Fraunhofer
IIS

CPMNS TAGUNGSBAND

17. BIS 18. APRIL 2012, DRESDEN

9. ITG/GI/GMM WORKSHOP

CYBER-PHYSICAL SYSTEMS – ENABLING MULTI-NATURE SYSTEMS (CPMNS)

Domänenübergreifender Entwurf von heterogenen eingebetteten Systemen

GMM ITG

FRAUNHOFER VERLAG

ISBN 978-3-8396-0398-7

9. Workshop Cyber-Physical Systems – Enabling Multi-Nature Systems (CPMNS)

17. bis 18. April 2012

Dresden

Der Workshop wird 2012 mitorganisiert vom:

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS
Institutsteil Entwurfsautomatisierung EAS

CPMNS Workshop 2012
17./18.04.2012
Hotel Königshof
Dresden

Organisation:
Karsten Einwich
Fraunhofer-Institut für Integrierte Schaltungen IIS
Institutsteil Entwurfsautomatisierung EAS
Zeunerstraße 38
01069 Dresden
Tel: +49 351 4640-701
Fax: +49 351 4640-703
www.eas.iis.fraunhofer.de

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.
ISBN: 978-3-8396-0398-7

© by FRAUNHOFER VERLAG, 2012
Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 800469, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 0711 970-2500
Telefax 0711 970-2508
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

Die Autoren sind für den Inhalt der Beiträge dieses Tagungsbandes verantwortlich.
Layout: Fraunhofer IIS, Institutsteil EAS
Titelbil: MEV-Verlag

Alle Rechte vorbehalten
Dieses Werk ist einschließlich aller seiner Teile urheberrechtlich geschützt. Jede Verwertung, die über die engen Grenzen des Urheberrechtsgesetzes hinausgeht, ist ohne schriftliche Zustimmung des Herausgebers unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Speicherung in elektronischen Systemen.
Die Wiedergabe von Warenbezeichnungen und Handelsnamen in diesem Buch berechtigt nicht zu der Annahme, dass solche Bezeichnungen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und deshalb von jedermann benutzt werden dürften.
Soweit in diesem Werk direkt oder indirekt auf Gesetze, Vorschriften oder Richtlinien (z.B. DIN, VDI) Bezug genommen oder aus ihnen zitiert worden ist, kann keine Gewähr für Richtigkeit, Vollständigkeit oder Aktualität übernehmen.

Vorwort

Wir möchten alle Teilnehmerinnen und Teilnehmer ganz herzlich zum 9. Workshop Cyber-Physical Systems - Enabling Multi-Nature Systems (CPMNS) begrüßen. Der Workshop wird von der Fachgruppe 1 „Allgemeine Entwurfsmethodik und Unterstützung von Entwurfsprozessen für Schaltungen und Systeme“ der Kooperationsgemeinschaft RSS (Rechnergestützter Schaltungs- und Systementwurf) von GI, GMM und ITG mit Unterstützung des Fraunhofer IIS/EAS Dresden organisiert. Er führt die langjährige Tradition des Workshops Multi-Nature Systems fort.

Im Mittelpunkt steht der domänenübergreifende Entwurf von heterogenen eingebetteten Systemen und dabei insbesondere die Verknüpfung von mikro- und nanoelektronischen mit nichtelektronischen Komponenten. Die letztgenannten umfassen beispielsweise optische, mechanische, thermische, fluidische, akustische oder biologische Teilsysteme. Die Thematik umfasst sowohl die Sicht auf Cyber-Physical Systems als Ganzes (Spezifikation, Modellierung, Entwurf, Beispiele) als auch deren Systemkomponenten (Sensoren, Aktoren, spezielle elektronische Komponenten) und ihre Anwendung.

Unser besonderer Dank gilt dabei allen Autorinnen und Autoren sowie dem Programmkomitee, die es ermöglichten, ein vielfältiges wissenschaftliches Programm zu erstellen. Darüber hinaus möchten wir uns bei dem eingeladenen Redner Dr. Florian Voit von der Siemens AG bedanken. Der Workshop wird in diesem Jahr vom Fraunhofer IIS/EAS Dresden ausgerichtet. Hierbei gilt unser besonderer Dank Marina Ewert und Sandra Kundel. Für die Unterstützung bei der Durchführung des Workshops sei darüber hinaus dem Domero Hotel Königshof gedankt.

Wir hoffen, dass Sie während des Workshops zahlreiche Gelegenheiten haben werden, sich über aktuelle Themen auszutauschen sowie Kontakte zu den Teilnehmerinnen und Teilnehmern zu vertiefen. Im Namen der Ausrichter der Veranstaltung wünschen wir Ihnen einen schönen Aufenthalt in Dresden.

Dresden April 2012

Oliver Bringmann
Vorsitzender des Programmkomitees

Karsten Einwich
Organisationsleitung

Inhaltsverzeichnis

Eingeladener Vortrag

Interdisziplinäre Entwicklung komplexer eingebetteter Systeme im industriellen Umfeld

Florian Voit
Siemens AG
Industry Sector, Drive Technologies Division, Motion Control Systems
Erlangen, Deutschland
florian.voit@siemens.com

***Zusammenfassung*—Die Entwicklung eingebetteter Systeme ist ein interdisziplinärer, arbeitsteiliger Prozess, der große Anforderungen an die Modellierung stellt. Herausragende Bedeutung haben die Abstraktionsfähigkeit und die Einsetzbarkeit in heterogenen Entwicklungsumgebungen. Vor diesem Hintergrund werden Aspekte der Echtzeitsimulation von SystemC-Modellen beleuchtet.**

I. EINFÜHRUNG

Der Entwurf komplexer Systeme ist zumeist eine nicht triviale Aufgabe, für die es diverse Lösungsansätze und erprobte Methoden gibt. Viele dieser Ansätze sind jedoch auf eine spezifische Domäne ausgerichtet und spezialisiert, wie zum Beispiel das Software-Engineering oder die Konstruktionsverfahren in der Mechanik.

Schwieriger und leider oftmals auch unsystematischer sind die Vorgehensweisen beim Entwurf heterogener Systeme, in denen sich mehrere verschiedene technische Disziplinen wiederfinden und domänenübergreifende Ansätze notwendig werden. Zu den methodischen Schwierigkeiten kommen noch Missverständnisse und Inkompatibilitäten der unterschiedlichen Beschreibungsmittel und Notationen hinzu.

Aus Sicht eines industriellen Produkt- und Systemanbieters spielt neben dem Entwurf auch die Implementierung eine große Rolle. Die Entwicklung umfasst immer beide Tätigkeiten. Somit gewinnen die Differenzierung der Abstraktionsgrade und die Übernahme von Modellen zwischen den Abstraktionsebenen an Bedeutung.

II. HETEROGENE TECHNISCHE DOMÄNEN

Das Beispiel eines elektrischen Industrieantriebs in Abb. 1 zeigt, wie viele technische Domänen die Produktentwicklung beeinflussen: Applikationsspezifische Prozesse und regelungstechnische Algorithmen werden in Software implementiert. Echtzeitfähige Multitasking-Software mit Reglern, Kommunikations- und Betriebssystemfunktionalität läuft auf Mikrocontroller-basierter Hardware. Digitale Logik wird in ASICs oder programmierbaren Logikbausteinen abgebildet, denen ein zeitlich deterministisches Verhalten bis in den Nanosekundenbereich abverlangt wird. Leistungselektronik wird mit den unterschiedlichsten Schaltungstopologien und besonderen Anforderungen an die elektromagnetische Verträglichkeit und das thermische Verhalten entwickelt. Motoren sind elektromechanische Komponenten, die klassischerweise als Multi-Physik-Systeme betrachtet werden und mit der Finite-Elemente-Methode bezüglich mechanischer Festigkeit, elektromagnetischer Felder und thermischen Verhaltens untersucht werden. Schließlich sind noch applikationsspezifische Prozesse zu berücksichtigen, die mechanisch angetrieben und gesteuert werden, zum Beispiel Mehrkörpersysteme.

Viele weitere Domänen sind ebenso wichtig, werden

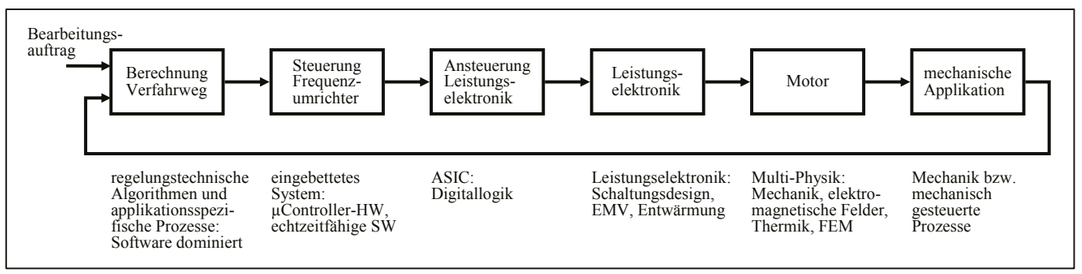

Abb. 1: Technische Domänen eines elektrischen Industrieantriebs

hier aber nicht weiter behandelt: Mensch-Maschine-Schnittstelle, Sensorik, Safety, Kommunikationstechniken, Inbetriebnahme-Unterstützung, Instandhaltung, Online-Services, automatisierte Diagnose und so weiter.

III. Anforderungen an die Modellierung

Bei der Vorbereitung der Modellierung richtet sich der Fokus meistens auf die Auswahl einer geeigneten Modellbeschreibungssprache beziehungsweise einer Kombination von Modellbeschreibungssprachen. Aus den adressierten technischen Domänen wird oftmals direkt auf das einzusetzende Werkzeug geschlossen: „Es sind Modelle der Domäne A und der Domäne B zu erstellen, also verwende ich das Tool X, das beide Domänen beherrscht.“ Dagegen macht sich der etwas erfahrenere Modellierer mit Ansprüchen an Genauigkeit und Zuverlässigkeit der Simulationsergebnisse zuerst Gedanken über die mathematischen Repräsentationsformen der domänenspezifischen Modelle. Er wählt beispielsweise ein Werkzeug aus, das sowohl einfache Differentialgleichungen für die regelungstechnischen Glieder als auch differentialalgebraische Gleichungen für die elektrischen Schaltungen lösen kann. Aus der Sicht eines industriellen Produktentwicklers sind jedoch beide Ansätze unvollständig.

Neben Kriterien wie analoge, digitale oder hybride Modellbildung sind weitere Aspekte zu berücksichtigen: Die Einbettung in den Entwicklungsprozess, die Wiederverwendung von Modellen in späteren Entwicklungsprojektphasen oder nachfolgenden Projekten, die Eignung für die Testfallerzeugung und Testdurchführung, interdisziplinäre Anwendbarkeit auch in unterschiedlichen Entwicklungsumgebungen und -plattformen sowie die Überleitung der Modelle in andere Abstraktionsebenen.

Auch die Verfügbarkeit hochwertiger kommerzieller Werkzeuge, die Unterstützung durch Supportprozesse wie Versions-, Konfigurations- und Änderungsmanagement sind wichtig. Nicht zuletzt spielt auch die Simulationseffizienz, das heißt der Rechenzeitbedarf und die beherrschbaren Modellgrößen, eine wichtige Rolle.

IV. Entwicklungsprozess

Die Entwicklung komplexer eingebetteter Systeme ist ein stark arbeitsteiliger Vorgang. Es entstehen zwangsläufig viele organisatorische, fachliche und technische Schnittstellen, was trotz unterschiedlicher Sichtweisen der Disziplinen und einer Vielfalt von Sprachen eine gute

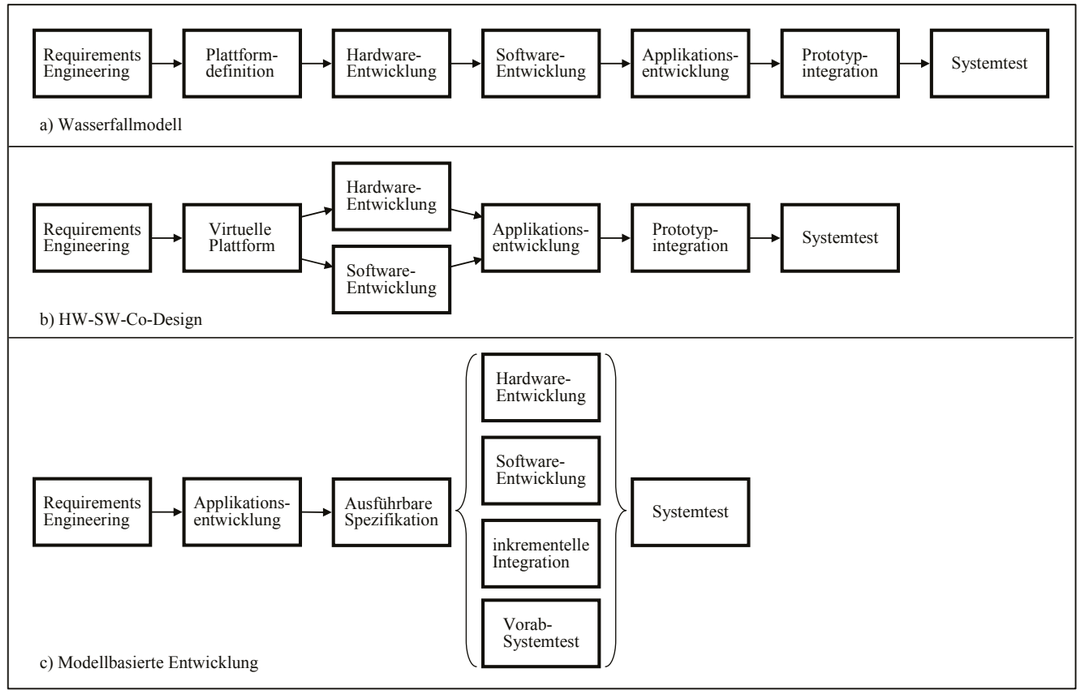

Abb. 2: Entwicklungsprozesse

und eindeutige Dokumentation der Arbeitsergebnisse und insbesondere der Spezifikationen verlangt.

Der traditionelle Ansatz eines operativen Entwicklungsprozesses (Abb. 2) ist das Wasserfallmodell. Beginnend mit dem Requirements Engineering reihen sich die Plattformdefinition, die Hardware-Entwicklung, die Software-Entwicklung und die Applikationsentwicklung nacheinander auf. Sie münden nach der Fertigstellung eines integrierten Prototypen und der Durchführung eines Systemtests schließlich in der Produktionsfreigabe.

Mit zunehmender Rechnerleistung und gestiegenen Anforderungen an die Entwicklungszeiten wurden Konzepte des Hardware-Software-Co-Designs eingeführt. Die Hardware wird dabei zuerst in Form eines Simulationsmodells spezifiziert und dann den Software-Entwicklern als Entwicklungsplattform zur Verfügung gestellt. Dieses „Virtuelle Plattform“ genannte Modell erlaubt einerseits den Software-Entwicklern einen viel tieferen Einblick in die Abläufe des eingebetteten Systems, als es ein physikalischer Prototyp mit seinen Echtzeitzwängen und eingeschränkten Debugging-Möglichkeiten je vermag. Andererseits nutzen die Hardware-Entwickler die virtuelle Plattform für Performance-Analysen und Optimierungen sowie als Vorlage für die konkrete Hardware-Implementierung.

Der nächste große Umbruch im Entwicklungsprozess ist die „Modellbasierte Entwicklung“, bei der die Applikationsentwicklung vom Ende des Prozesses an dessen Anfang verschoben wird. Die Applikationsentwicklung erzeugt ein funktionales Modell („Model of Computation“) des zu entwickelnden Systems und bildet es auf die Systemplattform ab („Mapping“). Dieses Modell wird weiter verfeinert bis zu einem Transaktionsmodell, welches die komplette Funktionalität und Dynamik des zu entwickelnden Systems gemäß einem vorgegebenen Abstraktionsgrad besitzt.

Im Electronic System Level Design (ESL) erzeugt man das Transaktionsmodell mit SystemC und nennt es ausführbare Spezifikation. Diese steht nun als virtuelle Plattform nicht nur dem bereits etablierten Hardware-Software-Co-Design zur Verfügung, sondern sie erlaubt

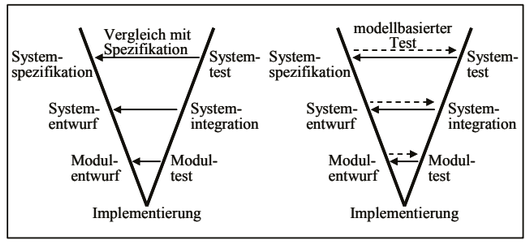

Abb. 3: Konventionelles (li.) und modellbasiertes (re.) V-Modell

eine weitergehende Parallelisierung der Entwicklungsaktivitäten, da sie allen technischen Domänen zur Verfügung steht und frühzeitige Tests des Systemaufbaus ermöglicht. Siehe [6].

Kennzeichen des in Abb. 3 gezeigten V-Modells ist die Gegenüberstellung der spezifizierenden Phasen und der testenden Phasen. Während der Testphasen im rechten V-Zweig wird auf die korrespondierende Spezifikationen des linken V-Zweiges zurückgegriffen. Der Vorteil der modellbasierten Entwicklung ist, dass nun unmittelbar nach jeder Spezifikationsphase im linken V-Zweig bereits die korrespondierenden Tests des rechten V-Zweigs durchgeführt werden können, noch bevor die im V-Modell jeweils weiter unten liegenden Schritte ausgeführt wurden. Für die dynamische Überprüfung der Systemspezifikation benötigt man also keine abgeschlossene Implementierung der einzelnen Module mehr. Etwaige Fehler oder Inkonsistenzen der Spezifikation lassen sich früher erkennen und ohne erneute Implementierung der detaillierten Module beheben.

In der Praxis ist das dort hilfreich, wo interdisziplinäre komplexe Systeme entwickelt werden und die Wahrscheinlichkeit von Missverständnissen zwischen den technischen Domänen besonders groß ist. Entwicklungsschleifen kosten Zeit und Aufwand.

V. Modellbeschreibungssprachen und Werkzeuge

Die praktische Anwendung von Modellbeschreibungssprachen ist eng an die Verfügbarkeit geeigneter Werkzeuge gebunden. Dadurch entstehen neben den oben genannten Anforderungen an die Modellierung weitere Ansprüche, deren Erfüllungsgrad die Arbeitsweise und die Prozessgestaltung beeinflussen.

Da ein Werkzeug kaum alle und insbesondere nicht die domänenübergreifenden Anforderungen erfüllen kann, ist ein Modellaustausch zwischen den verwendeten Werkzeugen notwendig. Das kann durch Änderungen der Repräsentationsform eines Modells und Einbettung in ein anderes Modell erfolgen, aber auch durch die verschiedenen Arten der Co-Simulation. Der erstgenannte Weg verändert das Modell, zum Beispiel durch Modellreduktion, andere mathematische Formulierungen oder modifizierte Simulationsschrittweiten und Gleichungslöser. Der zweite Weg hält zwar die Modelle unverändert, kostet aber einerseits Simulationsperformance durch Prozesswechsel und Inter-Tool- oder gar Netzwerkkommunikation. Und andererseits sind die Synchronisationsmechanismen von Co-Simulationen oft ungenügend, da die Schrittweitensteuerungen meistens nur eingeschränkt und fast nie bidirektional zwischen den Werkzeugen arbeiten oder gar feste Schrittweiten erzwingen.

Sollen in der modellbasierten Entwicklung Simulationsuntersuchungen unter Einbindung realer physikalischer, also nicht modellierter Komponenten erfolgen, dann ist man auf eine Echtzeitsimulation angewiesen. Typische Anwendungen sind das Rapid Prototyping, bei dem das zu entwickelnde Produkt simuliert und in eine reale Umwelt eingebettet wird, und der Produkttest, bei dem das zu testende reale Produkt in eine virtuelle Umwelt integriert wird.

Das Grundprinzip der Echtzeitsimulation ist es, erst schneller als in Echtzeit zu rechnen und dann den Rechner auf vorgegebene Synchronisationszeitpunkte warten zu lassen. Konzeptionell entstammen die echtzeitfähigen Werkzeuge überwiegend aus dem Gebiet der kontinuierlichen Modellbildung und arbeiten mit fester Schrittweite, um per deterministischen Rechenzeitbedarf die Einhaltung der Echtzeitsynchronität sicherzustellen.

Der Zwang zu fester Schrittweite und der Rechenzeitdeterminismus verlangen allerdings Kompromisse in der Modellierungsgenauigkeit. In kontinuierlichen Systemen ist es offensichtlich, dass an Arbeitspunkten mit wenig Dynamik unnötig detailliert und an Arbeitspunkten mit hoher Dynamik verhältnismäßig ungenau gerechnet wird. Für ereignisdiskrete Systeme entstehen Fehler bei der Bestimmung der Ereigniszeitpunkte, da die Ereignisse zeitlich auf die konstanten Schrittweiten gezwungen werden. Abhilfe oder zumindest Linderung dieses Problems versprechen Zeitstempelverfahren, die aber noch nicht weit verbreitet sind und wegen mangelnder Werkzeugunterstützung explizit und applikationsspezifisch in den Simulationsmodellen zu implementieren sind.

Die Sprache SystemC hat in Kombination mit den TLM-Mechanismen (Transaction Level Modelling) und der Erweiterung auf analoge Systeme per SystemC-AMS (SystemC Analog Mixed Signal) das Potenzial, viele Nachteile der Echtzeitsimulation zu überwinden.

VI. Einsatz von SystemC

SystemC hat seine Wurzeln im ASIC-Design, wird aber zunehmend für virtuelle Plattformen im Hardware-Software-Co-Design eingesetzt. Es findet, nicht zuletzt auch in Kombination mit SystemC-AMS, Verwendung als interdisziplinäre, ausführbare Spezifikation. Wegen des frei verfügbaren OSCI-Simulatorkerns [1] lässt sich SystemC auf praktisch alle C++-tauglichen Plattformen portieren. Und nach wenigen Eingriffen in die Implementierung des Simulatorkerns können SystemC-Modelle von anderen Simulatoren gesteuert und co-simuliert werden.

Damit erfüllt SystemC eine wesentliche Voraussetzung für den Einsatz in einem interdisziplinären, arbeitsteiligen Entwicklungsprozess. Jede beteiligte Domäne konfiguriert sich die in SystemC modellierte ausführbare Spezifikation nach ihren Anforderungen und führt dann die notwendigen Reviews und Tests der Spezifikation in ihrer eigenen Entwicklungsumgebung aus. Siehe [8].

Folgerichtig entsteht der Anspruch, auch SystemC-Modelle in Echtzeit zu simulieren. SystemC-Modelle sind de facto C++-Programme, die auf einem Rechnersystem mit einem Mikroprozessor ausgeführt werden. Mit gängiger PC-Technologie sind Synchronisations- und Reaktionszeiten im zwei- bis dreistelligen Mikrosekundenbereich machbar. Für schnellere Signale und kleinere Zeitraster sind weitere Maßnahmen notwendig, wie zum Beispiel die Auslagerung von zeitkritischen Berechnungen auf ein Field Programmable Gate Array (FPGA), das seinerseits vom SystemC-Modell gesteuert wird.

VII. SystemC in der Echtzeitsimulation

Bei der Echtzeitsimulation von SystemC-Modellen haben die Methoden zur Synchronisation mit der realen Zeit eine herausragende Bedeutung. Der einfachste Ansatz ist die äquidistante Abtastung, also die bereits erwähnten und bei rein kontinuierlichen Systemen üblichen festen Simulationsschrittweiten. Am Beispiel der Pulsweitenmodulation (PWM) in Abb. 4 erkennt man einige der resultierenden Simulationsartefakte. Die Signalflanken werden auf den Zeitpunkt des nächstfolgenden Abtastschritts verzögert. Das verfälscht nicht nur den Mittelwert des PWM-Signals, sondern auch sein Frequenzspektrum. Steuert das PWM-Signal einen integrierenden oder Mittelwert bildenden Prozess an, dann lässt sich der Mittelwertfehler bei genauer Kenntnis der Flankenzeitpunkte und der Schrittzeitpunkte durch eine Amplitudenanpassung korrigieren.

Das Frequenzspektrum ist und bleibt jedoch verzerrt und erschwert die simulative Überprüfung von Filterauslegungen oder Untersuchungen von Schwingungseffekten, Resonanzen, EMV-Verhalten und vielem mehr.

Verschärft wird dieses Problem beim Zusammenspiel mehrerer PWM-Signale, wie sie bei Frequenzumrichtern in der elektrischen Antriebstechnik vorkommen. Die resultierenden Raumzeiger stimmen weder in ihrer Amplitude noch in ihrer Richtung. Die derart erzeugten Simulationsergebnisse eignen sich nur eingeschränkt für die Analyse des regelungstechnischen Verhaltens eines komplexen Systems.

