern. Alle auf einem Controller empfangenen Botschaften werden sofort an den zweiten Controller gesendet, damit jedes Steuergerät die CAN-Botschaften aller anderen Steuergeräte empfängt. Fehlerfälle bis auf Botschafts- oder Einzelsignalebene herunter können erzeugt werden, indem die entsprechenden CAN-Nachrichten per Software verfälscht werden. Die resultierenden zeitlichen Verzögerungen wirken sich nicht auf die Funktionalität der Steuergeräte aus.

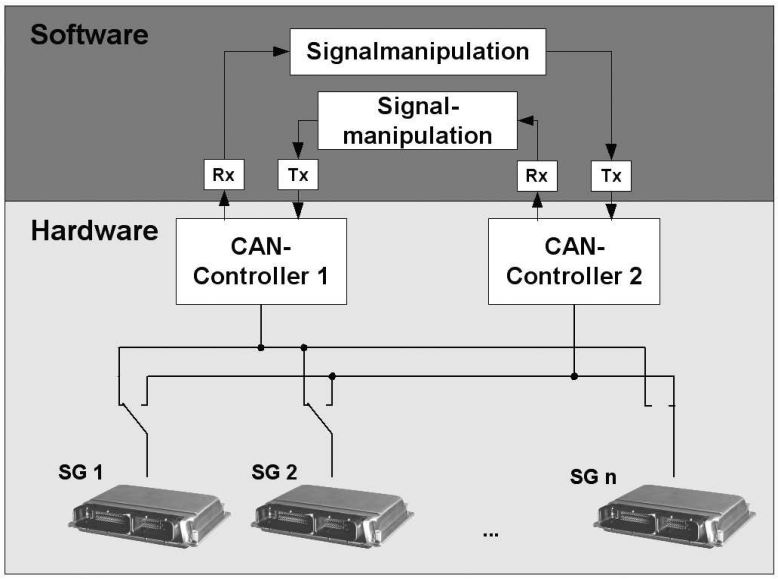

Abbildung 8: CAN-Konzept im dSPACE Simulator mit Fehler-Gateway

Im Hinblick auf den Test für vernetzte Steuergeräte mit FlexRay als Systembus ergeben sich prinzipiell ähnliche Konzepte wie zuvor genannt. So kann die in Abschnitt 4 beschriebene Werkzeugunterstützung einschließlich der genutzten Hardware grundsätzlich auch für den Aufbau von Simulatoren für FlexRay genutzt werden. Im Detail ergeben sich jedoch Besonderheiten, die im Folgenden charakterisiert werden sollen.

Restbussimulation: In der Restbussimulation werden Nachrichten von (noch) nicht real im Netzwerk vorhandenen Steuergeräten nachgebildet, so dass bereits vorliegende Steuergeräte getestet werden können. In einem zeitgesteuerten System soll für die Restbussimulation dieselbe zeitliche Einplanung von Nachrichten genutzt werden, die für das vollständig vernetzte System vorgesehen ist. Der Simulator muss dazu u.U. Reglerfunktionen rechnen, die auf den nachgebildeten Steuergeräten zeitgleich eingeplant werden, und somit dieselben zeitlichen Vorgaben in der eigenen Simulation berücksichtigen. Nur dadurch ist es möglich, die erzeugten Nachrichten in die entsprechenden Zeitfenster einzuordnen. Es ergeben sich somit verschärfte Anforderungen an die Echtzeitverarbeitung in der Restbussimulation.

Gateway: Ein Gateway-Konzept wie Abbildung 8 verwendet zwei CAN-Controller, um Signale auf einem Bus manipulieren zu können. Dieses für CAN taugliche Konzept kann voraussichtlich nicht unmittelbar auf FlexRay übertragen werden. Es würde eine Aufteilung eines FlexRay-Busses in zwei Segmente bedeuten und damit zwei Kommunikationssysteme mit eigenen globalen Zeiten schaffen. Zudem würde die Signalmanipulation im Gateway-Knoten Ausführungszeit erfordern, die die ursprüngliche globale Zeitplanung verwischt.