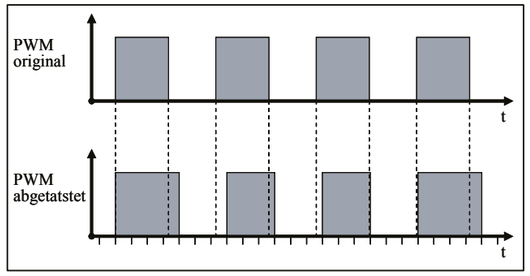

Abb. 4: Pulsweitenmodulation mit zeitlicher Abtastung

Es gibt derzeit keine allgemeinen und offensichtlichen Kriterien zur Festlegung der Synchronisationspunkte bei der Echtzeitsimulation ereignisdiskreter Systeme. Die in der Simulation kontinuierlicher Systeme angewendeten Kriterien wie das Abtasttheorem versagen hier. Einflussfaktoren sind Anzahl, Richtung, Häufigkeit und Vorhersehbarkeit von Ein- und Ausgangssignalen (IO), aber auch die modellierte Dynamik des Simulationssystems. Systeme können langsame oder schnelle Komponenten enthalten, mit jeweils wenig oder viel Rechenaufwand je Ereignis. So wie die sogenannten steifen Systeme in der kontinuierlichen Modellierung bereitet auch in der ereignisdiskreten Modellierung die Kombination stark unterschiedlicher Dynamiken die größten Schwierigkeiten. Die aus der Echtzeit-Software-Entwicklung bekannten Mechanismen, wie die Instanziierung parallel arbeitender Agenten und die Interrupt-Behandlung, sieht der SystemC-Sprachstandard in seiner derzeitigen Form nicht vor.

Resultierende Fragestellungen zielen auf die Einstellbarkeit oder zumindest die Vorhersagbarkeit von Latenzzeiten, Jitter und Event-Overrun-Situationen bei der Behandlung von IO-Signalen ab. Die Qualität der IO-Signale bestimmt die Brauchbarkeit der Echtzeitsimulation, die sich ausschließlich durch ihre Interaktion mit physikalischen Systemen via IO rechtfertigt.

In echtzeitfähigen Laboraufbauten wurde eine Kombination von SystemC- und Simulink-Modellen erfolgreich eingesetzt. Die zeitunkritischen SystemC-Teile und die Simulink-Komponenten werden auf einer klassischen CPU ausgeführt. Die zeitkritischen Steuerungsanteile übernimmt ein FPGA. Dabei kommt der Schnittstelle zwischen CPU und FPGA eine besondere Bedeutung zu. Die CPU berechnet die zeitkritischen Signale im Voraus und übergibt dem FPGA die auszugebenden Signalwerte gepaart mit Zeitstempeln. Das FPGA sorgt dann für die zeitkorrekte Ausgabe dieser Signale.

Auf dem umgekehrten Weg nimmt das FPGA Eingabesignale auf, vorverarbeitet sie bei Bedarf und leitet sie ebenfalls mit Zeitstempeln an die CPU weiter. Nicht die Verwendung eines FPGA ist hier zwingend, sondern die Zeitstempelung durch eine intelligente IO-Baugruppe. Das SystemC-Modell auf der CPU nimmt die Signale entgegen, ist aber wegen der oben beschriebenen Ausgabesystematik dem Eingangssignalstempel zeitlich immer ein Stück voraus.

Mit Kenntnis des Zeitstempels und der aktuellen SystemC-Simulationszeit kann nun eine applikationsspezifische Korrektur modelliert werden. Schwierig wird dieser Ansatz bei Systemen, in denen Komponenten von hoher Ereignishäufigkeit und geringem Rechenzeitbedarf pro Ereignis mit Komponenten von niedriger Ereignishäufigkeit und hohem Rechenzeitbedarf pro Ereignis zusammentreffen. Siehe [2].

Neben der IO-Behandlung haben auch die in SystemC-Modellen verwendeten Abstraktionsmethoden einen großen Einfluss auf das Echtzeitverhalten der Simulation. Um die Effizienz von Offline-Simulationen, also nicht echtzeitfähigen Simulationen, zu erhöhen, wird bei Konzepten wie TLM und Temporal Time Decoupling mit Zeitstempeln gearbeitet und die Rechenzeit auf wenige Simulationszeitpunkte konzentriert. Zu diesen Zeitpunkten ist der Rechenzeitbedarf sehr hoch bei gleichzeitig geringem Simulationszeitfortschritt, während er dazwischen sehr gering bei großem Simulationszeitfortschritt ausfällt.

Die Konzentration des Rechenzeitbedarfs auf wenige Zeitpunkte steht im Widerspruch zum klassischen Echtzeitansatz mit äquidistanten Abtastzeiten, die bei kontinuierlichen Systemen gleichmäßig verteilte Rechenlasten erzeugen und somit dort günstiges Verhalten aufweisen. Die kleinstmögliche äquidistante Abtastzeit muss größer oder gleich dem maximal pro Simulationsschritt anfallenden Rechenzeitbedarf sein. Siehe [7].

Die Rechenzeitproblematik verschärft sich noch weiter, wenn man Modellierungskonzepte mit konfigurierbaren Abstraktionsgraden einsetzt. Diese sind hilfreich, um die Simulationsmodelle auf die zu untersuchende Problemstellung anzupassen und kleine Rechenzeiten für die Abarbeitung unnötiger Modelldetails zu vermeiden. Es profitieren sowohl Echtzeit- als auch Nicht-Echtzeitsimulationen. Allerdings bedürfen die Echtzeitsimulationen zusätzlicher, konfigurationsabhängiger Rechenzeitabschätzungen und -messungen.

VIII. Ausblick

Es wird ein interdisziplinärer Entwicklungsprozess mit einem hohen Grad an Parallelität angestrebt. Das sollte einerseits gefördert werden durch werkzeugunterstütztes, Konsistenz wahrendes Wechseln der Abstraktionsgrade. Andererseits benötigt man modellbasierte Testmethoden, die die Testfälle Abstraktionsebenen übergreifend ableiten oder übernehmen und zusätzlich die Testergebnisse aus verschiedenen Abstraktionsebenen miteinander vergleichen können.

Um die Analogsimulation zu beschleunigen und mit den digitalen Komponenten im Einklang zu halten, wäre es wünschenswert, die aus SystemC bekannten Abstrak-

tionskonzepte wie TLM auf SystemC-AMS zu übertragen. Auch Methoden zur automatischen Generierung von SystemC-Modellschnittstellen aus abstrakteren Beschreibungsmitteln wie UML (Unified Modeling Language) oder SysML (Systems Modeling Language) würden den Entwicklungsprozess verbessern.

Nicht zuletzt lassen die Synchronisationskonzepte bei der Echtzeitsimulation Raum für Verbesserung. Intelligente, auch vom Modellierer beeinflussbare Schrittweitensteuerungen heben die Signalqualität deutlich. Bei dem oben genannten PWM-Beispiel würde das explizite Festlegen der Simulationsschritte auf die Signalflanken korrekte Signale erzeugen, ohne den Rechenzeitbedarf zu erhöhen oder dessen Determinismus zu verlieren.

[1] Accellera Systems Initiative: www.accellera.org

[2] Görgen, Voit, Rettberg: Simulation of SystemC Models in a Physical Environment. DATE 2012, Dresden 2012

[3] Nakamura, Hosokawa, Kuroda, Yoshikawa, Yoshimura: A Fast Hardware/Software Co-Verification Method for System-On-a-Chip by Using a C/C++ Simulator and FPGA Emulator with Shared Register Communication. DAC 2004, June 7–11, 2004, San Diego, California, USA.

[4] Trams, M: Realtimify – A Small Tool for Real Time SystemC Simulations. Digital Force White Paper, 2005. http://www.digital-force.net

[5] Ramaswamy, Tessier: The Integration of SystemC and Hardware-assisted Verification. In: Glesner, Zipf, Renovell (editors): Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream. Lecture Notes in Computer Science, Ausgabe 2438/2002, S. 513-539. Springer, Berlin / Heidelberg, 2002

[6] Voit, F.: Modellbasiertes Hardware-Software-Codesign für eingebettete Mixed-Signal-Systeme. In: Bender, Schumacher, Verl (Hrsg.), SPS IPC DRIVES 2010, VDE-Verlag, 2010

[7] Voit, F.: Echtzeitsimulation von Electronic-System-Level-Modellen. In: Maurer (Hrsg.), ASIM 2011, 21. Symposium Simulationstechnik. Winterthur, Schweiz, 2011

[8] Voit, F.: Konfigurierbare Systemmodelle im Model-based Embedded-System-Design. In: Bender, Schumacher, Verl (Hrsg.), SPS IPC DRIVES 2011, VDE-Verlag, 2011

SystemC / SystemC AMS basierte Modellierungs-, Simulations- und Verifikationstechniken

Design Understanding by Feature Localization on ESL*

Marc Michael[1] Daniel Große[1] Rolf Drechsler[1,2]

[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{mmichael,grosse,drechsle}@informatik.uni-bremen.de

***Abstract*—The increasing variety of functionality in embedded systems leads to more and more complex designs. Even abstraction techniques as applied in ESL design solve this problem only to a certain extend. In this paper we present an approach to improve design understanding of ESL models. Our approach localizes features of ESL models, i.e. source code implementing a certain feature is highlighted. This significantly helps the different team members since for major design tasks like e.g. refinement, partitioning or optimization it is required to know where a certain functionality has been implemented We demonstrate the advantages of our approach for a complex image processing system.**

I. Introduction

Each new generation of embedded systems adds a huge amount of new functionality. This steady progress is only possible by using abstract modeling. Hence, an ecosystem around *Electronic System Level* (ESL) design has been developed by the EDA companies. A major language for ESL design is the C++ class library SystemC [1], [2], [3]. Besides abstract modeling, SystemC supports hardware/software co-design and the creation of heterogeneous systems (containing digital, analog and RF). A major approach in ESL design is the creation of advanced virtual prototypes. These prototypes heavily use *Transaction Level Modeling* (TLM) to abstract functionality, communication and timing [4]. Based on the SystemC TLM 2.0 standard [5] TLM models can be easily re-used in different projects and IP integration has become a simple task. Overall, high-speed simulation, architecture evaluation and early software development is standard practice in SystemC-based ESL flows today.

*This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the project SANITAS under contract no. 01M3088 and by the German Research Foundation (DFG) within the Reinhart Koselleck project DR 287/23-1.

However, the steadily increasing functionality results in large and complex ESL models, even if unimportant details at the higher levels have been abstracted using TLM. Thus, understanding these models is a non-trivial task. During the design of an ESL model the specification is implemented as a large set of system features. Following the IEEE Standard 829 [6] a *feature* is a distinguishing characteristic of the ESL model. More specifically, besides concrete functional features (e.g. running a video filter, converting audio data) also other features such as performance or dependability are important properties of an ESL model.

In this paper, we propose an approach to improve design understanding of ESL models by localizing features automatically, that is, identifying the respective source code implementing the concrete feature. The approach is based on simulation. We compare a simulation run where the feature is activated against a run where the feature is not active. During simulation, code coverage techniques are used such that we can determine the differences between both runs. Essentially, this delta identifies the relevant source code implementing the feature.

Major ESL design tasks like for instance refinement, partitioning or optimization heavily depend on this information. In general, since the implementation of different features are typically spread over several team members this knowledge is distributed. Furthermore, the integration of new members (e.g. system architects, software developers or hardware engineers) is accelerated since the design familiarization with our approach not only consists of reading long text books and manual code inspection.

In summary, the contributions of this paper are:

- Pinpointing to source code implementing a feature
- Following the information flow when presenting the feature code
- Highlighting involved SystemC modules

In an experimental evaluation we demonstrate the advantages of our approach for a complex image processing system. Different features can be automatically localized very fast with very little user interaction. In this way, the design understanding is significantly improved.

The remainder of the paper is structured as follows: Section II describes related work. In Section III our feature localization approach for SystemC ESL models is introduced. The evaluation is given in Section IV. Finally, the paper is concluded in Section V.

II. Related Work

Feature localization has been studied intensively in the context of software engineering. However, so far no approaches have been presented for ESL models. The goal of [7] is to help the software engineer during the process of software maintenance, i.e. if the functionality of a software system needs to be updated or extended. Different test-cases are executed and in combination with a test coverage monitor the approach can be used to discover where a particular program feature is implemented. A quite similar approach has been presented in [8]. Both approaches consider a set of test-case and compute a categorization. These dynamic approaches have been advanced by incooperating static analysis techniques. In [9] computational units are identified that specifically implement a feature as well as the set of jointly or distinctly required computational units for a set of features.

In the context of fault localization, coverage-based methods have been developed to represent how source code lines act during passed and failed tests [10]. Several coverage metrics can be integrated to improve the results [11].

III. Feature Localization for SystemC Models

In this section the proposed approach for feature localization in SystemC ESL models is introduced. First, the general idea is described. Then, we present our method in more detail.

A. General Idea

For finding a feature in a SystemC ESL model two simulation scenarios are required. The first scenario needs to include the requested feature and the second scenario must not include the feature. Then, by comparing the activated source code of both scenarios after simulation we can extract the source code which has only been executed in the first scenario. These code lines show the implementation of the requested feature. The involved SystemC modules can now be highlighted. If more than one SystemC module is involved, the respective source code is presented with regard to the activation during simulation.

In the following we explain our approach in more detail. At first, we explain the terms feature and scenario for SystemC ESL models. Then, we define the problem and present our implementation.

B. Features and Scenarios

As mentioned above several definitions of a feature have been proposed in the software community, in particular in the context of feature-oriented software development (for an overview see [12]). Since we deal with abstract SystemC TLM models, a *feature* is a unit of behavior that satisfies a functional and/or non-functional requirement. For example, a feature could be to open the menu, or to save a picture, or to save the current state of a system.

A *scenario* is a sequence of inputs to a SystemC TLM model. Here is a concrete example:

- Pressing *power* to turn on the system
- Activating the camera by pressing *camera mode*
- Taking a picture by pressing the *shutter release*
- Showing the taken picture by pressing *play mode*
- Turning off the camera by pressing *power*

Based on these terms, we give the problem formulation next.

C. Problem Formulation

Before we give the problem formulation, we provide the used notations:

- S denotes the set of scenarios, and
- $F = \{F_1, ..., F_m\}$, $m \in \mathbb{N}$ denotes the set of features.

For localization of the implementation of the feature F_i we need two scenarios, i.e. S_a and S_n, respectively. S_a and S_n are almost identical except that S_a activates F_i and S_n does not activate F_i. Now, the desired source code implementing F_i can be determined using code coverage techniques as:

$$cov(S_a) \setminus cov(S_n) = code(F_i),$$

where $cov(A)$ returns the covered source code lines of scenario A, and $code(B)$ represents the relevant source code of feature B.

In the next section the implementation of this approach is introduced.

D. Implementation

As mentioned in the previous section two scenarios activating (S_a) and not activating (S_n) the feature need be to defined by the user. This is a manual step. For the creation of these scenarios the specification, documentation or existing test cases can be used.

Now the SystemC ESL model is simulated with scenario S_a, and afterwards with scenario S_n. During both simulation runs we use gcov [13], a test coverage program for C++, to log how often each line of code executes and what lines of code are actually executed. A concrete example showing the logged coverage information is:

```
        -:  174: // create grey image
   391534:  175: for(y=0;y<height;y++)
        -:  176: {
250451520:  177:   for(x=0;x<width;x+=3)
        -:  178:   {
250060800:  179:     pixel  = *(image+(y*width)+x);
250060800:  180:     *(grey+(y*grey_width)+(x/3))
                       = pixel;
        -:  181:   }
        -:  182: }
```

The first number in each row shows how often a line of code has been executed. The second number is the actual line number followed by the original source code line.

By comparing the logged coverage information of both scenarios we can extract the source code lines which have only been executed in one scenario, i.e. in S_a. Hence, these lines of code are the implementation of the requested feature. Moreover, after localizing the relevant code lines the involved SystemC modules are known too. By using visualization techniques similar to [14] we can additionally provide a graphical representation. In general, the modules and their interconnection is available. But with the localized source code we can now only highlight the modules which belong to the requested feature. This information gives additional information on how the system is implemented and which module is responsible for which task.

If the source code of a feature is spread over several SystemC modules all involved modules are highlighted. For a better understanding of how the feature is implemented all involved modules are ordered with respect to their execution. In that way we can visualize where a feature starts and how the execution continues. To get the correct order of the involved modules a second run of the scenario which includes the feature is required. On the second run time-stamps are added to the determined lines of code which enables the ordering.

In the following section we present the experimental evaluation.

IV. Evaluation

This section presents the experimental results for the proposed approach. At first, we describe our test environment. Then, we explain the test cases for our feature localization approach. Finally, the results are discussed.

A. Test Environment

We are using an improved version of the image processing system described in [15]. This system determines the position of a PlayStation™ Move Controller [16] in a video stream. On the top of the controller is an illuminated ball. We calculate its 3D position, i.e. x, y, z coordinates where the z coordinate can be derived from the radius of the ball.

Figure 1 shows the architecture of the SystemC TLM model. The system is controlled by the *Instruction Set Simulator* (ISS) Or1ksim [17].

The image capturing module receives the pictures from a camera and sends them via the bus to the memory. This module also initializes the image processing of the *grey* converter module which converts the colored image to a greyscale image. Then, edge detection is carried out in the *sobel* module. The result is used for performing a *hough* transformation [18]. Now, the circle around the ball of the controller is determined (see *find circles* module). The position of the controller will be displayed on the video stream produced by the *image viewer*. For a more detailed description of the system we refer to [15].

The calculated position of the controller can be used to interact with the system. For example, starting a game which can be played via the controller. When starting the game a white circle, as depicted in Figure 2, appears on the video stream. The goal of the game is to cover this circle with the top of the controller. If the circle is covered the player gets one point and the circle will appear at a new position. By reaching a certain number of points the level of difficulty will be increased by flipping and rotating the video which is computed in the *modifier* module.

For the evaluation process of our feature localization approach we replaced the camera stream with video files. Hence, we ensure that the input to the system is deterministic.

B. Test Cases

In Test Case 1 we aim to localize the feature F_1: Switching to the next level. For localization we use the following scenario S_{1a} which activates F_1:

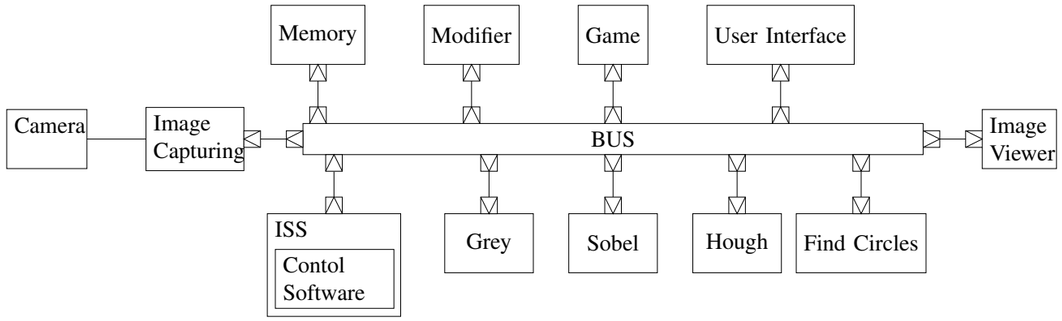

Fig. 1. Architecture of image processing system

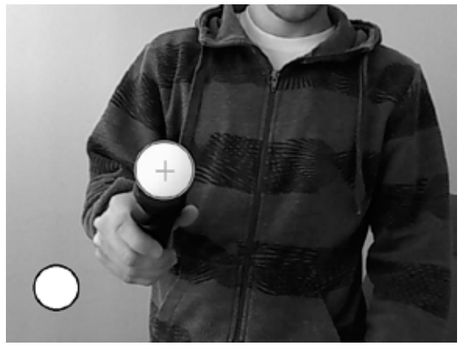

Fig. 2. Example video output

- Turning on the system by starting the simulation
- Starting the game with the controller
- Collecting points until we reach the second level

We compare S_{1a} with the almost identical scenario S_{1n}. The only difference to scenario S_{1a} is to get one point less than needed to reach the next level.

To make the system robust different dependability measures have been implemented [15]. Hence, a damaged camera can be simulated to validate the one of the dependability measures. The idea is to introduce some image artifacts using mutation on the image data. The respective mutations are injected using XML-definitions (e.g. size of the artifacts) following [15]. Now, in Test Case 2 the feature F_2 is: Controlled modification of image data. For our approach we use a standard scenario but including the injection of 5% random artifacts for S_{2a}, and only the standard scenario without any error injection as S_{2n}, respectively.

Test Case 3 also simulates an error similar to Test Case 2. However, Test Case 3 represents a corrupted connectivity between the *bus* and the *grey* module. The corrupted connectivity is represented as mutated TLM attributes, in particular the TLM command and the data length. Unlike Test Case 2 it is more important if the mutations can be fixed and where they will be fixed. The goal is to locate feature F_3: Error correction of communication. Like in S_{2a} of Test Case 2 we activate in S_{3a} the communication errors, and in S_{3n} no errors are injected.

TABLE I
EXPERIMENTAL RESULTS

Feature	# Modules	# Methods	# LOC
F_1	2	2	14
F_2	0	3	141
F_3	1	5	192

C. Results and Discussion

Table I summarizes the results of all test cases. The first column *Feature* gives the requested features of the test cases. The column *Modules* gives the number of modules which includes located source code of the feature. The respective number of methods are given in column *Methods*. Finally, the column *LOC* reports the actual number of lines of code for the identified relevant source code implementing the feature.

In Test Case 1 two relevant SystemC modules have been identified by our approach. Figure 3 shows the involved modules for feature F_1 in the order of execution, i.e. at first the *game* module is active followed by the *modifier* module. A screen shot showing the relevant feature code is depicted in Figure 4. As can be seen the variable mem−>level of the module *game* will be increased when points%points2nextLvl becomes zero. As soon as the mem−>level will be increased from 0 to 1 the

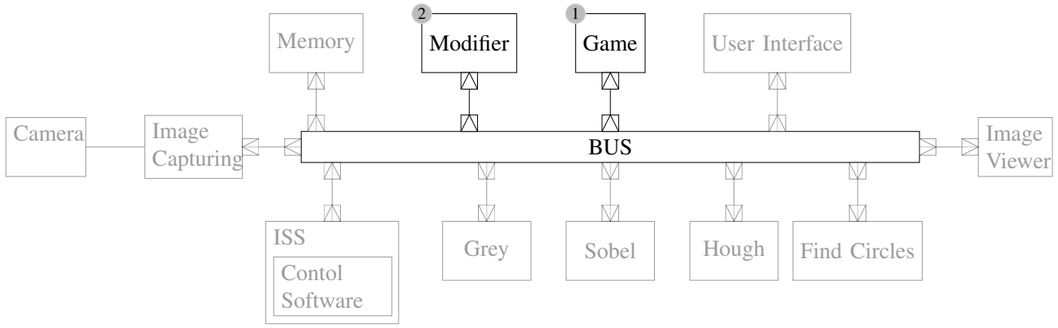

Fig. 3. Involved modules for Test Case 1

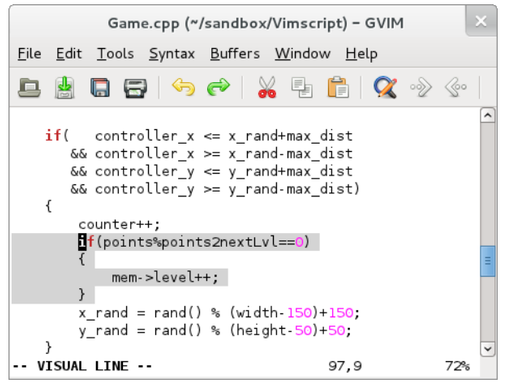

Fig. 4. Highlighted code in game module of feature F_1

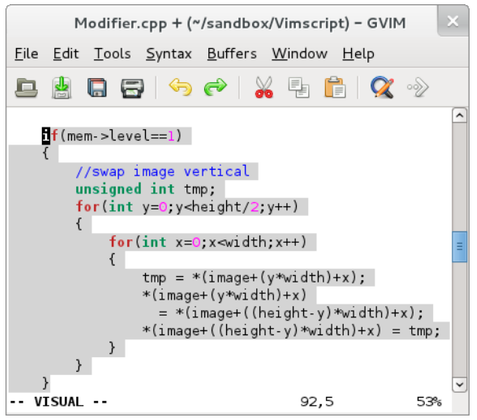

Fig. 5. Highlighted code in modifier module of feature F_1

code in the SystemC *modifier* module becomes active (see Figure 5). In this case the input image from the camera will be swapped vertically. Using this information a developer can easily add additional conditions for reaching the next level or change the swap axis for the image.

In Test Case 2 no modules of the system are involved for feature F_2. The modification has been implemented in the extended version of the TLM library (see [15]). Three methods have been identified to be relevant by our feature localization approach. Taking a closer look on these methods we saw that one method is responsible to load the specified modifications from the XML file, the second method is looking for modifications and calls the third method if modifications are specified.

For Test Case 3 our approach also localized code of the extended TLM library to be relevant for feature F_3. In particular two methods in the *grey* module are located which correct the communication error.

In summary, our feature localization approach clearly improves design understanding since automatically relevant source code as well as involved modules (or methods) for a feature are identified.

V. CONCLUSIONS

In this paper we have presented an approach for feature localization in ESL models. Basically, we compare a run which invokes the feature to a run which does not invoke the feature. Based on code coverage techniques we then compute the relevant source code implementing the feature. Besides the code we are also able to show the involved SystemC modules according to their order of activation. For a complex SystemC ESL design, implementing a video image processing system, we localized different features very fast.

REFERENCES

[1] Accellera Systems Initiative, "SystemC," 2012, available at http://www.systemc.org.

[2] D. C. Black and J. Donovan, *SystemC: From the Ground Up*. Springer-Verlag New York, Inc., 2005.

[3] D. Große and R. Drechsler, *Quality-Driven SystemC Design*. Springer, 2010.

[4] L. Cai and D. Gajski, "Transaction level modeling: an overview," in *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2003, pp. 19–24.

[5] *IEEE Standard SystemC Language Reference Manual*, IEEE Std. 1666, 2005.

[6] "IEEE standard for software and system test documentation," *IEEE Std 829-2008*, 2008.

[7] N. Wilde and M. C. Scully, "Software reconnaissance: Mapping program features to code," *Journal of Software Maintenance*, vol. 7, pp. 49–62, January 1995.

[8] W. Wong, S. Gokhale, J. Horgan, and K. Trivedi, "Locating program features using execution slices," in *Application-Specific Systems and Software Engineering and Technology, 1999. ASSET '99. Proceedings. 1999 IEEE Symposium on*, 1999, pp. 194 –203.

[9] T. Eisenbarth, R. Koschke, and D. Simon, "Locating features in source code," *IEEE Trans. Software Eng.*, vol. 29, no. 3, pp. 210–224, 2003.

[10] J. A. Jones, M. J. Harrold, and J. Stasko, "Visualization for fault localization," in *Proceedings of thethe Workshop on Software Visualization*, 2001.

[11] R. A. Santelices, J. A. Jones, Y. Yu, and M. J. Harrold, "Lightweight fault-localization using multiple coverage types," in *ICSE*, 2009, pp. 56–66.

[12] S. Apel and C. Kästner, "An overview of feature-oriented software development," *Journal of Object Technology (JOT)*, vol. 8, no. 5, pp. 49–84, July/August 2009.

[13] "gcov a test coverage program," http://gcc.gnu.org/onlinedocs/gcc/Gcov.html.

[14] D. Große, R. Drechsler, L. Linhard, and G. Angst, "Efficient automatic visualization of SystemC designs," in *Forum on specification and Design Languages*, 2003, pp. 646–657.

[15] M. Michael, D. Große, and R. Drechsler, "Analyzing dependability measures at the Electronic System Level," in *Forum on specification and Design Languages*, 2011, pp. 1–8.

[16] "Playstation™ move motion controller," http://uk.playstation.com/psmove.

[17] J. Bennett, *Or1ksim User Guide*, 2010.

[18] D. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111 – 122, 1981.

SystemC Based Verification of Complex Heterogeneous Systems

Ralph Görgen*, Henning Kleen*, Jan-Hendrik Oetjens†, Peter Jores†, Wolfgang Nebel‡

* OFFIS Institute for Information Technology, 26121 Oldenburg, Germany

ralph.goergen / henning.kleen@offis.de

† Robert Bosch GmbH, 72762 Reutlingen, Germany

jan-hendrik.oetjens / peter.jores@de.bosch.com

‡ Carl von Ossietzky University, 26111 Oldenburg, Germany

nebel@uni-oldenburg.de

***Abstract*—This paper presents an industrial verification flow for heterogeneous electronic hardware components. The verification process is described with the help of the SystemC AMS model of a current controller, which is part of an automotive anti-lock breaking system. To enable the verification of heterogeneous designs, an existing SystemC based framework is extended to support mixed-signal test-bench modules implemented in SystemC AMS. Additionally, a concept to re-use digital test-bench modules for TLM and RTL verification is shown.**

I. INTRODUCTION

The domain of automotive electronics system design is characterised by specific constraints and environmental conditions that significantly differ from those in other design domains like mobile communications, multimedia or high performance computing. This results from the fact that automotive electronics systems are often involved in safety-critical car-functions that have to be highly reliable and robust over a long lifetime, and are used in harsh environments in terms of vibrations, temperature changes, or electromagnetic interferences. Thus, verification is one of the main tasks in the development process. The heterogeneity of many automotive electronics systems complicates their verification even more.

In this contribution, we describe the verification process of a current controller which is part of an anti-lock breaking system. The current controller model is implemented in SystemC AMS [1]. This modelling language provides good facilities for the implementation of complex heterogeneous systems together with high simulation performance. In our case, its main benefit is the possibility to integrate analogue components into virtual platforms very easily. The verification is done with a verification framework for heterogeneous system, which has been developed at the semiconductor development division of Robert Bosch GmbH together with OFFIS.

The paper is organised as follows: Section II gives a detailed description of the current controller implementation. In Section III, the verification objectives are given. Section IV introduces the verification framework, before the concrete implementation of the verification scenario and the verification results are described in Section V. Section VI concludes the paper and gives a short outlook on future work.

II. USE-CASE CURRENT CONTROLLER

Driving the various actuators in a car often requires current controllers. In this paper we use an anti-lock braking system (ABS) as example. A detailed description of the whole system can be found in [2]. The ABS ECU (electronic control unit) independently controls the braking valves of all wheels preventing the wheels to be locked up and thus avoid skidding of the car. This requires a precise control of the current through the inductor of the magnetic valve. In total, four current controller channels are required; one of them is described in the following text.

A. Operation Breakdown

The target value of the current is given by a pulse-width modulated signal (PWM). An internal control circuit compares the actual value of the current with the target value and determines the excitation of the power transistor in the output stage. The output stage is realised as chopper output to avoid overheating of the power transistor. The chopper frequency is given by an internal oscillator. The current controller also contains a surveillance logic that checks for example the power supply (VZP, UVR) and shorts or opens at the output

(LX) of the output stage. The system architecture is represented in Figure 1.

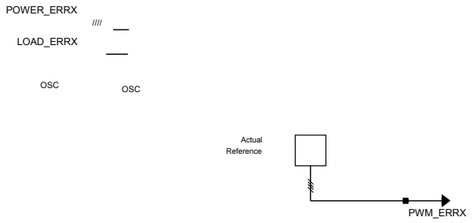

Figure 1. Architecture of the Current Controller

B. SystemC AMS Implementation

When implementing the current controller in SystemC AMS several requirements had to be considered:

- The valve load has to be modelled as an electrical element in order to vary the inductance resp. the resistance value.
- The ports UVR, LX and $PGNDX$ have to be electrical terminals to be able to simulate the transient behaviour (charging/discharging of the storing elements) of this circuit which is important for the overall system behaviour (e. g. timing).
- The influence of the power supply has to be considered also.

The SystemC AMS implementation makes use of all 3 models of computation (MoC) that come with Standard SystemC AMS Extension 1.0. For the description of the load and the power transistor *ELN* (Electrical Linear Network) primitives (inductor, switched resistor) have been used. The control circuit is described using *LSF* (Linear Signal Flow) elements and for the remaining functions *TDF* (Timed Data Flow) models have been developed (Figure 2).

C. Output Stage

Due to the requirement to use an electrical model of the valve load, the power transistor and the freewheeling diode of the output stage had to be implemented as electrical model. In this circuit the power transistor acts as a switch. For correct functionality the switch has to

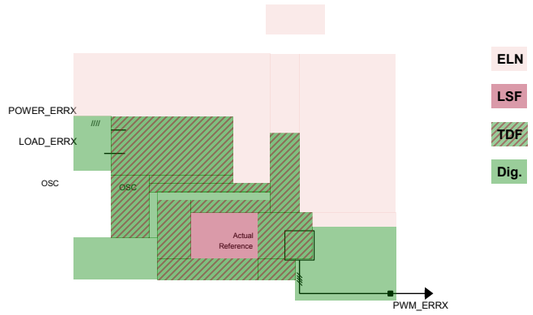

Figure 2. MoC Usage in SystemC AMS Model

change between a specified on- and an off-resistance. The freewheeling diode is required to avoid that the output node exceeds the supply voltage by more than diode voltage when switching off. The nonlinear behaviour of these components (diode, switch) has been described using the TDF MoC and encapsulated in models with electrical ports. Together with the load, this leads to a realistic description of the output behaviour. Figure 3 shows the output stage in combination with a pull down resistor (R_{LX}). The valve load is described by a series connection of an inductor (L) and a resistor (RL).

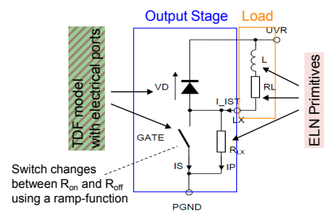

Figure 3. Output Stage in SystemC AMS

a) Diode: Currently, SystemC AMS 1.0 supports only linear electrical networks (ELN). Thus, nonlinear devices – like a diode – require a special modelling. In this case, the diode behaviour is described by a series connection of voltage source and a resistor, where the DC voltage and the resistance value can be varied by an external signal. The control of these values is carried out using a TDF model that calculates the required values based on the measured diode voltage. Due to sampling the setting of the DC voltage and the resistance value has to be delayed by one time step. A parallel capacitor eliminates possible voltage peaks.

b) Switch (Transistor Power): The power transistor has been modelled as a switch with a given on- and off-resistance, R_{on} and R_{off}. The use of a non-linear ramping function results in a realistic switching behaviour. Oversampling has been used to perform the ramping between two adjacent sample points of the remaining model.

D. Controller

For this application a PI controller is required that has been implemented as LSF model. As most other parts of the model are realised as TDF models TDF to LSF and LSF to TDF converters are required at the inputs resp. output (see Figure 4).

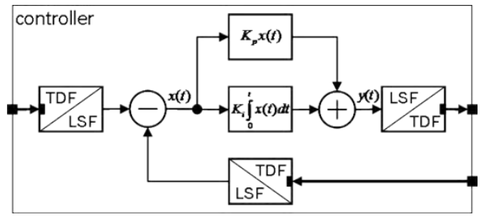

Figure 4. Block Diagram of the PI Controller

The value of the integration capacitor C_{INT} has been used as controller parameter and thus the external capacitor has been left out.

The TDF MoC requires delaying the actual value of the measured output-current by one time step. This is done in the output stage in the present implementation.

E. Other Models

All other models are realised using the TDF MoC.

The logic input signal $PWMX$ is converted to a TDF signal. The reference value is determined from the duty cycle of the PWM signal using a low pass filter, modelled with the predefined "ltf"-function. The value of the external capacitor C_P (connected to $CSWX$) determines the cut-off frequency of the low-pass filter. In the present implementation, the external capacitor has been omitted and its value is a parameter to the low-pass filter model. The oscillator frequency is determined by the external capacitor C_{OSZ}. Like in the previous example, the capacitor has been omitted and the capacitance value is used to parameterise the model. The current controller module contains one oscillator for all four channels.

F. Surveillance Logic

The input signal $PWMX$ and the output LX are monitored. In case of an error corresponding error signals, PWM_ERRX, $POWER\_ERRX$, $LOAD\_ERRX$, are set. The signal PWM_ERRX is set TRUE, if for more than eight oscillator cycles the $PWMX$ signal is high. The second error signal, $POWER\_ERRX$, checks whether the supply voltage UVR is connected to the valve load and $LOAD\_ERRX$ is used to indicate a disconnected load or a shorted output. If the power transistor is switched off and the voltage at port LX becomes lower than a certain value, the corresponding signal is set TRUE. When the fault disappears, this signal stays true for another 3-4 cycles before it is deasserted.

III. Verification Scenario

The verification scenario is intended to check the correct functionality of the fault-detection signals. To run the current controller, the test-bench has to provide five components:

- two voltage sources representing UVR and VPZ,
- one load representing the valve,
- one module to control the PWM signal generation,
- one unit to observe the fault detection signals.

The test-case should cover three different situations. The first is triggering *pwm_err*. Therefore, the PWM control module should apply several, valid and faulty, PWM sequences to the current controller. In the second situation, the *power_err* signal is supposed to be triggered. Hence, the test-case has to modify the value of the UVR voltage source and to check the intended fault signalling in the fault observation unit. The third situation should check the functionality of *load_err*. Therefore, the parameters and the connectivity of the load module have to be changed.

This scenario requires a verification environment with mixed-signal capabilities. Analogue test-bench elements are necessary to realise UVR and the load, whereas the PWM control and fault detection units are digital components.

IV. Verification Strategy

To implement the verification scenario described above, we used a company-internal verification framework developed at Robert Bosch GmbH in cooperation with OFFIS[3]. It is called *VF* in the following text. The basic idea of this framework is to separate test-bench and test-case descriptions. Beyond automated test runs,

it provides facilities to support *constrained-random verification* and *coverage-driven verification*. Initially, it has been designed to verify RTL descriptions of hardware components. Now, we extended this framework to allow the verification of complex heterogeneous systems.

A. Existing Verification Approaches

A very common verification approach is the standardised *Universal Verification Methodology* (UVM) [4]. Its focus is to improve the interoperability of verification IP from different providers and to increase their re-use facilities. UVM is intended for functional verification based on simulation; constrained-random verification is supported as well as coverage-driven verification. Currently, implementations of the UVM concept are available in the languages SystemVerilog, SystemC, and e. An extension of UVM for mixed-signal verification is described in [5]. Here, Verilog-AMS is used to implement the lower-level mixed-signal functions of the test-bench components.

A drawback of UVM is that initially a big effort is required to set up a test-bench and to implement test-bench modules.

B. Verification Framework (VF)

In contrast to UVM, the implementation of test-bench modules in the VF is much more efficient in our case. It is fully integrated into our tool environment and well known by the verification engineers. The framework is long time in use; a lot of know-how as well as large numbers of verification modules and test-case descriptions are available already. As a result, setting up a test-bench requires less effort.

In Figure 5, a typical verification scenario is shown. The design under verification (DUV) is connected to several test-bench modules, which are used to stimulate the DUV's inputs and to validate its outputs. The test-case is given as a script file containing commands for the particular test-bench modules. These commands vary from simple configuration commands up to complex communication sequences. Furthermore, the command script can contain synchronisation commands and program control structures. Each test-bench module is derived from a test module base class. It contains a so-called command loop that asks the test-bench controller for new commands whenever the execution of the current command is finished.

Supported design languages for the DUT are Verilog, VHDL, and SystemC. Test-bench modules can be implemented in VHDL or SystemC; today, many of them are implemented in SystemC because of its major

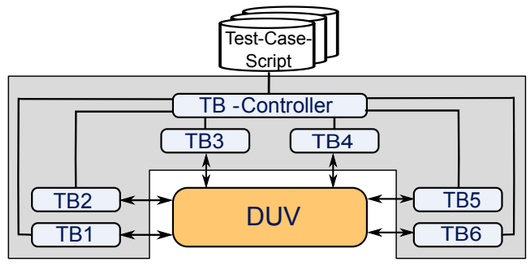

Figure 5. Test-Bench Example

expressive power. In addition, an FPGA interface and a JTAG module have been developed. The first enables hardware accelerated simulation, and the second can be used facilitate on-chip debugging.

C. Integration of Analogue Components

The verification task described here requires analogue test-bench components as mentioned in Section III. To handle this, we integrated SystemC AMS into the VF. Since the framework itself is based on SystemC, this is possible without major difficulties. Only a few slight changes had to be applied to the test module base class. ELN and LSF elements can be used without any restrictions in test-bench modules. But the use of TDF has some limitations. The test module base class is derived from `sc_module`, TDF modules have to be derived from `sca_tdf::sca_module` instead. As a result, a test-bench module itself cannot be a TDF module. A TDF module has to be instantiated in a test-bench module and its ports must be forwarded. In this form, all TDF features can be used, and commands to control the module from the test-case descriptions are possible as well.

D. Validation of Refinements Steps

A common verification challenge is checking the correctness of refinement steps, for instance the refinement from TLM to RTL. A test-bench has been set up to verify the TLM implementation of the DUV. Then, a similar test-bench is required to ensure that the RTL implementation of the DUV shows the same behaviour. Instead of creating a new test-bench, we developed a methodology that allows re-using a TLM test-bench as well as the test-case descriptions in an RTL scenario. It is based on the configurability of the particular test-bench modules; they allow switching between TLM and RTL with very little effort.

The difference between TLM and RTL is mainly in the communication infrastructure. A TLM model uses sockets and method calls to communicate whereas an RTL component has ports and signals to be connected to others. In our approach, each access to the communication infrastructure is encapsulated by an additional communication adaptor. The adaptor provides the test-bench module with methods to execute a single communication access or a specific communication sequence. The test-bench module calls these methods instead of using the actual communication infrastructure directly. Examples of such adaptor methods are given in Listing 1.

```
// TLM adaptor
struct sw_tlm_if : public sw_if_base {
  // TLM-2 socket, base protocol
   simple_initiator_socket <sw_tlm_if> socket;
  virtual void if_write( int addr, int value ) {
    // set up transaction object
     ...
    socket->b_transport( *trans_obj, delay );
     ...
  }
  ...
};

// RTL adaptor
struct sw_rtl_if : public sw_if_base {
  sc_core::sc_out<bool> req;
  sc_core::sc_in<bool> ack;
  sc_core::sc_out<int> addr;
  sc_core::sc_out<int> data;
  ...
  virtual void if_write( int addr, int value ) {
    // realise communication protocol
    req.write(true);
    while ( ack != true ) wait();
    addr.write(addr);
    data.write(data);
     ...
  }
  ...
};
```

Listing 1. Communication Adaptor

In the instantiation of a test-bench module, the desired adaptor version is annotated in form of a template argument (Listing 2). Since the test-bench module is derived from the adaptor class, the binding is the same as in standard SystemC.

As a result, the same test-bench modules and test-case descriptions can be used to verify the TLM and the RTL implementation of a component.

```
// TLM
tbm<sw_tlm_if> tbm1;
tbm1.socket(duv.socket);
...
// RTL
tbm<sw_rtl_if> tbm1;
tbm1.req(s_req);
tbm1.ack(s_ack);
tbm1.addr(s_addr);
tbm1.data(s_data);
...
```

Listing 2. Instantiation of a Test-Bench Module

V. Test-Bench Implementation and Verification Results

In this section, we describe the realisation of the verification scenario from Section III implemented in the verification framework form Section IV. As stated before, five test-bench components are required, the voltage sources UVR and VPZ, the load, the PWM control unit, and an observer for the fault signals. An overview of this test-bench scenario is shown in Figure 6.

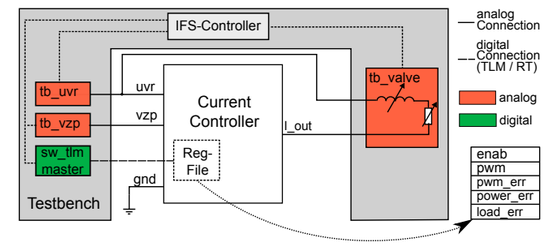

Figure 6. Test-Bench of Current Controller

The voltage sources are modelled in the ELN formalism. Each of them provides a command to set the voltage value from the test-case script. The load is modelled as an ELN component as well. It consists of a resistance and an inductance, both configurable. In addition, it contains a switch to connect and disconnect the load from the DUV. For the test-case implementation, it provides commands to configure the resistance and the inductance and to open or close the switch. The unit to control the PWM signal and the fault observation unit are implemented in a single software module. The software task runs several PWM profiles and generates error messages whenever a fault has been detected.

In addition to the test-bench, a test-case script is required. A part of it that checks the correct functionality

of the $LOAD\_ERRX$ signal is shown in Listing 3. It starts with the initialisation of the analogue test-bench components with valid values (lines 1 to 4). Then, the software task is started to run a PWM profile (line 7). Here, the output current is gradually increased to 1000‰ of the maximum output current with a slope of 2. After a simulation time of 25 ms, an error is forced: The load is disconnected from the controller. After another millisecond, the load is reconnected and the normal operation should continue.

```
1  load RESISTANCE 5
2  load INDUCTANCE 0.025
3  uvr_source VOLTAGE 14.0
4  vzp_source VOLTAGE 14.0
5  sw_task WAIT simtime 0.5 ms
6  ALL SYNC ALL
7  -- start sw task
8  sw_task RUN_SW 1000 2
9  -- disconnect load at 25 ms
10 load WAIT simtime 25 ms
11 load CONNECTION open
12 -- reconnect load at 26 ms
13 load WAIT simtime 26 ms
14 load CONNECTION close
15 ...
```

Listing 3. Command File Example

Listing 4 shows the output of the test-case simulation. Like expected, the software module reports an error after the load has been disconnected and resumes its normal operation when it is reconnected again. Hence, the current controller recognises the erroneous situation and signals it correctly.

```
[load, 0 s] resistance set to 5 Ohm
[load, 0 s] inductance set to 0.025 Henry
[sw_task, 500 us] SW started with parameters 1000 and 2
[load, 25 ms] connection set to open
WARNING (25163220 ns)
 [sw_task] Error detected! Resetting. Error reg. val: 3
[load, 26 ms] connection set to closed
INFO (27254040 ns)
 [sw_task] Error signal cleared, resume normal operation
```

Listing 4. Test Result

In the first implementation of the current controller, the control registers of the PWM unit as well as the error signals are accessible via TLM 2.0 sockets. Thus, the according test-bench module has to provide a TLM socket interface, too. Then, the digital parts of the current controller are refined. The TLM interfaces are replaced by RTL interfaces, i. e. signals and ports. To allow the validation of this refinement step without changing the test-bench, the software module is implemented as described in Section IV-D. Thus, only the parameter of the software test-bench module in its instantiation and the according bindings have to be changed. The rest of the test-bench as well as the test-case script remains unchanged. Finally, another simulation is run and its results are compared to the previous ones. As long as they are identical, the refinement of the current controller has been done correctly.

VI. Conclusion

In this contribution, we have presented the application of a SystemC based verification framework for heterogeneous systems with a complex automotive electronics design, the model of a current controller. The current controller is part of an ABS system and contains analogue as well as digital electronic components. The model is implemented in SystemC AMS and uses all available formalisms, ELN, LSF, and TDF. To verify the model, an existing industrial verification framework has been extended to support heterogeneous systems. In addition, a methodology to re-use verification modules for TLM and RTL has been presented. With this, the validation of refinement steps can be done with only very slight changes of the test-bench. This verification framework has been used to set up a test-bench and a test-case description to check the functionality of the current controller.

Acknowledgements

This work has been partially funded by the German Bundesministerium für Bildung und Forschung (BMBF) in the projects SANITAS (01M3088) and SyEnA (01M3086).

References

[1] *Standard SystemC AMS Extension 1.0*, 2010, http://www.accellera.org/downloads/standards/systemc/ams.

[2] Robert Bosch GmbH, *Automotive Handbook*. John Wiley & Sons, 2011, pp. 806–817.

[3] R. Lissel and J. Gerlach, "Introducing new verification methods into a company's design flow: an industrial user's point of view," in *DATE*, R. Lauwereins and J. Madsen, Eds. ACM, 2007, pp. 689–694.

[4] *Standard Universal Verification Methodology*, 2011, http://www.accellera.org/downloads/standards/uvm.

[5] N. Khan, Y. Kashai, and H. Fang, "Metric driven verification of mixed-signal designs," in *Proceedings of DVCon 2011*, 2011.

Ein Bondgraphberechnungsmodell für SystemC AMS

Torsten Mähne
Laboratoire d'Informatique de Paris 6 (LIP6)
Université Pierre et Marie Curie (UPMC), Paris
E-Mail: Torsten.Maehne@lip6.fr

Alain Vachoux
Laboratoire de Systèmes Microélectroniques (LSM)
École Polytechnique Fédérale de Lausanne (EPFL)
E-Mail: alain.vachoux@epfl.ch

***Zusammenfassung*—In diesem Beitrag stellen wir unsere Erweiterung der C++-basierten Simulationsumgebung SystemC AMS zur Beschreibung multiphysikalischer Subsysteme mit Hilfe der Bondgraph- und Blockdiagrammformalismen vor. Die Architektur und Möglichkeiten des dazu implementierten Bondgraphberechnungsmodells (BG MoC) werden vorgestellt. Das Anwendungsbeispiel eines Vibrationssensors mit digitalem Frontend wird zeigen, wie die verschiedenen Berechnungsmodelle in SystemC AMS effektiv zusammenarbeiten können.**

I. Einleitung

Elektronische Einchipsysteme sind Schlüsselkomponenten einer immer größeren Vielfalt an Produkten, angefangen von Mobiltelefonen bis hin zu Kraftfahrzeugen. Neben komplexer digitaler Hard- und Software zur Datenverarbeitung und -speicherung integrieren sie immer mehr analoge/RF Schaltungen, Sensoren und Aktoren, um mit ihrer (analogen) Umgebung zu interagieren.