7 Fazit und Ausblick

Mit dem zunehmenden Interesse an zeitgesteuerten Konzepten für die Entwicklung von elektronischen Fahrzeugsystemen steigt der Bedarf an unterstützenden Werkzeugen. Derzeit etabliert sich FlexRay als das in solchen Anwendungen zum Einsatz kommende Busprotokoll. Der Schwerpunkt

der Entwicklungstätigkeiten bei FlexRay richtete sich bislang vorrangig auf die Definition und Absicherung des Protokolls sowie die Entwicklung entsprechender Kommunikations-Hardware. Auf der erreichten Grundlage können nun die eigentlichen Anwendungsentwicklungen stärker in den Fokus rücken.

In diesem Beitrag ist dargestellt worden, mit welchen Veränderungen bei den die KFZ-Elektronikentwicklung unterstützenden Werkzeugen zu rechnen ist. Dabei können die Erfahrungen in Bezug auf vernetzte Systeme aufgegriffen und fortgeschrieben werden. Die wichtigsten Änderungen sind die zusätzlichen planerischen Schritte in der Entwurfsphase sowie ein erweiterter Echtzeit-Kernel zur Einhaltung der resultierenden Ablaufpläne. Erst dadurch können das gewünschte deterministische Systemverhalten sowie Fehlertoleranzmechanismen gewährleistet werden.

Der Beitrag zeigt auch, dass bereits zum jetzigen Zeitpunkt mit dem Entwurf von Systemen, die auf FlexRay basieren, begonnen werden kann. Dieses ist auch darin begründet, dass die bewährten Werkzeuge in Bezug auf die neuen Herausforderungen bereits weiterentwickelt wurden und auch zukünftig weiterentwickelt werden können. Der erreichte Stand ist exemplarisch für die von diesen Werkzeugen unterstützen Entwicklungsphasen Rapid Control Prototyping, Seriencode-Generierung sowie Steuergerätetest dokumentiert worden.

Literatur

[1] Beine, M; Jungmann, M.: Code-Generierung für sicherheitskritische Systeme. Automotive Electronics, Sonderausgabe II/2003, S. 50-54.

[2] Braking at its Best. Kundenartikel in dSPACE NEWS, Nummer 2/2002, S. 4-5.

[3] Der neue BMW 5er. Sonderausgabe von ATZ und MTZ. August 2003.

[4] FIBEX – Field Bus Exchange Format. http://www.asam.de.

[5] FlexRay Konsortium. http://www.flexray.com.

[6] Frank, M.: Auf dem Weg zum intelligenten Fahrwerk. Auto & Elektronik, Ausgabe 1/ 2002, S. 60-64.

[7] Köhl, S.; Lemp, D.; Plöger, M.; Otterbach, R.: Steuergeräteverbundtest mittels Hardware-in-the-Loop-Simulation. Beitrag zur VDI-Tagung Mess- und Versuchstechnik in der Fahrzeugentwicklung, 2003.

[8] Gehring, J.; Schütte, H.: A Hardware-in-the-Loop Test Bench for the Validation od Complex ECU Networks. SAE World Congress 2003, Detroit, USA.

[9] Lamberg, K.: Testen mit System – Moderne Lösungen für die Hardware-in-the-Loop-Simulation. ATZ/MTZ Automotive Electronics, 2002, S. 42-47.

[10] Lamberg, K.; Richert, J.; Rasche, R.: A New Environment for Integrated Development and Management of ECU Tests. SAE World Congress 2003, Detroit, USA.

[11] Millinger, D.: Software-Werkzeuge für FlexRay und TTCAN. Netzwerke umfassend planen und konfigurieren. Sonderheft Elektronik Automotive, Ausgabe 2/2003, S. 31-35.

[12] OSEK/VDX Time-Triggered Operating System. http://www.osek-vdx.org/mirror/ttos10.pdf.