Ihr Entwurf verlangt die parallele Integration der sehr unterschiedlichen multidisziplinären Teilentwurfsprozesse. Eine gemeinsame Entwurfs- und Simulationsplattform zur Erstellung und Verfeinerung einer ausführbaren Spezifikation des Gesamtsystems auf einem hohen Abstraktionsniveau wird benötigt, die verschiedene Berechnungsmodelle (Models of Computation – MoCs) und ihre gekoppelte Simulation unterstützt. Für digital/analoge Hard-/Softwaresysteme ist das C++-basierte SystemC [1] mit seinen AMS-Erweiterungen [2] dabei sich als solche Plattform zu etablieren. Die von SystemC AMS bereitgestellten Berechnungsmodelle, Timed Data Flow (TDF), Linear Signal Flow (LSF) und Electrical Linear Network (ELN), eignen sich besonders zur Beschreibung der RF- und digitalen Signalverarbeitungssubsysteme. Sie sind über eine die TDF-Semantik nutzende Synchronisierungsschicht mit dem Discrete Event (DE)-Simulationskernel von SystemC gekoppelt (Abb. 1), der für die Ausführung der digitalen Hardwarebeschreibung verantwortlich ist.

SystemC AMS fehlt es noch an einer dedizierten Unterstützung zur Modellierung des nichtlinearen energieerhaltenden und -nichterhaltenden Verhaltens multiphysikalischer Subsysteme. Es existiert bereits ein Prototyp eines nichtlinearen Netzwerkberechnungsmodells, der SystemC AMS um zu VHDL-AMS und SPICE analoge Modellierungsmöglichkeiten erweitert [3]. Für eine potenziell effizientere Systemsimulation haben wir uns für den *Bondgraphformalismus* [4] entschieden, der die direkte Modellierung der Energieströme in einem multiphysikalischen System erlaubt und gut mit dem *Blockdiagrammformalismus* und anderen Datenflussformalismen harmoniert, die für die Beschreibung der Regelung und Signalverarbeitung in solchen Systemen zum Einsatz kommen. Zur Unterstützung beider Formalismen entwickelten wir ein neues *Bondgraphberechnungsmodell (BG MoC)* als Erweiterung für SystemC AMS (Abb. 1).

Im folgenden Abschnitt II werden wir die prinzipiellen Eigenschaften und Vorteile des Bondgraphformalismus im Kontext multiphysikalischer heterogener Einchipsysteme diskutieren. Abschnitt III stellt dann die Architektur und Möglichkeiten unseres neuen BG MoCs vor. Das Anwendungsbeispiel eines Vibrationssensors mit digitalem Frontend in Abschnitt IV wird zeigen, wie die verschiedenen Berechnungsmodelle (DE, TDF, BG) effektiv zusammenarbeiten können. Eine Zusammenfassung und Ausblick in Abschnitt V schließen unseren Beitrag ab.

II. Modellierung multiphysikalischer Systeme

Multiphysikalische Systeme können auf verschiedenen Abstraktionsebenen beschrieben werden, die sich unterschiedlich gut in SystemC AMS abbilden lassen. Als Beispiel betrachten wir den Vibrationssensor aus unserem Anwendungsbeispiel, das in Abschnitt IV vorgestellt wird. Er besteht im Wesentlichen aus einem mechanischen Resonator in einem Referenzrahmen, dessen relative Bewegung mit Hilfe eines elektrostatischen

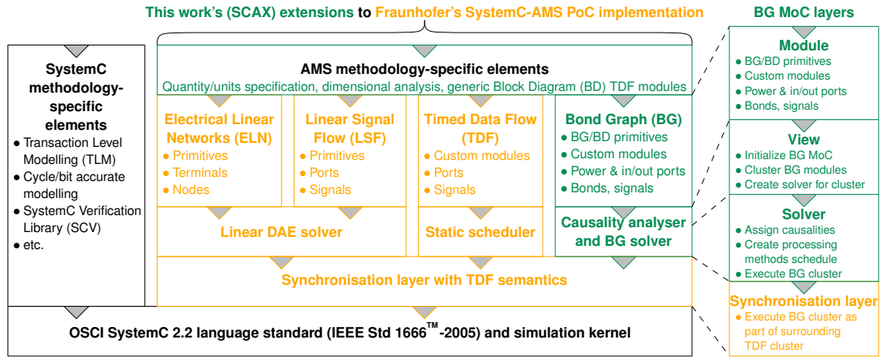

Abb. 1: Architektur von SystemC AMS mit dem neuen SCAX-Bondgraphberechnungsmodell.

Transducers in Form eines Verschiebungsstroms gemessen wird (Abb. 3). In einem ersten Schritt kann dieses System mit Hilfe domänenspezifischer Grundelemente (Feder, Masse, Dämpfer, Widerstand, Voltmeter, Spannungsquelle und Transducer) beschrieben werden. Das entstandene Modell gibt präzise die Struktur des physikalischen Systems wieder. Allerdings muss ein Simulator für diesen Ansatz dedizierte Modelle für jedes Grundelement der verschiedenen physikalischen Domänen bereithalten. Ist dies nicht der Fall, so können Analogien zwischen den Grundelementen verschiedener physikalischer Domänen genutzt werden um, z.B., mechanische Resonatoren in SPICE zu simulieren. Allerdings leidet die Klarheit darunter und Modellierungsfehler werden begünstigt. SystemC AMS unterstützt auf dieser Abstraktionsebene nur lineare elektrische Grundelemente.

Der Bondgraphformalismus [4] nutzt diese Analogien, um die Beschreibung multiphysikalischer Systeme durch einen reduzierten Satz von *generischen Grundelementen* zu vereinheitlichen, die Energiequellen (S_e, S_f), resistives/kapazitives/träges Verhalten (R, C, I), Energieumformungen (TF, GY) und Energieverteilungen durch sogenannte Junktionen (0, 1) repräsentieren. Jede domänenspezifische Beschreibung (z.B.: elektrische Schaltung, mechanisches Mehrkörpersystem) kann auf systematische Weise auf einen Graphen abgebildet werden (Abb. 3), der den Energiefluss zwischen diesen Grundelementen beschreibt. Der Energiefluss zwischen zwei Grundelemente Pr_i und Pr_k wird durch einen *Bond* in Form eines Halbpfeils repräsentiert: $Pr_i \xrightarrow[f]{e} Pr_k$, dem eine Potential- (*effort e*) und eine Stromvariable (*flow f*) zugeordnet sind. Sie sind abhängig von der physikalischen Domäne, z.B.: Spannung V und Strom I für die elektrische Domäne und Kraft F und Geschwindigkeit v für die translatorische mechanische Domäne. Die Beschreibungsgleichungen der Grundelemente und die Regeln für ihre Verschaltung durch Strom- (0) und Potential-Junktionen (1) garantieren die Energieerhaltung.

Der Energieaustausch zwischen zwei Grundelementen bedingt, dass e und f in entgegengesetzte Richtung wirken. Visualisiert wird dies durch einen *Kausalitätsstrich*, der einem Ende eines jeden Bonds hinzugefügt wird. Er drückt aus, dass an dieser Seite e bekannt ist und damit $f := \Phi_k^{-1}(e)$. Damit ist auf der anderen Seite des Bonds f bekannt und $e := \Phi_i(f)$. Die Beschreibungsgleichungen der Grundelemente *erzwingen*, *bevorzugen* (z.B. aus numerischen Gründen) eine Kausalität (*effort-in* oder *flow-in*) oder lassen sie *frei*. Die sich ergebenden Randbedingungen erlauben es jedem Bond auf systematische Weise eine Kausalität zuzuweisen, z.B. mit der *Sequential Causality Assignment Procedure (SCAP)* [4].

Durch die Kausalitätszuweisung können Bondgraphen einfach mit Blockdiagrammmodellen integriert werden und in letztere transformiert werden. SystemC AMS unterstützt zeitdiskrete und lineare zeitkontinuierliche Blockdiagrammmodelle durch die TDF und LSF MoCs.

Ein wichtiger Aspekt zur Prüfung multiphysikalischer Systemmodelle ist es die Dimension und das zugehörige Einheitensystem für alle Konstanten, Variablen und Parameter eines Modells, die physikalische Größen repräsentieren, zu deklarieren. Durch Dimensionsanalyse können dann die Kohärenz der Modellgleichungen geprüft und Verschaltungsfehler zwischen Teilmodellen unterschiedlicher physikalischer Domänen detektiert werden.

III. Bondgraphberechnungsmodell in SystemC AMS

Der vorherige Abschnitt hat gezeigt, dass der Bondgraphformalismus die bestehenden Modellierungsfähigkeiten von SystemC AMS ideal in Richtung der Beschreibung nichtlinearer multiphysikalischer Systeme ergänzt. Zur Bewahrung seiner Vorteile haben wir uns für die Implementierung eines dedizierten *Bondgraphberechnungsmodells (BG MoC)* entschieden, das sowohl Bondgraph- als auch Blockdiagrammmodellierung unterstützt. Dieses baut auf der Synchronisierungsschicht von SystemC AMS auf, welche die Kommunikation mit den anderen Berechnungsmodellen (DE, TDF, LSF, ELN) ermöglicht. Das neue BG MoC ist Teil der von den Autoren entwickelten SystemC AMS extensions eXperiments (SCAX)-Bibliothek [5], die auf den quelloffenen Implementierungen des SystemC-Standards, OSCI SystemC 2.2, und seiner AMS Erweiterungen, Fraunhofer SystemC-AMS 1.0beta2, aufbaut. Die Implementierung vermeidet jegliche Veränderung der zugrundeliegenden Bibliotheken, in dem sie neben den standardisierten Funktionen nur die von SystemC-AMS zur Implementierung seiner eigenen Berechnungsmodelle bereitgestellten internen Funktionen nutzt. Abb. 1 zeigt die sich ergebende Softwarearchitektur.

Die *Modulschicht* des BG MoCs bildet die Nutzerschnittstelle zur Definition von Bondgraph- und Blockdiagrammgrundelementen, die als *BG-Module* mit *Anschlüssen* für *Bonds* und *gerichtete Signale* beschrieben werden. Zusätzlich kann jedes BG-Modul auch TDF- und DE-Konverteranschlüsse verfügen, die die Kommunikation mit den anderen Berechnungsmodulen ermöglichen. Bonds und gerichtete Signale sind, analog zu DE- und TDF-Signalen, als Kanalklassen implementiert und erlauben die Verschaltung von Modulinstanzen in einer strukturellen hierarchischen Beschreibung innerhalb eines normalen SC_MODULEs. Das BG MoC stellt eine Reihe von vordefinierten Bondgraph- (z.B.: S_e, S_f, R, C, I, TF, GY, 0, 1, MS_e, MS_f, MTF, MGY) und Blockdiagrammgrundelementen (z.B.: Quelle, Senke, Skalierer, Summierer, Multiplizierer, Integrator, Differenzier, Funktionsblock) bereit. Die von ihnen manipulierte physikalischen Domänen oder Größendatentypen können bei der Instanziierung von BG-Modulen, -Bonds, -Signalen oder -Anschlüssen als Template-Parameter übergeben werden. Zu diesem Zweck integriert sich die SCAX-Bibliothek mit der Boost.Units-Bibliothek [6], welche einen Größendatentyp quantity als Ersatz für **double** bereitstellt mit ihm eine statische Dimensionsanalyse während der Kompilation erlaubt. Alle BG-Modelle können so auf Verschaltungs-, Parametrierungs- und Berechnungsfehler geprüft werden, die durch inkonsistente Schnittstellen oder Gleichungen verursacht werden.

Listing 1 zeigt die Implementierung einer vereinfachten Version des generischen Kapazitäts-Eintors C der SCAX-Bibliothek. Die Struktur eines BG-Moduls ähnelt der eines TDF-Moduls mit seinen Callbacks für die Elaboration und Simulation. Die Klasse scax_bg::scax_C deklariert den Template-Parameter Domain, über welchen die physikalische Domäne spezifiziert wird. Von ihm werden mit Hilfe einer Traits-Klasse die Größendatentypen für die Speicherung von Potential, Fluss, Ladung, Zeit und den Parameter des C-Grundelements abgeleitet. Als Eintor verfügt es nur über einen Bondanschluss bp. Im Callback set_attributes() werden die Kausalitätsrandbedingungen spezifiziert. Aus numerischen Gründen bevorzugt das C-Grundelement die flow-in Kausalität. Diese Randbedingungen werden für die spätere Kausalitätszuweisung berücksichtigt. Dabei ruft das BG MoC den Callback propagate_causality() eines BG-Moduls auf, wenn es eine Kausalität an einen seiner Bondports zuweist. So kann die Kausalität entsprechend der Randbedingungen zwischen den Bondports propagiert werden. Nach Abschluss der Kausalitätszuweisung wird der Callback after_causality_assignment() aktiviert, in dem das BG-Modul ein oder mehrere Verarbeitungsfunktionen mit Hilfe des Makros SCAX_BG_METHOD() registriert und ihre jeweiligen E/A-Variablen spezifiziert werden. Die Callbacks unconverged(), reset() und update() benötigen BG-Module, die über Zustandsvariablen vom Typ scax_bg::scax_variable<T> verfügen. Diese Callbacks erlauben es dem BG-Löser die Anzahl der nichtkonvergierten Variablen abzufragen sowie ihren Wert für den aktuellen Lösungsschritt zu akzeptieren oder rückzuweisen.

Listing 1: Generisches C-Eintor-BG-Modul.

```
template<typename Domain> class scax_C : public scax_module {
public:
  // Determine data types for e, f, q, and t from physical domain.
  typedef typename scax_domain_traits<Domain>::effort_type effort_type;
  typedef typename scax_domain_traits<Domain>::flow_type flow_type;
  // ...
  // Linear parameter data type result of displacement_type / effort_type.
  typedef typename boost::units::divide_typeof_helper<
    displacement_type, effort_type>::type parameter_type;
  // ...
  scax_port<Domain> bp; // (Power) bond port.

  SCAX_BG_HAS_METHOD(scax_C); // Module registers callbacks.
  // Construct a 1-port capacitor from name, compliance, and initial state.
  scax_C(const sc_core::sc_module_name& nm,
         const parameter_type& value,
         const displacement_type q0 = displacement_type())
  : scax_module(nm), bp("bp"), value_(value), q_(q0) {}

  // Callback to return the number of unconverged variables.
  virtual int unconverged() const { return q_.unconverged(); }
protected:
  // Callback to set causality constraints.
  virtual void set_attributes() {
```

```
    bp.set_causality_constraint(PREFERRED_FLOW_IN);
  }
  // Callback to implement causality constraints between bond ports.
  virtual bool propagate_causality() { /* Not needed for 1-ports. */ }
  // Callback to register the processing methods according to causality.
  virtual void after_causality_assignment() {
    if (bp.get_causality() == EFFORT_IN) {
      SCAX_BG_METHOD(process_linearly_e_to_f);
      method_io << bp.effort >> bp.flow;
    } else if (bp.get_causality() == FLOW_IN) {
      SCAX_BG_METHOD(integrate_f_to_q);
      method_io << bp.flow;
      SCAX_BG_METHOD(process_linearly_q_to_e);
      method_io >> bp.effort;
    } else { SC_REPORT_ERROR("/SCAX/BG", "No causality."); }
  }
  // Callbacks to accept or reject the solution for the internal state variables.
  virtual void update() { q_.update(); }
  virtual void reset() { q_.reset(); }
  // Processing member functions.
  void integrate_f_to_q() {
    // Use trapezoidal method to improve the initial Euler guess of q_.
    time_type h = scax_util::sca_time_cast<time_type>(get_timestep());
    q_ = q_.read_last() + 0.5 * h * (bp.read_last_flow() + bp.read_flow());
  }
  void process_linearly_q_to_e() {
    if (!get_iteration()) { // For 1st iteration, estimate q_ with Euler step.
      time_type h = scax_util::sca_time_cast<time_type>(get_timestep());
      q_ = q_.read_last() + h * bp.read_last_flow(); }
    bp.write_effort(q_.read() / value_);
  }
  void process_linearly_e_to_f() { /* ... */ }
private:
  parameter_type value_; // Linear parameter value.
  scax_variable<displacement_type> q_; // Displacement state variable.
}; // class scax_bg::scax_C<Domain>
```

Die Instanziierung und Verschaltung der BG-Module erfolgt wie die der TDF- und DE-Module. Allerdings verfügt jeder Bond `b`, im Gegensatz zu gerichteten Signalen, über einen Kopf `b.head` und einen Schwanz `b.tail`, die getrennt jeweils an nur einen Bondanschluss gebunden werden müssen. Als Beispiel zeigt Abb. 2, wie ein einfacher Bondgraph bestehend aus einer Potentialquelle S_e, einem Bond und einem Widerstand R mit unserem BG MoC beschrieben, elaboriert und simuliert wird.

Die *Sichtschicht* handhabt die MoC-spezifischen Schritte während der Elaboration. Sie wird von SystemC AMS nach der Anbindung der Anschlüsse an die Kanäle aufgerufen. Sodann gruppiert sie alle BG-Module, die durch gerichtete Signale oder Bonds verbunden sind, in Cluster. Für jeden Cluster erzeugt die Sichtschicht einen BG-Löser und ordnet ihm die entsprechenden BG-Module, Signale, Bonds und Konverteranschlüsse zu. Abschließend bestimmt sie für jeden BG-Löser den Synchronisierungszeitschritt mit den anderen MoCs.

Die *Löserschicht* führt die BG-Löser nach der Elaboration aus. Während der Initialisierung weißt der BG-Löser allen Bonds eine Kausalität zu, welche die in `set_attributes()` definierten Randbedingungen erfüllt. In Abhängigkeit von der Kausalität werden dann die Verarbeitungsfunktionen der BG-Module registriert und ein statischer Ablaufplan für ihren Aufruf bestimmt (Abb. 2). Während der transienten Simulation wird die *Cluster-Verarbeitungsfunktion* eines jeden BG-Lösers für jeden Synchronisationszeitpunkt aktiviert. Diese schreibt die Lösung vom letzten Synchronisationszeitpunkt zum Aktuellen fort, indem sie variable Zeitschritte durchführt. Für jeden Zeitschritt werden die registrierten Verarbeitungsfunktionen der BG-Module gemäß dem Ablaufplan ausgeführt und dann die Lösung auf Konvergenz geprüft. Bei Nichtkonvergenz werden weitere Fixpunktiterationen durchgeführt, bis die Lösung akzeptiert werden kann. Sind zu viele Iterationen nötig, wird die Lösung verworfen und ein kleinerer Zeitschritt versucht.

Die *Synchronisierungsschicht* koordiniert die parallele Ausführung der verschiedenen zeitkontinuierlichen Berechnungsmodelle (TDF, LSF, ELN, BG) und ermöglicht ihre Kommunikation untereinander sowie mit dem DE-Simulationskernel von SystemC.

IV. Anwendungsbeispiel

Zur Demonstration des Zusammenspiels der DE-, TDF- und BG-Berechnungsmodelle wählten wir ein elektromechanisches Vibrationssensor mit digitalem Frontend aus (Abb. 3). Sein mechanischer Resonator, bestehend aus Masse m, Feder k und Dämpfer d, wird über den Referenzrahmen durch eine externe Vibrationsgeschwindigkeit $v_{\mathrm{vib}}(t)$ angeregt. Der elektromechanische Transducer wird als C-Feld-Zweitor modelliert, welcher die relative Bewegung der Masse in einen Verschiebungsstrom umwandelt. Dieser wird über einen Shunt-Widerstand R_1 durch ein, als Potentialdetektor D_e repräsentiertes, Voltmeter gemessen. Dessen Signal wird von einem programmierbaren Verstärker (PGA) vorverstärkt, bevor es von einem A/D-Wandler (ADC) digitalisiert wird. Der folgende Block mittelt die absolute Amplitude von n_{s} Samples. Das Ergebnis nutzt der Verstärkungsregler um den Verstärkungsfaktor des PGAs so anzupassen, dass der Dynamikbereich des ADCs möglichst gut für die Digitalisierung des Vibrationssignals ausgenutzt wird.

Alle Komponenten aus Abb. 3 werden in eine SystemC-Testbench instanziiert und mit BG-Bonds, BG-, TDF- und DE-Signalen verbunden. Listing 2 zeigt, als Ausschnitt aus der strukturellen Beschreibung, den Vibrationssensor. Alle einfachen Bongraphgrundelemente werden bereits durch die SCAX-Bibliothek bereitgestellt. Das C-Feld und der PGA sind als benutzerdefinierte BG-Module

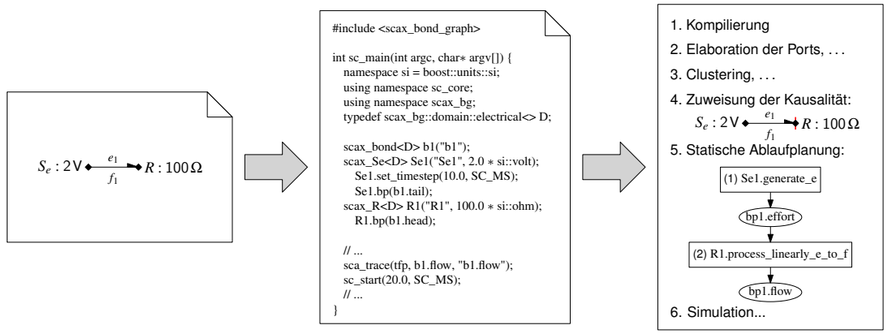

Abb. 2: Übersicht über die Schritte zur Elaboration und Simulation eines Modells durch den SCAX BG MoC.

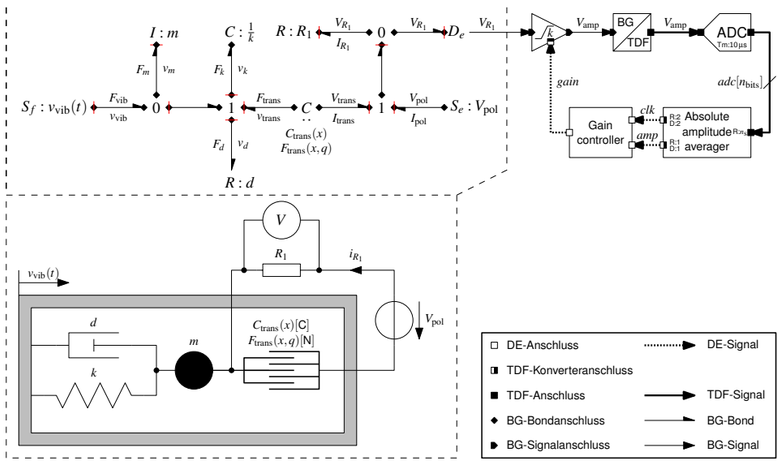

Abb. 3: Schema des Vibrationssensormodells bestehend aus BG-, TDF- und DE-Modulen.

realisiert (Abschnitt III). Jedes der BG-Module wird auf die physikalische Domäne parametriert und numerische Konstanten mit ihren Maßeinheiten annotiert, so dass der Compiler die Kohärenz der Beschreibung überprüfen kann. Der ADC und Amplitudenmittelwertbilder sind als TDF-Module realisiert und nutzen dabei die Multirate-Fähigkeiten von SystemC AMS. Der Verstärkungsregler ist als DE-Modul realisiert.

Listing 2: Strukturbeschreibung des Vibrationssensors.

```
// Convenience typedefs for quantity types.
typedef quantity<si::electric_potential> voltage_type;
typedef quantity<si::electric_charge> charge_type;
typedef quantity<si::force> force_type;
// ...
// Bonds and signals.
scax_bond<translational<> > b_m1("b_m1"), /* ... */, b_m6("b_m6");
scax_bond<electrical<> > b_e1("b_e1"), b_e2("b_e2"), /* ... */, b_e5("b_e5");
// Mechanical vibration source.
scax_Sf<translational<> >
Sf_m1("Sf_m1", /* Functor defining waveform... */);
  Sf_m1.bp(b_m1.tail);
// Mechanical resonator.
scax_J0<translational<> > J0_m1("J0_m1");
  J0_m1.bp(b_m1.head); J0_m1.bp(b_m2.tail); J0_m1.bp(b_m3.tail);
scax_I<translational<> > I_m1("I_m1", 1.0e-3 * si::kilogram);
  I_m1.bp(b_m2.head);
scax_J1<translational<> > J1_m1("J1_m1");
  J1_m1.bp(b_m3.head); J1_m1.bp(b_m4.tail); /* ... */; J1_m1.bp(b_m6.tail);
scax_C<translational<> > C_m1("C_m1", 1.0 / 40.0 * si::newton / si::meter);
  C_m1.bp(b_m4.head);
scax_R<translational<> >
  R_m1("R_m1", 0.1 * si::newton * si::second / si::meter);
  R_m1.bp(b_m5.head);
// Transducer capacitance formula v_trans(x, q) using Boost.Lambda.
function<voltage_type (displacement_type, charge_type)>
  v_trans_fn = _2 / (C_trans_0 * (1.0 + (_1/overlap)));
// Transducer force formula F_trans(x, q) using Boost.Lambda.
```

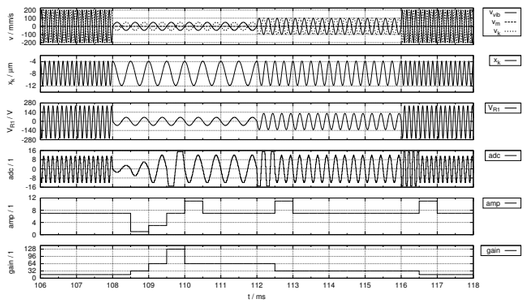

Abb. 4: Simulation des Vibrationssensormodells.

```
function<force_type (displacement_type, charge_type)>
   F_trans_fn = (_2*_2*overlap) / (2.0*C_trans_0*(overlap+_1)*(overlap+_1));
// Transducer.
C2_transducer<translational<>, electrical<> >
   C2_me1("C2_me1", F_trans_fn, v_trans_fn);
   C2_me1.bp1(b_m6.head); C2_me1.bp2(b_e3.tail);
// Electrical sensing circuit...
```

Abb. 4 zeigt die Simulationsergebnisse. Das ausgewählte sinusförmige Vibrationstestsignal v_{vib} ändert periodisch seine Frequenz und Amplitude, welche eine Grundfrequenz von 2 kHz und die erste und zweite harmonische Schwingung repräsentieren. Die Amplituden wurden so gewählt, dass die Verschiebungsamplitude konstant ist. An der Geschwindigkeit v_m sieht man die Trägheit der Masse gegenüber dieser Anregung. Dadurch wird die Anregung vollständig von der Feder, dem Dämpfer und dem Transducer aufgenommen, was durch die Geschwindigkeit v_k und die Verschiebung x_k ersichtlich ist. Der Verschiebungsstrom in dem elektrostatischen Transducer is abhängig von v_k und wird als v_{R_1} über den Shunt-Widerstand gemessen. Die Spannung wird vom PGA um den Faktor *gain* verstärkt und vom ADC in das TDF-Signal *adc* digitalisiert. Die Sättigung des Ausgangssignals ist deutlich sichtbar für zu großes *gain*. Das DE-Signal *amp* zeigt die gemittelte Amplitude, die zur Regelung von *gain* genutzt wird.

V. Zusammenfassung und Ausblick

Dieser Beitrag präsentiert unsere Erweiterung von SystemC AMS zur effizienten Beschreibung und Simulation nichtlinearer multiphysikalischer Systeme mit Hilfe der Bondgraph- und Blockdiagrammformalismen. Das dazu neu entwickelte Bondgraphberechnungsmodell (BG MoC) integriert sich nahtlos in die OSCI/Fraunhofer-Implementierung von SystemC AMS. Mit dem Vibrationssensor-Anwendungsbeispiel demonstrierten wir, wie die BG-, TDF- und DE-Berechnungsmodelle eng miteinander in einem Systemmodell interagieren können. Die Ergebnisse unserer Arbeit zeigen, dass es möglich ist sonst gegensätzliche Anforderungen in der Modellierung zu verschmelzen: *Generizität*, um die Wiederverwendung eines Modells zu ermöglichen, und *Präzision in der Spezifikation*, um die Verifikation zu erleichtern.

Die weitere Arbeit am BG MoC zielt auf bessere Algorithmen zur Lösung algebraischer Schleifen in Bondgraph-/Blockdiagrammmodellen, damit sie zuverlässiger konvergieren. Eine weitere Aufgabe ist eine bessere Unterstützung für die Modellierung der Interaktion zwischen den zeitkontinuierlichen und zeitdiskreten Domänen. Sie beinhaltet das Umschalten von Energie- und Signalflüssen in Bongraphen und Blockdiagrammen aufgrund von digitalen Ereignissen sowie das Erzeugen digitaler Ereignisse aufgrund der Überschreitung von Grenzwerten durch zeitkontinuierliche Signale.

Danksagung

Torsten Mähne entwickelte die SCAX-Bibliothek während seiner Promotion an der EPFL im LSM. Die Autoren bedanken sich bei der Hasler Stiftung für die Finanzierung dieser Arbeit unter der Projektnummer 2161.

Literatur

[1] IEEE Computer Society, *1666-2005 IEEE standard SystemC language reference manual*, Englisch, 1666-2005, IEEE, 31. März 2006, ISBN: 0-7381-4871-7.

[2] OSCI AMSWG, *Standard SystemC AMS extensions language reference manual*, Englisch, Version 1.0, März 2010. Adresse: http://www.systemc.org/.

[3] T. Uhle und K. Einwich, „A SystemC AMS extension for the simulation of non-linear circuits“, in *Proc. of the 23 IEEE International SoC Conference (SOCC) 2010*, (Las Vegas, NV, USA, Sep. 2010), S. 193–198, ISBN: 978-1-4244-6683-2.

[4] D. C. Karnopp, D. L. Margolis und R. C. Rosenberg, *System Dynamics: Modeling and Simulation of Mechatronic Systems*, Englisch, 4. Aufl. New York, USA: Wiley, Jan. 2006, ISBN: 978-0471709657.

[5] T. Mähne, „Efficient modelling and simulation methodology for the design of heterogeneous mixed-signal systems on chip“, Diss., EPF Lausanne, CH, Apr. 2011. DOI: 10.5075/epfl-thesis-4993.

[6] T. Maehne und A. Vachoux, „Supporting dimensional analysis in SystemC-AMS“, in *Proc. of the 2009 IEEE International Behavioral Modeling and Simulation (BMAS) Workshop*, (San Jose CA, USA, Sep. 2009), S. 108–113, ISBN: 978-1-4244-5359-7.

Eine Erweiterung des linearen Lösers in SystemC AMS für zeitlich veränderliche Module

Christiane Reuther, Karsten Einwich
Fraunhofer-Institut für Integrierte Schaltungen
Institutsteil Entwurfsautomatisierung
Zeunerstr. 38, 01069 Dresden, Germany

***Zusammenfassung*—Dieser Artikel stellt eine Möglichkeit vor, die Simulationsgeschwindigkeit des linearen Lösers von SystemC AMS für bestimmte Modellklassen signifikant zu erhöhen. Dies wird mit Hilfe der Woodbury-Formel erreicht, welche es erlaubt, die zeitaufwendige Refaktorisierung für den Fall von wenigen Änderungen in der Koeffizientenmatrix zu ersetzen. Der Artikel schließt mit einer Demonstration des Gewinns der Simulationsgeschwindigkeit an relevanten Industriebeispielen.**

I. Einleitung

SystemC [1] ist eine standardisierte Beschreibungssprache für große digitale Hardware/Software-Systeme. Die Sprache basiert auf der objekt-orientierten Programmiersprache C++. SystemC bietet verschiedene Berechnungsmodelle, die eine effiziente Entwicklung von Beschreibungen auf hohen Abstraktionsebenen und schnelle Simulationen für abstrakte Modelle erlauben. SystemC AMS erweitert die Beschreibungssprache SystemC um die Möglichkeit, analoge oder mixed-signal-Systeme zu modellieren, siehe dazu u.a. Vachoux et al. [2] und Grimm et al. [3].

SystemC AMS weist zusätzlich drei Berechnungsmodelle auf, um auf unterschiedlichen Abstraktionsebenen bezüglich Verhalten, Struktur, Kommunikation und Zeit/Frequenz Systeme zu beschreiben. Diese Modelle sind ELN (electrical linear network), LSF (linear signal flow) und TDF (timed data flow). Die Modellbeschreibungen für die ersten beiden Berechnungsmodelle stellen ein lineares Algebro-Differentialgleichungssystem (DAE, differential algebraic equations) dar. Für die Simulation ist dafür ein einfacher linearer Löser erforderlich. Eine Übersicht über die drei genannten Verhaltensmodelle und ihre Charakteristiken findet man in Barnasconi [4]. Aufgrund angepasster numerischer Algorithmen für die verschiedenen Berechnungsmodelle ist es möglich, mit SystemC AMS hohe Simulationsgeschwindigkeiten zu erreichen.

Im Blickfeld dieses Artikels stehen ELN- und LSF-Module, deren charakteristische Eigenschaften sich in jedem Zeitschritt ändern können. Solche Modifikationen treten beispielsweise in ELN Netzwerken auf, wenn sich Widerstands-, Kapazitäts-, Induktivitätswerte bzw. die Stellung von Schaltern zeitabhängig ändern. Bei LSF betrifft dies variable Verstärker-, Multiplexer- und Demultiplexer-Module. SystemC AMS definiert diese als Module, die von einem TDF- bzw. einem DE-Signal (SystemC discrete event) gesteuert werden. Werden solche Module in Regelschleifen eingesetzt bzw. verändert sich das steuernde Signal stetig, ändert sich der Wert des Moduls in fast jedem Zeitschritt. Dies tritt sehr häufig in Sensormodellen auf, z.B. bei einem Drucksensor, bei welchem das Sensorelement ein druckabhängiger Kondensator ist. Ein anderes Beispiel ist der Drehratensensor, bei welchem die Drehrate als Koeffizient in eine lineare Übertragungsfunktion multipliziert wird.

Um elektrische lineare Netzwerke oder lineare Signalflüsse zu simulieren, wird mithilfe des linearen Lösers in jedem Zeitschritt ein lineares Gleichungssystem (LGS) gelöst. Falls Änderungen der variablen Module auftreten, ändern sich die zugehörigen Koeffizienten in der Matrix des LGS. Dies bewirkt, dass der lineare Löser eine erneute Faktorisierung mit Pivotsuche der Koeffizientenmatrix veranlasst, um das LGS zu lösen. Vor allem in der Simulation großer Systeme hat die Neufaktorisierung negative Auswirkungen auf die Gesamtsimulationsgeschwindigkeit.

Mittels Anwendung der Woodbury-Formel ist es möglich, das veränderte LGS ohne erneute Faktorisierung zu lösen. Gerade bei großen Systemen mit einer geringen Anzahl von variablen Modulen bringt die Anwendung der Woodbury-Formel auf das LGS eine verbesserte Simulationszeit im Gegensatz zur Neufaktorisierung der Koeffizientenmatrix des LGS. Die Idee, die Woodbury-Formel für Modifikationen in ELN zu verwenden, ist nicht neu, siehe z.B. Edelmann [5]. Inhalt dieses Artikels

ist die Beschreibung der Umsetzung der Woodbury-Formel im linearen Löser von SystemC AMS.

II. Die Woodbury-Formel

In Hager [6] ist eine umfassende Einführung zur Geschichte und zu einzelnen Anwendungen der Woodbury-Formel formuliert. Wir werden in diesem Abschnitt auf einige Aspekte von Hager eingehen. Im Mittelpunkt steht das folgende Resultat.

Theorem 1 (Woodbury-Formel, Woodbury [7])**.** *Die Matrizen A und $I + WA^{-1}V$ seien quadratische nichtsinguläre Matrizen, die Matrizen V und W seien rechteckige Matrizen und I die Einheitsmatrix, so dass die auftretenden Matrixmultiplikationen möglich sind. Dann ist auch $A + VW$ nichtsingulär und es gilt*

$$\begin{aligned}&(A+VW)^{-1}\\ =\;& A^{-1} - A^{-1}V\left(I + WA^{-1}V\right)^{-1}WA^{-1}.\end{aligned}$$

Den Beweis der Woodbury-Formel findet man u.a. in Edelmann [5].

Wir nehmen an, dass A eine $n \times n$ Matrix ist. Außerdem seien V eine $n \times k$ Matrix mit den Spaltenvektoren $v_1, ..., v_k$ und W eine $k \times n$ Matrix mit den Zeilenvektoren $w_1, ..., w_k$. Das Matrix-Produkt VW ist dann gerade

$$VW = \sum_{i=1}^{k} v_i w_i. \tag{1}$$

Dies impliziert, dass $I + WA^{-1}V$ eine $k \times k$ Matrix ist. Jeder Summand in (1) kennzeichnet eine Rang-1-Änderung der Matrix A. Deshalb stellt die Woodbury-Formel in Theorem 1 eine Gleichung bereit, die Inverse einer Matrix mit k Rang-1-Änderungen zu ermitteln.

Sei x_0 der Lösungsvektor des LGS $Ax_0 = r$, wobei r ein n-dimensionaler Vektor ist. Dann ergibt sich der Lösungsvektor x des modifizierten Systems

$$(A + VW)x = r \tag{2}$$

mittels der Woodbury-Formel:

$$\begin{aligned}x &= (A+VW)^{-1}r\\ &= x_0 - A^{-1}V\left(I + WA^{-1}V\right)^{-1}Wx_0.\end{aligned}$$

Die Anwendung der Woodbury-Formel ist nur dann sinnvoll, falls die Ordnung k der Modifikation hinreichend klein ist im Vergleich zur Dimension n. Denn anstelle einer Faktorisierung der $n \times n$ Matrix $A + VW$ müssen eine Faktorisierung der $k \times k$ Matrix $I + WA^{-1}V$ sowie zusätzliche Additionen und Multiplikationen durchgeführt werden. Ob die Woodbury-Formel tatsächlich eine Möglichkeit darstellt, die Inverse der gestörten Matrix mit weniger Aufwand zu berechnen, ist auch abhängig von der (Dünn-)Besetztheit der Matrix A und der Struktur der Vektoren v_i, w_i, $i = 1, ..., k$.

A. Berechnung mittels Woodbury-Formel

Wir setzen voraus, dass die Einsetzung der Woodbury-Formel zur Lösung von (2) geringeren Aufwand erfordert im Gegensatz zur Neufaktorisierung. Aufgrund der in Abschnitt III beschriebenen Struktur der Vektoren v_i und w_i, erscheint es zweckmäßig den Berechnungsschritten von Duff et al. [8] zu folgen. Die natürliche Implementierung erzeugt die Lösung x des Gleichungssystems (2) wie folgt:

Schritt 1:	Löse $x_0 = A^{-1}r$.
Schritt 2:	Löse $X_1 = A^{-1}V$.
Schritt 3:	Berechne $X_2 = I + WX_1$.
Schritt 4:	Berechne $x_3 = Wx_0$.
Schritt 5:	Löse $x_4 = X_2^{-1}x_3$.
Schritt 6:	Berechne $x = x_0 - X_1x_4$.

Schritt 1, Schritt 2 und Schritt 5 erfordern den größten rechnerischen Aufwand in der Implementierung der Woodbury-Formel. Da die Matrix W, wie unten angegeben, von einfacher Struktur ist, benötigen die Schritte 3 und 4 nur eine geringe Anzahl von Berechnungen.

B. Numerische Stabilität

Es ist von der Wahl der Matrizen V und W abhängig, ob die Woodbury-Formel ein brauchbares Ergebnis liefert. Im schlechtesten Fall ist die Matrix X_2 in Schritt 3 schlechtkonditioniert, obwohl sowohl A als auch $A + VW$ gut konditionierte Matrizen sind. Dies kann dazu führen, dass numerische Fehler bei der Lösung von (2) bei Verwendung der Woodbury-Formel auftreten, da das numerische Problem in Schritt 5 instabil ist aufgrund der schlechtkonditionierten Matrix X_2.

Ob ein LGS numerisch stabil ist, ist abhängig von der Konditionszahl κ der zugehörigen Matrix. Diese ist für eine Matrix A gegeben durch $\kappa(A) = \|A\|\|A^{-1}\|$ bezüglich einer geeigneten Matrixnorm, siehe hierzu u.a. Demmel [9]. Je größer $\kappa(A)$ ist, desto schlechter konditioniert ist das numerische Problem.

In [6], [10] zeigen Hager und Yip, dass unter gegebenen Voraussetzungen an die Matrizen V und W sicher gestellt werden kann, dass

$$\kappa(X_2) \leq \kappa(A)\kappa(A + VW). \tag{3}$$

Das heißt, falls die Matrizen A und $A + VW$ von beschränkter Kondition sind, dann gilt das auch für die Matrix X_2. Gemäß Yip ist die Ungleichung (3) erfüllt, falls V oder W aus linear unabhängigen Spalten der Einheitsmatrix bestehen.

In der SystemC AMS Implementierung der Woodbury-Formel (wie weiter unten gezeigt) ist in der Matrix W jede Spalte gerade ein Einheitsvektor oder die Differenz zweier Einheitsvektoren. Falls W vollen Spaltenrang besitzt, gilt folgendes:

$$\kappa_2(X_2) \leq C_W\kappa_2(A)\kappa_2(A + VW),$$

wobei κ_2 die Konditionszahl bzgl. der Spektralnorm und C_W eine beschränkte Konstante darstellen.

III. Die Woodbury-Formel in SystemC AMS

Die Woodbury-Formel wurde in SystemC AMS für die Berechnungsmodelle ELN und LSF eingeführt. Mit Hilfe von ELN werden elektrische lineare Netzwerke modelliert. In SystemC AMS liegt dem die Modifizierte Knotenspannungsanalyse zugrunde, die auf den Kirchhoffschen Gesetzen basiert, siehe dazu auch Chua et al. [11]. Das Ziel der Modifizierten Knotenspannungsanalyse ist es, ein DAE mit einer möglichst geringen Anzahl von Gleichungen zu erstellen. Die Unbekannten des Gleichungssystems sind die Knotenspannungen und ausgewählte Zweigströme des Netzwerks. Das Berechnungsmodell LSF ermöglicht dagegen die Modellierung von gerichteten reellwertigen Signalflüssen, die als lineare algebraische Gleichungen dargestellt werden. Mit LSF können nicht-konservative Systeme beschrieben werden.

Zur Erstellung von ELN- oder LSF-Modellen werden DAE der Form

$$C\dot{x} + Bx = q \tag{4}$$

betrachtet. Dabei stellen C und B Matrizen sowie q und x Vektoren dar, die von der Zeit t abhängen. Der Lösungsvektor x besteht in ELN aus den gesuchten Spannungen und Strömen und in LSF aus reellwertigen Signalen.

Zur Lösung von (4) werden im linearen Löser von SystemC AMS das implizierte Euler-Verfahren sowie das Trapezregelverfahren genutzt, indem $\dot{x}$ mit der entsprechenden Verfahrensformel ersetzt wird, siehe dazu z.B. Ascher und Petzold [12]. Demzufolge wird in jedem Zeitpunkt t_l, $l = 0, ..., N$, $N \in \mathbb{N}$, das LGS

$$A_l x_l = r_l \tag{5}$$

gelöst. Dabei bezeichnet x_l die approximierte Lösung für x in (4) im Zeitschritt t_l. Die Matrix A_l ist gegeben durch

$$A_l = \frac{1}{h}C + B_l \quad \text{bzw.} \quad A_l = \frac{2}{h}C + B_l \tag{6}$$

für die implizite Euler-Formel bzw. die Trapezformel mit Schrittweite h. Das heißt, dass $h = t_l - t_{l-1}$. Die Matrix C ist konstant und B_l bezeichnet B zum Zeitpunkt t_l. Die rechte Seite r_l in (5) ist abhängig von h, C, $q(t_l)$, x_{l-1} und x_{l-2}, letzteres gilt nur für die Trapezregel.

A. Zeitlich veränderliche Module in SystemC AMS

Die zeitlich veränderlichen ELN-Module sind die durch TDF- oder DE-Eingangssignale gesteuerten Widerstände, Spulen oder Kondensatoren sowie schaltbare Widerstände. Diese werden in SystemC AMS durch die Sprachelemente

```
sca_eln::sca_tdf_r sca_eln::sca_de_r
sca_eln::sca_tdf_l sca_eln::sca_de_l
sca_eln::sca_tdf_c sca_eln::sca_de_c
sca_eln::sca_tdf_rswitch
sca_eln::sca_de_rswitch
```

definiert. Für LSF-Module sind Multiplexer, Demultiplexer und Verstärker relevant, die von TDF- bzw. DE-Signalen gesteuert werden. Die zugehörigen SystemC AMS Sprachelemente sind

```
sca_lsf::sca_tdf_mux
sca_lsf::sca_de_mux
sca_lsf::sca_tdf_demux
sca_lsf::sca_de_demux
sca_lsf::sca_tdf_gain
sca_lsf::sca_de_gain
```

Eine eingehende Beschreibung der Module findet man im SystemC AMS 1.0 Standard [13].

Eine Änderung eines TDF- oder DE-Eingangssignals zum Zeitpunkt t_l bezüglich der oben genannten variablen Module bewirkt jeweils eine Rang-1-Modifikation der Matrix B_l in (6) und damit der Matrix A_l in (5). Vor Hinzufügen der Woodbury-Formel in SystemC AMS wurde bei jeder Änderung eine Refaktorisierung der Matrix A_l mittels Gauß-Elimination mit Markowitz-Pivotsuche angestoßen, um das System (5) zu lösen.

B. Konkrete Darstellung der Modifikationen

Wir nehmen jetzt an, dass eine Faktorisierung der Matrix A vorliegt und A_l aus einer Rang-1-Änderung aus A hervorgeht. Zum Beschreiben einer solchen Modifikation nutzen wir die in Abschnitt II eingeführten Bezeichnungen. Es sei also

$$A_l = A + vw,$$

wobei die Vektoren v und w wie in Gleichung (1) die Modifikation darstellen. Für die oben genannten ELN- und LSF-Module kann die Rang-1-Änderung der Matrix A_l wie folgt beschrieben werden. Dabei seien e_i, $i = 1, ..., n$, die i-ten Einheitsvektoren. Es treten vier Fälle auf.

Fall 1: Ein Eintrag der Matrix A wird in Zeile i und Spalte j um den Wert a modifiziert. Wir wählen $v = a \cdot e_i$ und $w^T = e_j$. Dann gilt

$$vw = i \begin{pmatrix} a \end{pmatrix} \begin{pmatrix} j \\ 1 \end{pmatrix} = \begin{matrix} j \\ i \end{matrix} \begin{pmatrix} a \end{pmatrix}.$$

Dies wird beispielsweise für `sca_eln::sca_tdf_r` in der Zweig-Impedanz-Darstellung verwendet.

Fall 2: Zwei Einträge der Matrix A werden in Zeile i und den Spalten j_0, j_1 um den Wert $\pm a$ verändert. Wir schreiben $v = a \cdot e_i$ und $w^T = e_{j_0} - e_{j_1}$, also

$$vw = i \begin{pmatrix} a \end{pmatrix} \begin{pmatrix} j_0 & j_1 \\ 1 & -1 \end{pmatrix} = \begin{matrix} j_0 & j_1 \\ i \end{matrix} \begin{pmatrix} a & -a \end{pmatrix}.$$

Beispielsweise wird diese Wahl von v, w in der Beschreibung der Änderung des Kondensators `sca_eln::sca_tdf_c` genutzt.

Fall 3: In der Spalte j der Matrix A werden zwei Einträge in den Zeilen i_0 und i_1 um den Wert $\pm a$ modifiziert. Das wird ausgedrückt mittels $v = a \cdot (e_{i_0} - e_{i_1})$ und $w^T = e_j$. Es gilt also

$$vw = \begin{matrix} i_0 \\ i_1 \end{matrix} \begin{pmatrix} a \\ -a \end{pmatrix} \begin{pmatrix} j \\ 1 \end{pmatrix} = \begin{matrix} j \\ i_0 \\ i_1 \end{matrix} \begin{pmatrix} a \\ -a \end{pmatrix}.$$

Diese Darstellung findet z.B. Anwendung in der Beschreibung der Rang-1-Modifikation durch den Demultiplexer `sca_lsf::sca_de_demux`.

Fall 4: Es werden vier Einträge der Matrix A in den Zeilen i, j und den Spalten i, j um den Wert $\pm a$ wie folgt modifiziert. Wir setzen $v = a \cdot (e_i - e_j)$ und $w^T = e_i - e_j$ und damit

$$vw = \begin{matrix} i \\ j \end{matrix} \begin{pmatrix} a \\ -a \end{pmatrix} \begin{pmatrix} i & j \\ 1 & -1 \end{pmatrix} = \begin{matrix} i & j \\ j \end{matrix} \begin{pmatrix} a & -a \\ -a & a \end{pmatrix}.$$

Ein Beispiel zu Fall 4 ist der schaltbare Widerstand `sca_eln::sca_de_rswitch` in der Knoten-Admittanz-Darstellung.

C. Einbindung der Woodbury-Formel in den linearen Löser von SystemC AMS

Die Einbindung der Woodbury-Formel geschieht in drei Ebenen von SystemC AMS. Die erste Ebene betrifft die Beschreibung der Module durch ihre Matrixeinträge, die durch äußere Signale gesteuert werden. Im Zeitpunkt t_0 werden die Matrix-Einträge aller Module gesetzt. Falls zu einem späteren Zeitpunkt in einem variablen Modul eine Rang-1-Modifikation der Koeffizientenmatrix auftritt, werden nicht wie bisher die entsprechenden Matrixeinträge neu gesetzt, sondern der Wert und die Art der Modifikation des Moduls (siehe dazu den vorherigen Unterabschnitt) werden an den linearen Löser übergeben.

Im eigentlichen linearen Löser, der zweiten Ebene, wird im Zeitpunkt t_0 die Koeffizientenmatrix des aufgestellten LGS faktorisiert. Falls zu einem späteren Zeitpunkt mindestens ein Modul eine Änderung der Koeffizientenmatrix bewirkt, wird die Anzahl der modifizierten Module und damit der Rang k der Änderungsmatrix dem Löser mitgeteilt. Im linearen Löser wird in Abhängigkeit des Modifikationsranges k und der Dimension n des LGS festgelegt, wie das LGS gelöst werden soll. Falls k hinreichend klein ist, wird die Ausführung der Woodbury-Formel angestoßen. Falls dies nicht gilt, werden die betreffenden Matrixeinträge neu gesetzt und die Refaktorisierung der Koeffizientenmatrix wird durchgeführt, um dann das LGS zu lösen.

Die Implementierung der Woodbury-Formel geschieht in der dritten Ebene. Es werden alle k Rang-1-Modifikationen mit Wert und Art (entspricht den Matrizen V, W) sowie die faktorisierte Koeffizientenmatrix A und die rechte Seite r übergeben. Die Methode veranlasst die Lösung des aktuellen LGS basierend auf der Woodbury-Formel, wie in Abschnitt II dargestellt.

IV. Beispiele

A. Parallel-Schwingkreis mit variabler Kapazität

An einem Beispiel demonstrieren wir zunächst die Einsetzung der Woodbury-Formel im elektrischen linearen Netzwerk. In Abb. 1 ist das elektrische Netzwerk eines Parallel-Schwingkreises als ELN-Modell angegeben, wobei sich die Kapazität des Kondensators in der Zeit ändert. Dies könnte z.B. ein einfaches Modell für einen Frequenzmodulator darstellen. Die konstanten Parameter des Schwingkreises sind die Induktivität $L = 50\text{mH}$ der Spule, der Widerstand der Spule $R_C = 1\Omega$ und die Dämpfung der Schwingung durch den Widerstand $R = 1.25\text{k}\Omega$. In SystemC AMS wird die veränderliche Kapazität mittels eines Kondensators modelliert,

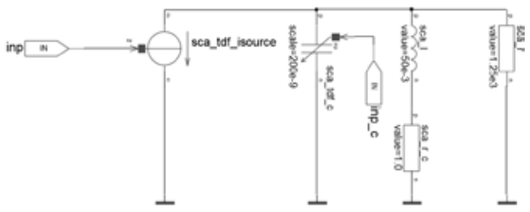

Abb. 1. ELN-Modell

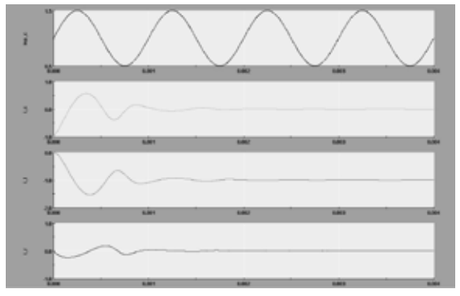

Abb. 2. Simulationsergebnisse

der durch ein TDF-Signal kontrolliert wird. In Abb. 2 sind die Simulationsergebnisse für den Strom des Kondensators, der Spule und des Widerstandes in den unteren drei Graphiken dargestellt. Mit den gegebenen Parametern wird eine gedämpfte Schwingung simuliert. Der Schwingkreis wird durch eine konstante Stromquelle angeregt. Die Kapazität des Kondensators ist eine sinusförmige Schwingung (erste Graphik in Abb. 2) mit dem Offset-Wert von 200nF.

Für die numerische Berechnung wird ein LGS mit 5 Gleichungen aufgestellt. Da sich die Kapazität in jedem Zeitschritt ändert, werden in der Matrix B des LGS zwei Matrixeinträge, wie in Fall 2 in Abschnitt III beschrieben, modifiziert. Für die Simulation wurde ein Linux-System mit 2.0GHz CPU verwendet. Tab. 1 zeigt die Simulationszeiten für die Simulation von 10 sec Echtzeit.

Tab.1. Simulationszeiten Oszillator

C konst.	C sinus-förmig	
	Woodbury	Refaktorisierung
150 sec	176 sec	225 sec

In der ersten Spalte wird zum Vergleich die Kapazität konstant bei $C = 200\text{nF}$ gehalten, d.h., die Koeffizientenmatrix des LGS verändert sich nicht. In der zweiten und dritten Spalte ändert sich die Kapazität sinus-förmig wie in Abb. 2 oben dargestellt. Zum einen wird bei der Simulation die Woodbury-Formel (zweite Spalte) eingesetzt, zum anderen wird wie bisher eine Refaktorisierung der Koeffizientenmatrix des LGS in jedem Zeitschritt neu angestoßen (dritte Spalte). Da das Gleichungssystem sehr klein ist, ergibt sich nur ein moderater Geschwindigkeitsgewinn.

B. Drehratensensor

In Abb. 4 ist ein Drehratensensor mittels LSF-Modellierung wiedergegeben. In das Modell gehen zwei TDF-Signale (inp_exc, inp_det) ein, welche die Sensoranregung (i.d.R. Oszillation der Sensorzungen) und die Detektion (i.d.R. Steuerspannung, welche zur Kompensation der Coriolis-Kraft erforderlich ist) darstellen. Die Ausgabe erfolgt in Form von TDF-Signalen, die vorher von LSF zu TDF transformiert werden. Das TDF-Eingabe-Signal inp_omega steuert die beiden Verstärker tdf_gain_teta und tdf_gain_psi. Dieses Signal repräsentiert die aktuelle Drehrate und geht entsprechend der Sensorfunktionalität als Multiplikation in die Übertragungsfunktion ein.

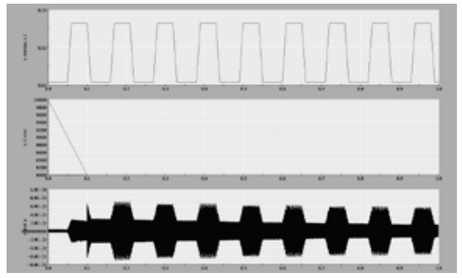

Abb. 3. Simulationsresultate

Abb. 3 zeigt ausgewählte Simulationsergebnisse. Dabei wird die Eingabe inp_det durch ein konstantes Signal modelliert. Das Signal inp_exc ist eine Sinus-Schwingung, deren Frequenz sich, wie in der zweiten Graphik von Abb. 3 dargestellt, ändert. In der obersten Graphik wird das Eingabe-Signal von inp_omega gezeigt, welches ein ständiges Schalten zwischen zwei Werten modelliert. Die dritte Graphik von Abb. 3 beschreibt das Ergebnis der Simulation für die Ausgabe dcd.

Für die Simulation wird ein LGS mit 34 Gleichungen aufgestellt. Eine Änderung des Eingabe-Signals inp_omgea bewirkt, dass zwei Matrixeinträge wie in Fall 1 in Abschnitt III modifiziert werden. Die gemessenen

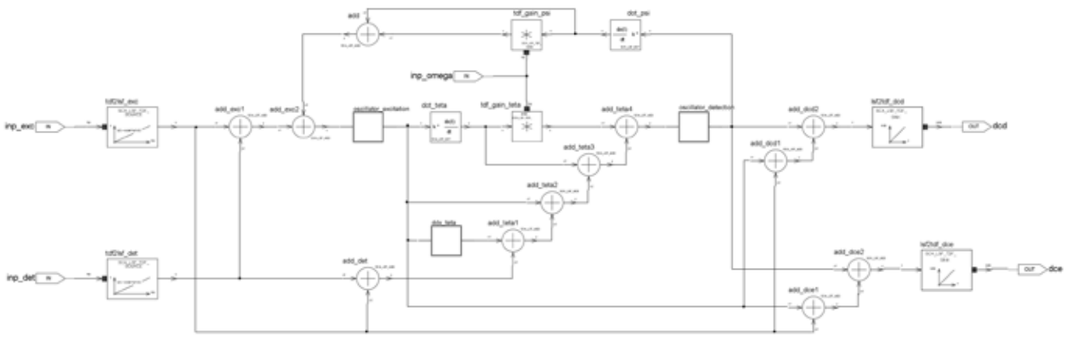

Abb. 4. LSF-Modell

Tab.2. Simulationszeiten Drehratensensor

Schalten		lineares Wachsen	
Woodbury	Refaktorisierung	Woodbury	Refaktorisierung
66 sec	135 sec	85 sec	540 sec

Zeiten für die Berechnungen für 5 sec Echtzeit mit dem oben genannten Linux-System werden in Tab. 2 präsentiert. Die beiden linken Spalten beziehen sich auf die Simulation des Modells mit Eingabe-Signalen inp_omega wie in Abb. 3 oben dargestellt. Hier ist zu bemerken, dass die Woodbury-Formel (erste Spalte) nur dann zum Einsatz kommt, falls sich das Signal zum aktuellen Zeitpunkt vom Signal zum Anfangszeitpunkt unterscheidet. Im Falle der Refaktorisierung (zweite Spalte) wird im Gegensatz dazu die Matrix-Zerlegung erneut aufgestellt, falls das aktuelle Signal auf einer Rampe liegt.

Falls die Eingabe-Werte von inp_omega im Simulationsintervall linear wachsen, wird die Koeffizientenmatrix des LGS in jedem Zeitschritt geändert. Die Auswirkungen auf die Simulationszeiten sind in den beiden rechten Spalten von Tab. 2 verdeutlicht.

V. Fazit

Der Inhalt dieses Artikels ist die Vorstellung der Woodbury-Formel und die Beschreibung zur Umsetzung der Formel in SystemC AMS. Die Anwendung der Woodbury-Formel bietet für große lineare Systeme die Möglichkeit, schnelle Simulationen durchzuführen, falls sich nur wenige Matrixeinträge der Koeffizientenmatrix des LGS häufig ändern. Dies wird anschaulich in den beiden Beispielen in Abschnitt IV dargestellt. In weiteren Diskussionen ist zu prüfen, ob die Anwendung der Woodbury-Formel auch zur Lösung von nichtlinearen DAE von Vorteil sein kann.

Literatur

[1] "IEEE 1666: Draft Standard for Standard SystemC(R) Language Reference Manual," 2011.

[2] A. Vachoux, C. Grimm, and K. Einwich, "Analog and mixed signal modelling with SystemC-AMS," in *Proceedings of the 2003 International Symposium on Circuits and Systems*, vol. 3, May 2003, pp. 914–917.

[3] C. Grimm, M. Barnasconi, A. Vachoux, and K. Einwich, "An introduction to modeling embedded analog/mixed-signal systems using SystemC AMS extensions," White paper: SystemC AMS, June 2008.

[4] M. Barnasconi, "SystemC AMS extensions: solving the need for speed," White paper: AMS modeling standard, May 2010.

[5] H. Edelmann, "Die Berücksichtigung nachträglicher Änderungen in den Impedanz- bzw. Admittanzmatrizen für besondere Klemmenpaare eines Netzes," *Electrical Engineering (Archiv für Elektrotechnik)*, vol. 47, pp. 357–369, 1963.

[6] W. W. Hager, "Updating the inverse of a matrix," *SIAM Review*, vol. 31, no. 2, pp. 221–239, 1989.

[7] M. Woodbury, "Inverting modified matrices," Statiscal research group, Princeton University, Princeton, NJ, Memorandum Rept. 42, 1950.

[8] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct methods for sparse matrices*. Oxford: Clarendon Press, 1992.

[9] J. W. Demmel, *Applied numerical linear algebra*. Philadelphia, PA: SIAM xi, 1997.

[10] E. L. Yip, "A Note on the Stability of Solving a Rank-*p* Modification of a Linear System by the Sherman–Morrison–Woodbury Formula," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 2, pp. 507–513, 1986.

[11] L. O. Chua, C. A. Desoer, and E. S. Kuh, *Linear and nonlinear circuits*. New York: McGraw-Hill, 1987.

[12] U. M. Ascher and L. R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations.* Philadelphia, PA: SIAM xvii, 1998.

[13] Open SystemC Initative, "SystemC AMS 1.0 Standard." [Online]. Available: http://www.systemc.org/downloads/standards/ams10/

Multi-Nature Modellierung und Simulation

Modellgestützte Analyse räumlich verteilter digitaler Regler in einem Schwerionen-Synchrotron

Christopher Spies, Manfred Glesner
Technische Universität Darmstadt
christopher.spies@mes.tu-darmstadt.de

Harald Klingbeil
GSI Helmholtzzentrum für Schwerionenforschung
h.klingbeil@gsi.de

***Zusammenfassung*—Die Regelung eines Teilchenbeschleunigers erfordert hohe Abtastraten und komplexe Berechnungen unter harten Echtzeitbedingungen. Die Regelkreise für die longitudinale Strahlphasenregelung bilden ein nichtlineares, zeitvariantes System mit zustandsabhängigen Totzeiten, welches einer analytischen Betrachtung nicht mehr zugänglich ist. Zur Beurteilung der Robustheit gegenüber Störungen, zur Bewertung von Entwurfsalternativen und zur gezielten Optimierung kann durch Simulationen erfolgen.**

I. Einleitung

Das *GSI-Helmholtzzentrum für Schwerionenforschung* (im Folgenden kurz als *GSI* bezeichnet) betreibt nahe Darmstadt das Schwerionensynchrotron SIS18 mit einem Umfang von $217\,\mathrm{m}$. Mit den Bauarbeiten für ein neues, größeres Synchrotron namens SIS100 mit einem Umfang von $1084\,\mathrm{m}$ wurde kürzlich begonnen; die Fertigstellung ist für das Jahr 2016 geplant [1]. Komponenten für das SIS100 werden derzeit am SIS18 erprobt.

Ein Synchrotron ist ein Teilchenbeschleuniger, in dem geladene Teilchen von magnetischen Feldern auf einer geschlossenen Flugbahn gehalten werden und wiederholt in so genannten Beschleunigungskavitäten von elektrischen Feldern beschleunigt werden. Die magnetische Flussdichte muss synchron mit der Umlauffrequenz erhöht werden, um den Bahnradius konstant zu halten — daher der Name *Synchrotron*.

Im SIS100 sollen nahezu ausschließlich digitale Regelsysteme zum Einsatz kommen und die bisher verwendeten Analogregler ablösen. Man erhofft sich davon eine geringere Anfälligkeit beispielsweise gegenüber Temperaturschwankungen und eine höhere Flexibilität, weil zusätzliche Funktionen vergleichsweise einfach hinzuprogrammiert werden können. Bis vor wenigen Jahren k digitale Regler jedoch nicht performant genug, um die erforderlichen Berechnungen in Echtzeit durchzuführen.

II. Longitudinale Strahlphasenregelung

Eine Beschleunigung durch konstante Felder ist in einem Ringbeschleuniger nicht möglich. Stattdessen werden hochfrequente Wechselfelder verwendet. Durch den Skineffekt sind Innen- und Außenseite der Kavitäten voneinander isoliert [2]. Benachbarte Kavitäten sind ebenfalls voneinander isoliert: Die Metallrohre, in denen der Strahl im Hochvakuum geführt wird, können als Wellenleiter betrachtet werden, deren Grenzfrequenz über der Frequenz der in den Kavitäten herrschenden Wechselfelder liegt. Diese können sich daher nicht in den Metallrohren ausbreiten und benachbarte Kavitäten nicht beeinflussen [2]. Die erreichbare Teilchenenergien sind nur von der maximalen magnetischen Flussdichte begrenzt. Die Verwendung von Wechselfeldern macht einen kontinuierlichen Strahl unmöglich; stattdessen zirkulieren Teilchenbündel (engl. *bunch*).

Das SIS18 verfügt über 2 Kavitäten, im SIS100 werden es voraussichtlich 14 sein. Es ist ein Mehrharmonischenbetrieb möglich, bei dem die Kavitäten bei unterschiedlichen Frequenzen betrieben werden. Die maximale Anzahl h der Teilchenbündel ergibt sich aus der Umlauffrequenz f_U und der kleinsten Frequenz f_{HF} einer Kavität. h wird als *Harmonischenzahl* bezeichnet.

$$f_{HF} = h \cdot f_U \tag{1}$$

Die Frequenz jeder Kavität muss ein ganzzahliges Vielfaches der Umlauffrequenz der Teilchen sein, und ihre Phase muss ihrer Position entlang des Beschleunigerrings entsprechen. Ist die erste Bedingung nicht erfüllt, so nehmen die Spannungen der einzelnen Kavitäten im Lauf der Zeit jede beliebige Phasenlage zueinander ein, und die zweite Bedingung ist unerfüllbar. Ist die zweite Bedingung nicht erfüllt, so werden die Teilchen in den Kavitäten unterschiedlich stark beschleunigt oder sogar abgebremst. Die *Kavitätensynchronisation* [3] hat daher die Aufgabe, eine konstante Phasenbeziehung der Kavitäten untereinander herbeizuführen.

Die Synchronisation der Frequenzen und Phasen und der magnetischen Flussdichte kann nur für ein Teilchen, das so genannte *synchrone Teilchen*, exakt gelingen. So lange keine Beschleunigung stattfindet, passiert das synchrone Teilchen jede Kavität im positiven Nulldurchgang der Beschleunigungsspannung. Der Momentanwert der Phase der Beschleunigungsspannung, welche das synchrone Teilchen „antrifft“, wird als *synchrone Phase* φ_{syn} bezeichnet. Ist $\varphi_{syn} > 0$, dann findet eine Beschleunigung statt, und das synchrone Teilchen nimmt in jedem Umlauf die Energie $\hat{u} \cdot \sin(\varphi_{syn})$ auf, wobei $\hat{u}$ die Amplitude der Beschleunigungsspannung ist. Dabei ist angenommen, dass die Zeit, die ein Teilchen zum Durchqueren der Kavität benötigt, viel kleiner als die Periodendauer der Beschleunigungsspannung ist.

In einem Bündel mit 10^{10} oder mehr Teilchen weichen deren Position und Geschwindigkeit schon durch die Coulomb-Abstoßung zwangsläufig voneinander ab. So genannte *magnetische Linsen* wirken dem Auseinanderlaufen des Teilchenbündels quer zur Flugrichtung entgegen, sollen hier aber nicht näher betrachtet werden. Ein Teilchen, welches dem synchronen Teilchen nacheilt, erfährt eine stärkere Beschleunigung und ein- und überholt jenes irgendwann. Dadurch eilt es dem synchronen Teilchen voraus, erfährt eine weniger starke Beschleunigung, fällt wieder zurück usw. Auf diese Weise oszillieren die Teilchen mit der so genannten *Synchrotronfrequenz* f_S um das synchrone Teilchen:

$$f_S = f_U \cdot \sqrt{\frac{h \cdot |\eta| \cdot q \cdot \hat{u} \cdot \cos(\varphi_{syn})}{2 \cdot \pi \cdot m \cdot v^2 \cdot \gamma}} \tag{2}$$

Dabei ist m die Ruhemasse, q die Ladung und v die Geschwindigkeit des Teilchens und η der so genannte *Phasenschlupffaktor*. Letzterer gibt an, wie stark sich die Umlaufzeit T_U bei einer Erhöhung des Impulses p eines Teilchens um Δp verändert.

$$\frac{\Delta T_U}{T_U} = \eta \cdot \frac{\Delta p}{p} \tag{3}$$

Es können jedoch auch *kohärente* Schwingungen auftreten, bei denen alle Teilchen eines Bündels gleichphasig oszillieren. Diese Schwingungen sind unerwünscht und werden durch ein *Strahlphasenregelsystem* [4] gedämpft. Das geschieht durch eine gezielte Veränderung der Phasenlage der Beschleunigungsspannung(en), was im Widerspruch zur Synchronisation steht. Kavitätensynchronisation und Strahlphasenregelung beeinflussen sich daher gegenseitig und können nur gemeinsam analysiert werden. Auch die lokalen Regelkreise jeder einzelnen Kavität, welche die Amplitude der Beschleunigungsspannung und die Eigenfrequenz der Ferritkern-Kavitäten [5] regeln, müssen in diese Betrachtung aufgenommen werden.

A. Kavitätensynchronisation

Die Aufgaben der Kavitätensynchronisation sind folgende: Erstens sollen die Frequenzen aller Kavitäten Harmonische derselben Grundfrequenz sein. Zweitens sollen die Felder in den einzelnen Kavitäten eine konstante Phasenbeziehung zueinander haben.

Jede Kavität wird von einem Leistungsverstärker gespeist, der von einem *Digital-Synthesizer* (DS) angesteuert wird. In den so genannten *Versorgungsräumen*, in welchen alle elektronischen Komponenten strahlungsgeschützt untergebracht ist, befindet sich neben den Kavitäten-DS jeweils ein *Referenz-DS*. Der zeitliche Verlauf der Grundfrequenz wird vor Beginn des Beschleunigungszyklus in jedem Versorgungsraum in einer Kontrolleinheit eingespeichert und dann „abgespielt“. Das *White-Rabbit*-System [6] bewirkt eine Synchronisation der Uhren der Kontrolleinheiten bis auf etwa $1\,\mathrm{ns}$. Die Uhren der Synthesizer werden mittels des *BuTiS*-Systems [7] sogar noch präziser synchronisiert.

Um Störungen ausregeln zu können, berechnet zusätzlich ein lokales DSP-System [8] die Phasendifferenz zwischen der tatsächlichen Kavitätenspannung und dem Signal des Referenz-DS und regelt Frequenz und Phase des lokalen DS nach. Es liegt also eine räumlich verteilte Regelung vor.

B. Strahlphasenregelung

Über einen *Strahllagesensor* wird die Position der Teilchenbündel erfasst. Ein DSP-System berechnet die Phasendifferenz zwischen dem Schwerpunkt der Bündel und der tatsächlichen Beschleunigungsspannung. Mittels eines frequenzvariablen Filters [9] werden Strahlphasenschwingungen mit der charakteristischen Synchrotronfrequenz (Gl. 2) oder Harmonischen derselben detektiert. Daraus werden Frequenzkorrekturen errechnet [10], welche eine Dämpfung der longitudinalen Schwingungen bewirken sollen. Das Strahlphasen-Regelsystem versendet diese Frequenzkorrekturen in Abständen von $10\,\mu\mathrm{s}$ (so genannter T_0-*Takt*) an alle Versorgungsräume, wo sie beim nächsten T_0-Takt wirksam werden.

III. Problemstellung

In einem Synchrotron gibt es noch viele weitere Regelsysteme. Die Strahldynamik wird auch von den magnetischen Feldern der Ablenkmagnete und der magnetischen Linsen beeinflusst. Die dafür verantwortlichen Regelsysteme kommunizieren jedoch mit den hier betrachteten nicht und arbeiten in anderen Frequenzbereichen; zudem sind longitudinale und laterale Strahldynamik in erster Näherung voneinander unabhängig.

Die nachfolgend beschriebene Analyse beschränkt sich auf die longitudinale Strahlphasenregelung. Es handelt sich dabei um ein räumlich verteiltes MIMO-Regelsystem. Die Kommunikationslatenzen müssen als Totzeiten berücksichtigt werden. Der zu regelnde physikalische Prozess zeigt deutliche Nichtlinearitäten und sehr kleine Zeitkonstanten, so dass sich trotz Abtastraten im MHz-Bereich die Zeitdiskretisierung nicht vernachlässigen lässt. Die Laufzeit des frequenzvariablen Filters ist nicht konstant, so dass zustandsabhängige Totzeiten entstehen. Zudem sind einige Parameter der Regelstrecke, insbesondere die der Kavitäten, nicht exakt bekannt.

A. Analytische Betrachtung

Das Problem der Stabilitätsanalyse nichtlinearer Systeme mit Totzeiten ist bislang nicht vollständig gelöst [11]. Die bekannten Verfahren konzentrieren sich meist auf nur einen der aufgeführten Aspekte oder sind anderweitig eingeschränkt, etwa auf nur eine Totzeit (z. B. [12], [13]), oder auf Totzeiten, die im Vergleich zu den Zeitkonstanten des zu regelnden Prozesses klein sind (z. B. [14]), oder auf mehrere Totzeiten, die aber ganzzahlige Vielfache einer gemeinsamen Zeitkonstante sein müssen, was die Analyse sehr vereinfacht [15], in dem hier betrachteten System aber nicht zutrifft. Je nach Art der Nichtlinearitäten ist eine geschlossen-analytische Betrachtung der Systemstabilität gar nicht möglich und die Stabilitätsanalyse muss numerisch erfolgen [16].

Ein „Arbeitspunkt“ lässt sich in einem Synchrotron nicht ohne weiteres definieren, da einige Größen eine große Dynamik aufweisen; beispielsweise kann die Synchrotronfrequenz im SIS100 zwischen unter $100\,\mathrm{Hz}$ und mehreren kHz betragen. Interessanter als die Stabilität eines bestimmten Arbeitspunkts ist die Frage, ob das Gesamtsystem in der Lage ist, einem von außen vorgegebenen Frequenz- und Phasenverlauf, der so genannten *Rampe*, zu folgen.

B. Numerische Simulation

Aus den vorgenannten Gründen wurde entschieden, ein Systemmodell des Synchrotrons und der relevanten Regelsysteme zu entwickeln. Das bringt eigene Probleme mit sich: Modellbildung bedeutet immer die Vernachlässigung von Details. Deshalb müssen die Simulationsergebnisse durch Vergleich mit Messwerten verifiziert werden, was aber für formale Betrachtungen ebenso gilt. Darüber hinaus führt die numerische Berechnung zu zusätzlichen Fehlern. Jedes einzelne Simulationsergebnis stellt nur einen Punkt des Entwurfsraums dar; Verallgemeinerungen daraus sind mit Vorsicht zu genießen.

Als Modellierungs- und Simulationsprogramm kommt *Ptolemy II* [17] zum Einsatz. Ptolemy unterstützt *heterogene* Modelle, deren Teile unterschiedlichen Berechnungsmodellen („Domänen“ genannt) folgen. Der zu regelnde physikalische Prozess und die analoge Elektronik werden durch zeitkontinuierliche Differentialgleichungen (CT-Domäne) beschrieben, die digitale Signalverarbeitung durch synchrone Datenflussgraphen (SDF-Domäne) und die Kommunikation in ereignisdiskreter Form (DE-Domäne). Außerdem ist Ptolemy *quelloffen*, so dass es möglich war, einige anwendungsspezifische Probleme mit eigenem Programmcode zu lösen. Leider werden die verwendeten Berechnungsmodelle sequentiell in einem einzigen Prozess abgearbeitet und moderne Mehrkernprozessoren daher nicht ausgenutzt. Nachteilig ist außerdem, dass Ptolemy nicht in der Lage ist, algebraische Schleifen selbsttätig aufzulösen.

C. Ziele

Mittels des Systemmodells sollte zunächst die Machbarkeit des an der GSI erarbeiteten Regelungskonzepts und seine Robustheit gegenüber Parameterabweichungen und Störungen gezeigt werden. Darüber hinaus sollten die für die Leistungsfähigkeit des Gesamtfaktors kritischsten Faktoren zwecks gezielter Optimierung identifiziert und Spielräume für zukünftige Erweiterungen ausgelotet werden [18], [19]. Das entstandene Modell eignet sich aber auch zur Optimierung von Reglerparametern und -strukturen und als Referenz beim Entwurf anwendungsspezifischer Komponenten [20], [21].

IV. Vorgehensweise

Bei der Entwicklung von Mikroprozessorarchitekturen und anwendungsspezifischen integrierten Schaltungen ist die Entwurfsraumexploration aufgrund der Vielzahl an Entwurfsparametern ebenfalls ein großes Problem, und man setzt möglichst abstrakte (*High-Level*) Modelle ein [22], [23]. Das Systemmodell wurde daher von Anfang an so einfach und so abstrakt wie möglich gehalten. Beispielsweise wurde — da nur kohärente Schwingungen von Interesse sind — ein Ansatz gewählt, bei dem ein so genanntes *Makropartikel* stellvertretend für alle Teilchen eines Bündels steht. Dies geschah, um die zur Simulation des Modells benötigte Rechenzeit so kurz wie möglich zu halten und somit die Durchführung möglichst

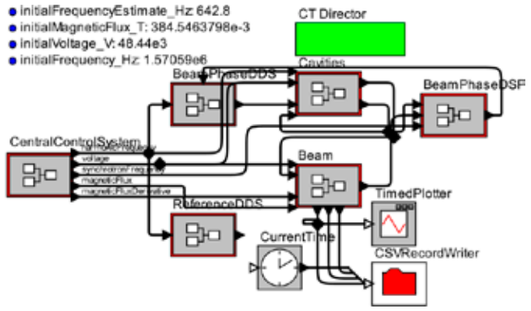

Abb. 1. Gesamtansicht des Ptolemy-II-Modells

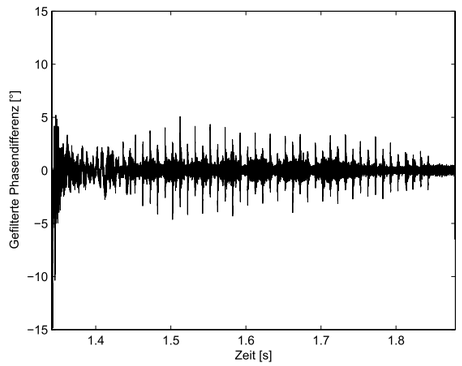

Abb. 2. Gemessene kohärente Dipolschwingungen

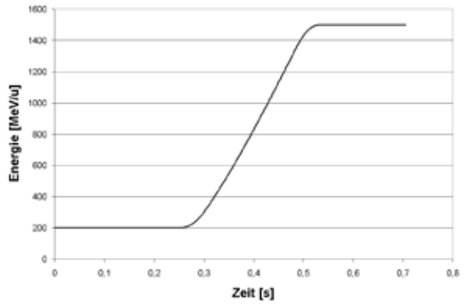

Abb. 3. Verlauf der Strahlenergie

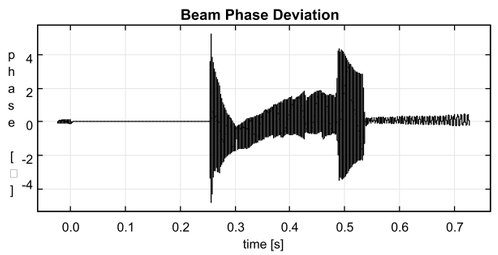

Abb. 4. Simulierte Abweichung von der synchronen Phase

zahlreicher Simulationen zu ermöglichen. Auf diese Weise wurde es möglich, ganze Beschleunigungszyklen von der Injektion der Teilchen bis zur ihrer Extraktion zu simulieren, was dieses Modell von anderen (z. B. [24]) unterscheidet. Die Simulation eines Zyklus' mit einer Dauer von $750\,\mathrm{ms}$ nimmt allerdings ca. $4\,\mathrm{h}$ auf einem Intel-Core-Prozessor mit $2,4\,\mathrm{GHz}$ in Anspruch.

A. Entwicklung und Verifikation

Die Entwicklung des Modells geschah in enger Zusammenarbeit mit der GSI. Abb. 1 zeigt die oberste Hierarchieebene des Modells.

Die physikalische Plausibilität der Simulationsergebnisse wurde regelmäßig überprüft. Abb. 2 zeigt Ergebnisse eines Experiments, welches im Jahre 2010 am SIS18 vorgenommen wurde. Dabei wurden $^{40}Ar^{18+}$-Ionen beschleunigt. Da das synchrone Teilchen nur eine fiktive Referenz darstellt und seine Position folglich nicht gemessen werden kann, wurde die Phasendifferenz zwischen Beschleunigungsspannung und Strahllagesignal mittels eines Bandpasses gefiltert, um kohärente Dipolschwingungen mit der Synchrotronfrequenz zu identifizieren. Die maximale Amplitude der Schwingungen beträgt etwa $5°$, die mittlere Amplitude etwa $1°$.

B. Referenzszenario

Ein typischer Anwendungsfall für das SIS100 wird die Beschleunigung von $^{238}U^{28+}$-Ionen sein. Aus diesem Grund wurde genau dieses Szenario auch als Testfall herangezogen. Abb. 3 zeigt den zeitlichen Verlauf der Teilchenenergie und Abb. 4 ein Ergebnis der Simulation, nämlich die Abweichung der Phase des Strahlschwerpunkts von der synchronen Phase (in Grad). Zu Beginn und Ende des Beschleunigungszyklus' treten besonders große Abweichungen auf, weil sich zu diesen Zeitpunkten die im System auftretenden Frequenzen besonders rasch ändern, was zu vorübergehenden Regelabweichungen führt. Die Werte liegen im erwarteten Bereich. In der Abbildung nicht erkennbar ist, dass es sich nicht um eine Dauerschwingung handelt, sondern in Abständen von $2\,\mathrm{ms}$ Schwingungen angeregt werden, die mit einer Zeitkonstanten unter $1\,\mathrm{ms}$ ausgeregelt werden.

C. Parameterraum

Das Modell besteht aus Teilmodellen, die entweder genau ein Mal vorkommen (z. B. das Modell der Strahlphysik) und Teilmodellen, die für jede der N Beschleunigungskavitäten je ein Mal instanziert werden. Insgesamt hat das Modell $25 + 14 \cdot N$ Parameter. Die Anzahl N der Kavitäten ist ebenfalls ein Parameter des Modells und definiert zusammen mit $3 + N$ weiteren Parameter die baulichen Gegebenheiten des Beschleunigers. 8 Parameter geben die Startbedingungen der Simulation an und müssen passend zu der simulierten Rampe gewählt werden. Von den verbleibenden Parametern sind $3+10 \cdot N$ unsichere Parameter der Regelstrecke und der Rest die Reglerparameter; zu letzteren zählen auch die nur indirekt beeinflussbaren Latenzen.

In Experimenten (z. B. [4]) konnten bereits hinreichend gute Reglerparameter bestimmt und die für Berechnung und Kommunikation benötigten Latenzzeiten abgeschätzt werden. Ferner existieren bereits Einzelkomponenten (z. B. [8], [9], [25]), deren charakteristische Eigenschaften Eingang in das Modell fanden.

D. Robustheits- und Sensitivitätsanalyse

Es wurden zunächst nur die unsicheren Parameter betrachtet. Deren Streubreite um die angenommenen Werte ist ebenfalls unsicher und wurde zunächst mit 20% angesetzt. Die lokalen Regler wurden für die angenommenen idealen Parameter der Regelstrecke entworfen. Es wurde daher angenommen, dass jede Abweichung von diesen idealen Parametern die Regelgüte nur verschlechtern kann. Unter diesen Annahmen wurden unterschiedliche *Corner Cases* definiert, welche den Kombinationen aus oberen und unteren Extremwerten und der Idealwerte der Parameter entsprechen. Anhand der Simulationsergebnisse wurde der schlechteste dieser Fälle identifiziert und für diesen der Betrag der Abweichungen sukzessive vergrößert. Zwischenzeitlich durchgeführte Untersuchungen [26] kamen zu dem Schluss, dass die meisten Parameter mit einer Unsicherheit von unter 10% behaftet sind. Selbst bei Abweichungen von $\pm 50\%$ bleibt das System in Simulationen stabil.

Für die bekannten Entwurfsparameter wurden anschließend Simulationen durchgeführt, bei denen jeweils ein Parameter so lange verändert wurde, bis der sinnvolle Wertebereich verlassen oder das System instabil wurde. Die Größe des so bestimmten zulässigen Wertebereichs ist auch ein Maß für die Empfindlichkeit des Systems gegenüber Änderungen des jeweiligen Parameters. Aus diesen Ergebnissen ist ersichtlich, welche Parameter sich zu optimieren lohnen und welche bei zukünftigen Erweiterungen zum begrenzenden Faktor werden könnten.

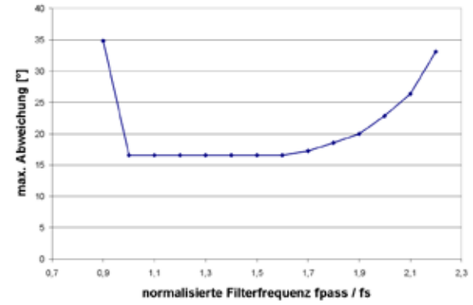

Abb. 5. Beispiel einer Gütefunktion

E. Metriken

Von unterschiedlicher Seite wurden zahlreiche mögliche Metriken zur Beurteilung der Regelgüte vorgeschlagen. Auf Basis von nahezu 200 im Rahmen der geschilderten Sensititivätsanalyse durchgeführten Simulationen wurde die Korrelation der Metriken untereinander untersucht. Dazu wurde der Kendall-Rangkorrelationskoeffizient [27] verwendet, weil es sich hierbei um ein robustes Verfahren handelt, welches keine Annahmen über die Verteilung der Variablen oder die Abhängigkeit zwischen ihnen macht. Wie sich herausstellt, korrelieren nahezu alle Metriken auf hohem Konfidenzniveau miteinander, was darauf zurückzuführen ist, dass sie letztendlich alle auf den gleichen von der Simulation berechneten Größen beruhen. Als anschaulichste Metriken wurden daher die maximale und mittlere Abweichung des Strahlschwerpunkts von der synchronen Phase sowie die mittlere Zeitkonstante, mit der diese Abweichungen abklingen, als anschaulichste Metriken gewählt. Die von Kang und Kumar [28] vorgeschlagene Korrelationsanalyse zwischen Parametern und Metriken ergab dagegen keine signifikanten Korrelationen. Mögliche Gründe dafür sind die hohe Anzahl der Parameter und die Tatsache, dass alle Metriken auf den gleichen Größen basieren.

Kleine Abweichungen vom Idealwert der Parameter bewirken nur geringe Änderungen der Metriken, nahe des Rands des Stabilitätsbereichs ändern sich die Funktionswerte deutlich. Ein Beispiel ist in Abb. 5 dargestellt. Außerhalb des dargestellten Bereichs ist das Gesamtsystem instabil; der ideale Parameterwert liegt bei $1,1$.

V. Ausblick

Anscheinend sind die verwendeten Gütefunktionen stetig und glatt. Sie eignen sich daher gut zur Steuerung gerichteter Optimierungsverfahren, mit deren Hilfe unter Annahme eines einfach-zusammenhängenden Stabilitätsbereichs der Rand desselben geortet und optimale Reglerparameter bestimmt werden können. Ferner kann (beispielsweise mittels Regressionsrechnung) der unbekannte Zusammenhang zwischen Parametern und Metriken mathematisch angenähert werden; die weitere Optimierung kann dann anhand dieser Näherung erfolgen (prediktive Optimierung [29]).

Danksagung

Das hier vorgestellte Projekt wird durch das BMBF finanziert (FKZ 06DA9028I).

Literatur

[1] GSI Helmholtzzentrum für Schwerionenforschung GmbH: *GSI Helmholtzzentrum für Schwerionenforschung*. http://www.gsi.de

[2] Pirkl, W.: Longitudinal beam dynamics. In: *CERN Accelerator School*, 1993, S. 233–257

[3] Klingbeil, H. ; Damerau, H. ; Kumm, M. ; Moritz, P. ; Schreiber, G. ; Zipfel, B.: Commissioning of the SIS12/18 Cavity Synchronization System. In: *GSI Scientific Report* (2005), S. 132

[4] Klingbeil, H. ; Zipfel, B. ; Kumm, M. ; Moritz, P.: A Digital Beam-Phase Control System for Heavy-Ion Synchrotrons. In: *IEEE Trans. Nuclear Science* 54 (2007), Nr. 6, S. 2604–2610

[5] Klingbeil, H.: Ferrite Cavities. In: *CERN Accelerator School*, 2010, S. 299–317

[6] Serrano, J.: The white rabbit project. In: *Proc. Int'l. Conf. Accelerators and Large Experimental Physics Control Systems*, 2009, S. 93–95

[7] Moritz, P.: BuTiS — Development of a Bunchphase Timing System. In: *GSI Scientific Report* (2006), S. 64

[8] Klingbeil, H.: A Fast DSP-Based Phase-Detector for Closed-Loop RF Control in Synchrotrons. In: *IEEE Trans. Instrumentation and Measurement* 54 (2005), Nr. 3, S. 1209–1213

[9] Surapong, P. ; Glesner, M. ; Klingbeil, H.: Implementation of Realtime Pipeline-Folding 64-Tap Filters on FPGA. In: *Ph. D. Research in Microelectronics and Electronics*, 2010, S. 1–4

[10] Lens, D. ; Klingbeil, H. ; Gussner, T. ; Popescu, A. ; Gross, K.: Damping of Longitudinal Modes in Heavy-Ion Synchrotrons by RF-Feedback. In: *Proc. Int'l. Conf. Control Applications*, 2010, S. 1737–1742

[11] Dong, Y. ; Mei, S. ; Wang, X.: Novel Stability Criteria of Nonlinear Uncertain Systems with Time-Varying Delay. In: *J. Abstract and Applied Analysis* (2011), Nr. 969674

[12] Michiels, W. ; Verheyden, K. ; Niculescu, S.-I.: Mathematical and Computational Tools for the Stability Analysis of Time-Varying Delay Systems and Applications in Mechanical Engineering. In: Chiasson, J. (Hrsg.) ; Loiseau, J. J. (Hrsg.): *Applications of Time Delay Systems*. Springer, 2007, S. 199–216

[13] Tracht, R. ; Thorausch, M.: Stabilität von Regelkreisen mit periodisch veränderlicher Totzeit. In: *at* 52 (2004), Nr. 12, S. 562–568

[14] Yook, J. K. ; Tilbury, D. M. ; Soparkar, N. R.: Trading computation for bandwidth — reducing communication in distributed control systems using state estimators. In: *IEEE Trans. Control Systems Technology* 10 (2002), Nr. 4, S. 503–518

[15] Gu, K. ; Nicolescu, S.-J.: Survey on Recent Results in the Stability and Control of Time-Delay Systems. In: *J. Dynamic Systems, Measurement and Control* 125 (2003), Nr. 2, S. 158–165

[16] Rösch, O. J.: *Regelung dynamischer Systeme mit stochastischer Zeitverzögerung durch ein Kommunikationsnetzwerk*, Universität Siegen, Diss., 2006

[17] The Ptolemy Project: *Ptolemy II Home Page*. http://ptolemy.eecs.berkeley.edu/ptolemyII

[18] Spies, C. ; Zipf, P. ; Glesner, M. ; Klingbeil, H.: Bandwidth Requirement Determination for a Digitally Controlled Cavity Synchronisation in a Heavy Ion Synchrotron Using Ptolemy II. In: *Proc. Int'l. Symp. Rapid System Prototyping*, 2008, S. 196–202

[19] Spies, C. ; Zipf, P. ; Guntoro, A. ; Glesner, M. ; Klingbeil, H.: A model-based development approach for multi-cavity RF control systems. In: *GSI Scientific Report* (2009), S. 113

[20] Samman, F. A. ; Surapong, P. ; Spies, C. ; Glesner, M.: Floating-point-based hardware accelerator of a beam phase-magnitude detector and filter for a beam phase control system in a heavy-ion synchrotron application. In: *Proc. Int'l. Conf. Accelerators and Large Experimental Physics Control Systems*, 2011, S. 683–686

[21] Surapong, P. ; Spies, C. ; Glesner, M. ; Klingbeil, H.: Design of frequency-variable digital filters for beam phase control. In: *GSI Scientific Report* (2011)

[22] Holzer, M. ; Knerr, B. ; Rupp, M.: Design Space Exploration with Evolutionary Multi-Objective Optimisation. In: *Proc. Int'l. Symp. Industrial Embedded Systems*, 2007, S. 126–133

[23] Catania, V. u. a.: An Effective Methodology to Multi-objective Design of Application Domain-specific Embedded architectures. In: *Proc. Euromicro Conf. Digital System Design*, 2009, S. 643–650

[24] Lens, D. ; Grieser, J. ; Klingbeil, H.: Longitudinal Closed-Loop Beam Control in Heavy Ion Synchrotrons: Simulation Methods and Results. In: *Proc. Int'l. Conf. Simulation Technology*, 2011

[25] Guntoro, A. ; Glesner, M.: Resolving longitudinal amplitude and phase information of two continuous data streams for high-speed and real-time processing. In: *Advances in Radio Science* (2009), S. 133–137

[26] Hartel, U. K.: *Modellierung des Regelungs- und Steuerungssystems einer Beschleunigungseinheit für Synchrotrons*, Technische Universität Darmstadt, Diplomarbeit, 2011

[27] Kendall Rank Correlation Coefficient. In: *The Concise Encyclopedia of Statistics*. Springer, 2008, S. 278–281

[28] Kang, S. ; Kumar, R.: Magellan — A Search and Machine Learning-based Framework for Fast Multi-core Design Space Exploration and Optimization. In: *Design, Automation and Test in Europe*, 2008, S. 1432–1437

[29] Ozisikyilmaz, B. ; Memik, G. ; Choudhary, A.: Efficient system design space exploration using machine learning techniques. In: *Proc. Design Automation Conf.*, 2008, S. 966–969

Cyber-Physical Systems im Maschinen- und Anlagenbau – ein Konzept für die Zukunft?

Alexander Dicks, Martyna Bator, Volker Lohweg
inIT-Institut Industrial IT,
Hochschule Ostwestfalen-Lippe,
Liebigstr. 87, 32657 Lemgo
Email: {alexander.dicks, martyna.bator, volker.lohweg}@hs-owl.de

Sebastian Faltinski, Oliver Niggemann
Fraunhofer IOSB-INA,
Anwendungszentrum Industrial Automation,
Langenbruch 6, 32657 Lemgo
Email: {sebastian.faltinski, oliver.niggemann}@iosb-ina.fraunhofer.de

***Zusammenfassung*—Der Betrieb komplexer Maschinen- und Anlagen benötigt zunehmend Konzepte und Entwicklungen, welche stark von der Informatik, Informationstechnologien (IT) und der industriellen Automation getrieben werden. Dabei spielt die Modellierung und Diagnose von Maschinen und Anlagen eine bedeutende Rolle. Mit Hilfe neuer Konzepte, die einerseits multisensorische Datenerfassung und -verarbeitung bzw. Informationsfusion berücksichtigen und andererseits neue Diagnose- und Modellierungsmethoden anwenden, gelingt es, komplexe Systeme mit Hilfe von Agentenstrukturen zu realisieren. Ein Vorschlag zur Realisierung eines Cyber-Physical Systems mittels entsprechender Software Engineering Werkzeuge wird aufgezeigt.**

I. Einführung

Der Maschinen- und Anlagenbau profitiert zunehmend von neuen Konzepten und Entwicklungen, welche stark von der Informatik, Informationstechnologien (IT) und der industriellen Automation getrieben werden. Dabei spielt die Modellierung und Diagnose von Maschinen- und Anlagen eine bedeutende Rolle. Hinzu kommen beispielsweise die Fernwartung von Maschinen, die durchgängige Vernetzung mit Netzwerken, der zunehmende Einsatz von Funktechnologien und die Sensor- und Informationsfusion für die zustandsorientierte und vorausschauende Maschinenwartung sowie Industrielle Bildverarbeitung und Mustererkennung (IBV&M) als eine Schlüssel-Technologie für die Qualitätssicherung in produzierenden Unternehmen.

Interdisziplinäre Ansätze aus Technik, Biologie und Psychologie ermöglichen dabei neue zukunftsweisende Lösungen, die sich vielfach in der Fusion von Information abbildet. Insbesondere bilden Konzepte der Multi-Sensorik, Vernetzung und Adaption innerhalb verschiedener Netzwerke die Basis zur Anwendung von Cyber-Physical Systems (CPS). Weiterhin erfährt der Themenkreis der Energieeffizienz und das energie-optimale Betreiben großer Fertigungsanlagen aktuell eine zunehmende Beachtung.

Der genannte Sachverhalt zeigt auf, dass die Komplexität heutiger Anlagen nur mit Hilfe multidisziplinärer Systeme – Cyber-Physical Systems – beherrscht werden kann. Diese müssen einerseits eine gewisse Autonomie aufweisen, andererseits muss die Komplexität im Sinne von Datenkonsistenz und -plausibilität selbst beherrscht werden. Weiterhin muss ein CPS in der Lage sein, die Anforderungen des heutigen Maschinen- und Anlagenbaues zu berücksichtigen.

Hierzu zählen unter anderem: Erforderliche Echtzeitfähigkeit und hohe Verfügbarkeitsanforderungen von Produktionsprozessen; beschränkte Ressourcen (Speicher, Prozessorleistung) der eingesetzten Gerätetechnik; notwendige Systemintegration der IuK-Technologien in bestehende Strukturen; hohe Anforderungen an IT-Sicherheit und funktionale Sicherheit; Konflikt zwischen langfristiger Verfügbarkeit von Investitionsgütern und der Änderungsgeschwindigkeit von IuK-Technologien; Verwendung von vorhandener Sensorik; Adaptionsfähigkeit von Systemen im Lebenszyklus.

Wichtige Faktoren in der Realisierung von CPS sind u. a. die Beherrschung der *Prozessechtzeit*, *Lokalitätsprinzip und lokale Kognition*, *verteilte eingebettete Kommunikation*, *Anlagen- und Systemmodellierung, Simulation und Diagnose* und *Software Language Engineering* (SLE).

In den folgenden Kapiteln werden die wichtigsten Konzepte, die sich u. a. im Maschinen- und Anlagenbau aktuell herauskristallisieren, vorgestellt.

II. Modellierung, Simulation und Diagnose

Im Bereich der Diagnosealgorithmen für technische Systeme kann zwischen sogenannten 'Black Box'-, 'Grey Box'- und 'White Box'-Modellen unterschieden werden. Die Modellierungen beider Rand-Konzepte besitzen Vor- und Nachteile, die nachfolgend kurz beschrieben werden. In der Regel werden die Ansätze gekoppelt, so dass typischerweise 'Grey Box'-Modelle entstehen.

Simulationsmodell-basierte Ansätze (White Box) (vgl. [1], [2], [3]) nutzen ein Modell des Systems, um während des Systembetriebs (i) durch einen Vergleich zwischen Messungen am System und der Modellvorhersage (Simulation des Modells) Symptome zu generieren und (ii) um den Zusammenhang zwischen Symptom und Ursache zu ermitteln. Punkt (ii) geschieht oft, indem hypothetische Fehlerursachen in das Modell integriert werden und dann durch eine erneute Simulation überprüft wird, ob die hypothetische Ursache alle Symptome erklären könnte. Modellbasierte Ansätze haben den Vorteil, dass Symptome und Fehlerursachen nicht bei Implementierung des Diagnosealgorithmus bekannt sein müssen und das Diagnosesystem damit auf unerwartete Fehler reagieren kann. Nachteilig ist, dass die Modellierung i) aufwendig werden kann und aus Aufwandsgründen die notwendige Präzision des Modells nicht der Realität standhält, ii) entsprechende Lernverfahren zur Modellrealisierung nicht alle möglichen Zustände des Modells erfassen. Modellbasierte Ansätze wurden z.B. in [4], [3] auf die Verfahrenstechnik angewendet. Erweiterungen der modellbasierten Diagnose ('Model Compilation', vgl. [5], [6], [7]) verlegen die Schritte (i) und (ii) weg vom Anlagenbetrieb in die frühere Planungsphase. Dies geschieht, indem auch das fehlerhafte System nur simuliert wird und dann aus den Simulationsergebnissen Klassifikationsfunktionen abstrahiert werden.

Metrologiemodell-basierte Ansätze (Grey Box) (vgl. [8], [9], [10], [11], [12]) basieren auf einer Modellierung, die in der Regel auf der messtechnischen Erfassung von Phänomenen an technischen Systemen basiert. Multi-Sensor-Fusion bezeichnet diese Zusammenführung und Aufbereitung von bruchstückhaften und teilweise widersprüchlichen Sensordaten in ein homogenes, für den Menschen verständliches Gesamtbild der aktuellen Situation. Hierzu ist es notwendig, Experten (Humanwissen) hinzuzuziehen, um eine geeignete Sensormodellierung im Sinne einer Fehleranalyse und Systemdiagnose herbeizuführen. Üblicherweise werden zur adaptiven Modellierung lernende Systeme, basierend auf künstlichen neuronalen Systemen oder Fuzzy-Systemen eingesetzt. Die Vorteile dieses Konzeptes liegen i) in der verhaltensorientierten Modellierung eines Systems (Behaviour), das durch Expertenwissen modelliert werden kann und ii) in der Erfassung der Systemzustände ohne Details des realen (komplexen) Systems zu kennen. Allerdings sind auch Nachteile dieses Ansatzes zu nennen: i) Die Qualität hängt stark vom Know-how der Experten ab und ii) kann der Fall auftreten, dass Sensoren nicht alle auftretende Effekte erfassen oder ungeeignete Sensoren eingesetzt werden.

Bei *heuristischen Verfahren* (Black Box) (Heuristiken, vgl. [13], [14]) wird mittels einer Klassifikationsfunktion direkt von Symptomen auf die Fehlerursache geschlossen. Klassifikationsfunktionen werden dabei oft mittels Entscheidungsbäumen, neuronalen Netzen oder Fuzzy-Ansätzen modelliert. Diese dienen zur näherungsweisen Lösung von komplexen Entscheidungs- und Optimierungsproblemen (Approximationsproblem). Heuristiken finden dort ihren Einsatz, wo andere Ansätze wegen des zu hohen Rechenaufwands scheitern. Diese Methoden haben den Nachteil, dass vorab alle Symptome und Ursachen (und deren Kausalität) bekannt sein müssen.

A. Wissensbasierte Systeme

Als wissensbasiertes System wird ein System bezeichnet, dass auf der Basis von (Experten)wissen zur Lösung oder Bewertung bestimmter Problemstellungen dient. Hierbei geht es immer darum, intelligentes Denken und Handeln in einem bestimmten Bereich zu simulieren und immer muss zu diesem Zweck Wissen dargestellt und verarbeitet werden [15]. Das erste System, das erfolgreich Wissen in der Form einer großen Anzahl von Regeln aus einem spezifischen Anwendungsbereich einsetzte, wird in [16] beschrieben. Dieses System aus dem Jahre 1969 dient zur Interpretation von Massenspektrogrammen, bei dem Molekül-Struktur-Formeln bestimmt werden. Zentraler Punkt eines wissensbasierten Systems ist die Trennung zwischen der Darstellung des Wissens über den betreffenden Problembereich (Wissensbasis) und der Verarbeitung dieses Wissens (Wissensverarbeitung). Während die Wissensbasis das spezifische Wissen über den Anwendungsbereich repräsentiert, stellt die Wissensverarbeitung eine anwendungsunabhängige Problemlösungskomponente dar. Es können u.a. folgende Aspekte realisiert werden:

- Klare Trennung zwischen Problembeschreibung und Problemlösung
- Wissen über den Anwendungsbereich ist direkt ausdrückbar.

Die wichtigsten Realisierungsmethoden sind fallbasierte Systeme, regelbasierte Systeme und Entscheidungsbäume. Es befinden sich mittlerweile weltweit eine unzählige Anzahl solcher Systeme im Einsatz ([17], [18], [19], [20]). Diese Anwendungen beinhalten:

- Einbindung von Expertenwissen
- Signalfilterung, Merkmalsextraktion, Suchen von Informationen
- Datenverwaltung und Informationscodierung in Wissensdatenbanken
- Fehler-Klassifikation, Diagnose und Lokalisierung
- Wissensbasis durch Simulation und/oder Experimente

Nachfolgend werden wissensbasierte Systeme vorgestellt, die in Systemen der Automation verwendet werden.

1) Künstliche neuronale Netze: Künstliche neuronale Netze (KNN) sind eines der ältesten Beispiele für den Einsatz von künstlicher Intelligenz (KI) in der Automation. Mittlerweile ist der Anwendungsbereich der KNN in zwei Bereiche unterteilbar [21]:

- Künstliche neuronale Netze, die modelliert werden, um menschliches Verhalten und Erleben bzw. die Funktionsweise des menschlichen Gehirns besser zu verstehen.
- Neuronale Netze, die dazu dienen, konkrete Anwendungsprobleme aus Bereichen wie z. B. Statistik, Wirtschaftswissenschaften, Technik und vielen anderen Gebieten zu lösen.

Im Maschinen- und Anlagenbau geht es um die Abstraktion von Informationsverarbeitung und nicht um die Nachbildung von biologischen neuronalen Netzen. Hierbei sind die KNN in der Lage, nichtlineare Zusammenhänge mit mehreren Ein- und Ausgabevariablen abzubilden. Diese Zusammenhänge werden anhand von Daten gelernt. Diese Daten können sowohl aus realen Messdaten oder aus Daten, die mittels eines Modells erzeugt wurden, stammen. Das neuronale Netz gibt es so nicht. Es gibt viele verschiedene Paradigmen, was neuronale Netze sind, wie sie trainiert und wo sie eingesetzt werden. KNN finden in Bezug auf den Maschinen- und Anlagenbau Anwendung auf dem Gebiet der Zustandsüberwachung und der Fehlerdiagnose ([22], [23], [24], [25]). Hierbei erfüllen diese eine oder mehrere der folgenden Aufgaben:

- Mustererkennung, Parameterschätzung, nichtlineare Abbildung angewandt auf die Zustandsüberwachung
- Anlernen der KNN mit simulierten und/oder experimentellen Signalen aus dem Zeit- und Frequenzbereich
- Echtzeitfähigkeit, nicht überwachte Online-Diagnose
- Dynamische Anpassung der Struktur ohne das gesamte KNN neu anzulernen
- Herausfiltern von Transienten, Störungen und Rauschen
- Fehlervorhersage im Anfangsstadium des Betriebs

2) Fuzzy-Systeme: Im täglichen Leben verpacken wir Informationen in vage, aber dennoch fassbare Aussagen: Maschinenführer handeln in der Regel in dieser Art und Weise. In vielen Fällen trägt diese Vagheit mehr Information als absolute Zahlenwerte. Diese können aber nur im Kontext bewertet werden. Des Weiteren existieren Fuzzy-Konzepte (siehe z.B. [26], [11], [10], [12]), die über Zugehörigkeitsfunktionen eine Klassifikation von Informationen durchführen. Diese beruhen auf dem von Steffen F. Bocklisch entworfenen Grundkonzept zur Prozessanalyse mit unscharfen Verfahren [27]. Hierbei gilt die besondere Betrachtungsweise dem Prozess, denn hierbei geht es um Handlungsabläufe in der Natur und der Technik, genauer um die Veränderung eines Systems. Dieses äußert sich in Zeitverläufen von Signalen, Zuständen, usw., die hier näher betrachtet werden. Fuzzy-Systeme haben sich als leistungsstarke KI-Techniken bewährt. Einige der dargestellten Fuzzy-Systeme, die für die Fehlerdiagnose im Maschinen- und Anlagenbau (vgl. u a. [28], [29], [30]) verwendet werden, beinhalten folgende Punkte:

- Evaluierung der Performanzindikatoren mit Hilfe linguistischer Variablen
- Vorhersage und Lokalisierung von Störungen
- Verwendung von Expertenwissen
- System-Modellierung, nichtlineare Abbildung und Optimierung von Diagnose-Systemparametern durch adaptive Fuzzy-Systeme
- Fehlerklassifikation und -prognose

III. Multisensorik

Die Sensoren sind zu Kommunikations- und Datenerfassungszwecken vorzugsweise drahtlos mit der 'intelligenten Elektronik' des Antriebs verbunden. Die autonome Energieversorgung der Sensoren erfolgt unter Verwendung vorhandener technischer Lösungen. In der intelligenten Elektronik ist die Ansteuerung der Sensoren für solche Aufgaben, wie die zeitliche Synchronisation oder die Festlegung individueller Messzeitpunkte und -intervalle, integriert. Hier erfolgt auch die Abarbeitung

lokaler Algorithmen zur Signalvorverarbeitung (inklusive Behandlung von Messfehlern und Plausibilitätsprüfung), zur Datenfusion und Merkmalsextraktion sowie zum Verhalten bei Fehlern oder in Gefahrensituationen (vgl. [11]). Autonomiekonzepte sehen üblicherweise innerhalb einer intelligenten Elektronik ebenfalls ein Management zur Beherrschung der enormen zu erwartenden Datenmengen vor. Hierbei spielen Algorithmen der zielgerichteten Datenfusion und -reduktion eine zentrale Rolle.

IV. Software Engineering

Wie bereits in Abschnitt II-A beschrieben, ist für die Bewertung des Zustandes einer Automatisierungsanlage eine Wissensbasis notwendig. Diese kann beispielsweise während des Betriebs angelernt werden. Sie kann jedoch auch aus Planungsdaten abgeleitet werden, sodass eine Anlagenüberwachung bereits ab der Inbetriebnahme möglich ist. Die dafür notwendigen Planungsdaten bestehen beispielsweise aus IEC-61131 Programmen, 3D-Modellen oder Signalplänen. Gerade die Heterogenität der Daten, basierend auf unterschiedlichen Sichtweisen auf eine Anlage und die dafür notwendigen Softwarewerkzeuge führt zu einem Mehraufwand, wenn es darum geht, ein geeignetes diagnosefähiges Modell der Anlage zu erstellen.

Neue Ansätze im Bereich des Umgangs mit den Anlagenplanungsdaten bietet das XML-basierte Datenformat AutomationML. Ziel hierbei ist ein Datenformat, welches Planungsdaten verschiedener Planungswerkzeuge abbilden kann und zum Datenaustausch zwischen den Werkzeugen eingesetzt werden kann. Aktuell ist es möglich, SPS-Software in Form von PLCopenXML oder 3D-Modelle mittels COLLADA in AutomationML abzuspeichern [31]. Zusätzlich gibt es erste Ansätze, simulationsfähige Modelle, basierend auf 'Modelica' mit den Planungsdaten zu verknüpfen, sodass im Gegensatz zu alleinstehenden Modellen, die Semantik der Modelle im Anlagenzusammenhang gewährleistet ist [32]. Aus dieser umfassenden Anlagenbeschreibung lässt sich eine breite Wissensbasis erzeugen, die zu Diagnose und frühen Tests einzelner Anlagenteile beispielsweise in einem Hardware-in-the-Loop Test verwendet werden kann.

V. Agenten-basierte Cyber-Physical Systems

Ein Paradigma, welches in den letzten Jahren die Entwicklung der KI stark beeinflusst hat, ist das Konzept des Agenten im Zusammenhang mit CPS. Eine eindeutige Definition oder Charakterisierung des Begriffs Agent ist in der Literatur nicht zu finden. Vielmehr wird Agent als ein Konzept benutzt, dem man gewisse Eigenschaften zuschreibt, die man für unerlässlich hält im Hinblick auf die Realisierung gewisser Ziele oder Aufgaben. Eine Beschreibung, die die wesentlichen Aspekte eines Agenten auf den Punkt bringt ist z.B. in [33] zu finden:

'*Ein Agent ist ein Computersystem in einer Umgebung, das in der Lage ist, in dieser Umgebung autonom zu agieren, um seine Ziele zu realisieren.*'

Die meisten Agentensysteme sind daher Architekturen für Systeme, die in eine Umgebung eingebettet sind und in der Lage sind, Entscheidungen zu treffen.

Die agentenorientierte Diagnose und Beschreibung von technischen Systemen ist in den letzten Jahren erfolgreich in Forschungs- und Industrieprojekten eingesetzt worden ([34], [35], [36]). Zusätzlich sind die klassischen heuristischen bzw. regelbasierten Diagnoseverfahren in den letzten Jahren durch Modell-, Fall- und spektral-basierte Verfahren ergänzt worden (siehe [37], [38], [39]).

VI. Ein mögliches CPS-Konzept

Ausgehend von den weiter oben gemachten Ausführungen erscheint folgendes Konzept für CPS zielführend (vgl. Abbildung 1): Das Gesamtsystem ruht im Wesentlichen auf vier Säulen: Agententechnik, Informationsverarbeitung, Mustererkennung und Agenten-basierte Kommunikation in verschiedenen Netzwerk-Hierarchien (Echtzeit-Ethernet, JADE/FIPA). Damit greift es die formulierten Forschungs-und Entwicklungsfragen auf. Ein zu entwerfendes intelligentes autonomes System wird als vernetztes Embedded System im Sinne eines intelligenten Multiagentensystems ausgeführt und als CPS interpretiert. Die Sozialfähigkeit ist die Basis des Multiagentensystems. Das System basiert auf der Grundlage, dass einzelne Prozessagenten nur unvollständige Informationen und spezielle, jedoch beschränkte Fähigkeiten besitzen, so dass mehrere Agenten kooperieren müssen, um den geforderten Dienst zu erbringen, für Fehlertoleranz zu sorgen und übergeordnete Ziele zu verfolgen.
Die Prozessagenten der unteren Schicht interagieren kontinuierlich mit seiner lokalen Umgebung und erbringen einen nützlichen Dienst auf autonome Weise. Die Autonomie ist immer zielorientiert. Insbesondere den Prozessagenten obliegt die Aufgabe, autonom mit Informationen aus ihrer Umgebung die notwendigen Plausibilitätsprüfungen anhand von Klassifikationsergebnissen zu erbringen. Die notwendigen Informationen zur Klassifikation werden durch einen adaptiven Anlernvorgang erzeugt. Anhand der entsprechenden Ergebnisse entscheiden sie, welche Daten an die nächsthöhere Schicht weitergeleitet werden, um sicherzustellen, dass ausschließlich die für den Prozess notwendigen Daten übermittelt werden (Plausibilität und Aufmerksamkeit).

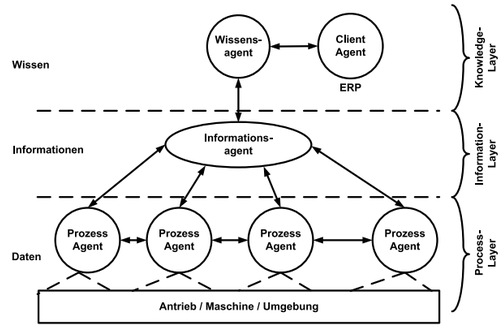

Abb. 1. Agenten-basiertes Cyber-Physical System

Eine besondere Herausforderung ist die Kommunikation zwischen Agenten auf den verschiedenen Ebenen wie z.B. der Feld-, Steuerungs-, Leit- und ERP-Ebene (Enterprise-Resource-Planning). Hierfür wird eine Middleware benötigt. Erste Projekte haben versucht, Middleware Ansätze für die Automation nutzbar zu machen z.B. OSACA [40], Socrades [41] oder der OPC-UA Standard [42]. Im Gegensatz zu anderen Ansätzen basiert OPC-UA auf dem klassischen, bereits etablierten OPC, und wird für die Anbindung von Leitebene an die Feldebene verwendet [43]. Aus Sicht der Middleware implementiert OPC-UA eine Server/Client Architektur, die sowohl über Webservices als auch unter Verwendung performaterer proprietärer Protokolle kommunizieren kann. Die Darstellung der Prozessdaten erfolgt im Server durch hierarchische Strukturen, sodass ein Rückschluss auf Beziehungen zwischen einzelnen Daten möglich ist. Aus Sicht der Automatisierungsapplikation erfolgt der Zugriff auf diese Daten über einen in OPC-UA standardisierten Satz von Schnittstellenmethoden.
Die nachfolgende Agentenschicht vergleicht bei einer Transaktion die üblichen Handlungsabläufe mit den bereits abgelegten Regeln und Methoden. Ein Wissensagent kooperiert mit den Prozessagenten der unteren Schicht, die von der Client-Ebene (Clientagent) über den Informationsagenten gelangen, um sich systemisch auf übergeordnete Ziele adaptieren zu können (ERP-Systeme). Anhand des multisensorischen Ansatzes soll es gelingen, Verhaltensanomalien im Sinne nicht zustandsgerechter oder ungewöhnlicher Situationen in komplexen Systemen zu unterscheiden.

VII. Zusammenfassung und Ausblick

Zusammenfassend lässt sich sagen, dass sich in den vorgenannten Themenfeldern der Automation intelligente autonome Systeme und simulationsbasierte Schadensmodellierung zur Überwachung und Analyse von Maschinen, Anlagen und Prozessen einsetzen lassen. Dennoch befindet sich die angewandte Forschung auf diesem Gebiet noch am Anfang. Die Prinzipien autonomer Systeme sind bekannt und werden im Bereich der Internettechnologien verbreitet eingesetzt – Stichworte hierzu sind Software-Multiagenten-Systeme, Web-Ontologien und deren Sprachen sowie semantische Technologien. Weiterhin besteht die Möglichkeit mit den heutigen Technologien Modellierungen von Maschinen und Anlagen, Sensorik und Kommunikationssystemen sowie Steuerungen zu realisieren. Problematisch erweisen sich jedoch durchgängige Hardware/Software-Lösungen und Middleware-Konzepte. Hier besteht im Bereich Cyber-Physical Systems verstärkter Forschungsbedarf.

Literatur

[1] P. Struss, "Fundamentals of model-based diagnosis of dynamic systems," in *International Joint Conference on Artificial Intelligence, Nagoya, Japan*, 1997.

[2] R. Isermann, "Model-based fault detection and diagnosis: status and applications," in *In Proceedings of the 16th IFAC Symposium on Automatic Control in Aerospace, St*, 2004, pp. 71–85.

[3] C. Frey, "Diagnosis and monitoring of complex industrial processes based on self-organizing maps and watershed transformations," in *Computational Intelligence for Measurement Systems and Applications, 2008. CIMSA 2008. 2008 IEEE International Conference on*, Jul. 2008, pp. 87 –92.

[4] D. Dvorak and B. Kuipers, "Process monitoring and diagnosis: a model-based approach," *IEEE Expert*, vol. 6, no. 3, pp. 67 –74, Jun. 1991.

[5] O. Niggemann, B. Stein, T. Spanuth, and H. Balzer, "Using models for dynamic system diagnosis - a case study in automotive engineering." Tagungsband des Dagstuhl-Workshops "Modellbasierte Entwicklung eingebetteter Systeme V"(MBEES), 2009, Schloß Dagstuhl, May 2009.

[6] B. Stein and U. Husemeyer, "Generierung heuristischer modelle zur diagnose," in *Proceedings of the 15th Symposium Simulationstechnik (ASIM 01), Paderborn, Germany*, 2001.

[7] B. Stein, O. Niggemann, and H. Balzer, "Diagnosis in automotive applications." 3rd Monet Workshop on Model-Based Systems (MBS-06) at the ECAI 06 at Riva de Gard, Italy, 2006, Aug. 2006.

[8] S. F. Bocklisch and U. Priber, Eds., *A Parametric Fuzzy Classification Concept*. Akademie-Verlag, Berlin, 1986.

[9] J. Schaede and V. Lohweg, "The mechanisms of human recognition as a guideline for security feature development," in *IST/SPIE 18th Annual Symposium on Electronic Imaging - Optical Security and Counterfeit Deterrence Techniques VI - Vol 6075*, R. L. van Renesse, Ed. SPIE, 2006, pp. 1–12.

[10] W. Dyck, T. Türke, J. Schaede, and V. Lohweg, "A new concept on quality inspection and machine conditioning for security prints," *Optical Document Security - The 2008 Conference on Optical Security and Counterfeit Deterrence; Reconnaissance International Publishers and Consultants, San Francisco, CA, USA*, Jan. 2008.

[11] V. Lohweg and U. Mönks, "Fuzzy-pattern-classifier based sensor fusion for machine conditioning," in *Sensor Fusion by Ciza Thomas*, C. T. (Ed), Ed. SCIYO, Sep. 2010.

[12] V. Lohweg, O. Niggemann, and (Eds.), "Machine learning in real-time applications (mlrta09 - ki 2009 workshop)," in *Lemgoer Schriften zur industriellen Informationstechnik, Vol. 3, Lemgo*, Sep. 2009.

[13] M. Pfeifer, T.; Richter, "Diagnose von technischen systemen - grundlagen, methoden und perspektiven der fehlerdiagnose." Deutscher Universitäts-Verlag, 1998.

[14] R. Bach, O. Niggemann, P. Winterling, and V. Zeller, "Die zuverlässigkeit der q-faktor-messung bei beliebigen signalstörungen." ITG Fachtagung Photonische Netze, Aug. 2003.

[15] G. K.-I. C. Beierle, "Methoden wissensbasierter systeme." Vieweg+Teubner, 2008.

[16] G. S. B. Buchmann, "Heuristic dendral: A program for generating explanatory hypotheses in organic chemistry." Computer Science Department, Stanford University, 1969.

[17] A. Elsaadawi, A. Kalas, and M. Fawzi, "Development of an expert system to fault diagnosis of three phase induction motor drive system," in *Power System Conference, 2008. MEPCON 2008. 12th International Middle-East*, Mar. 2008, pp. 497 –502.

[18] F. Filippetti, M. Martelli, G. Franceschini, and C. Tassoni, "Development of expert system knowledge base to on-line diagnosis of rotor electrical faults of induction motors," in *Industry Applications Society Annual Meeting, 1992., Conference Record of the 1992 IEEE*, Oct. 1992, pp. 92 –99 vol.1.

[19] T. S. Cheang and L. Zhang, "A new prototype of diagnosis system of inner-faults for three-phase induction motors developed by expert system," in *Electrical Machines and Systems, 2001. ICEMS 2001. Proceedings of the Fifth International Conference on*, vol. 1, 2001, pp. 312 –316 vol.1.

[20] M. Ahfaz Khan, A. Sharma, and R. Saxena, "Expert system for power transformer condition monitoring and diagnosis," in *Power Electronics, Drives and Energy Systems, 2006. PEDES '06. International Conference on*, Dec. 2006, pp. 1 –6.

[21] G. D. Rey and K. F. Wender, *Neuronale Netze: Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung*, 1st ed. Huber, Bern, Apr. 2008.

[22] F. Filippetti, G. Franceschini, C. Tassoni, and P. Vas, "Ai techniques in induction machines diagnosis including the speed ripple effect," *Industry Applications, IEEE Transactions on*, vol. 34, no. 1, pp. 98 –108, Jan. 1998.

[23] B. Karanayil, M. F. Rahman, and C. Grantham, "Identification of induction motor parameters in industrial drives with artificial neural networks," *Adv. Fuzzy Sys.*, vol. 2009, Jan. 2009.

[24] D. Gray, Z. Zhang, C. Apostoaia, and C. Xu, "A neural network based approach for the detection of faults in the brushless excitation of a synchronous motor," in *Electro/Information Technology, 2009. eit '09. IEEE International Conference on*, Jun. 2009, pp. 423 –428.

[25] X. Huang, T. Habetler, and R. Harley, "Detection of rotor eccentricity faults in a closed-loop drive-connected induction motor using an artificial neural network," *Power Electronics, IEEE Transactions on*, vol. 22, no. 4, pp. 1552 –1559, Jul. 2007.

[26] C. Kahraman, Ed., *Fuzzy multi-criteria decision making: Theory and applications with recent developments*, ser. Springer optimization and its applications. New York, NY: Springer, 2008.

[27] S. F. Bocklisch, *Prozeßanalyse mit unscharfen Verfahren*, 1st ed. Berlin: Verlag Technik, 1987.

[28] W. Wang, "A hybrid fuzzy system for real-time machinery health condition monitoring," A. T. Azar, Ed. InTech, 2010.

[29] L. Pereira, D. da Silva Gazzana, and L. Pereira, "Motor current signature analysis and fuzzy logic applied to the diagnosis of short-circuit faults in induction motors," in *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, Nov. 2005, p. 6 pp.

[30] S. Zouzou, W. Laala, S. Guedidi, and M. Sahraoui, "A fuzzy logic approach for the diagnosis of rotor faults in squirrel cage induction motors," in *Computer and Electrical Engineering, 2009. ICCEE '09. Second International Conference on*, vol. 2, Dec. 2009, pp. 173 –177.

[31] R. Drath, Ed., *Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA (VDI-Buch) (German Edition)*, 1st ed. Springer, Dec. 2009.

[32] M. Barth, O. Graeser, O. Niggemann, and A. Fay, "Integration und anwendung von objektorientierten simulationsmodellen in automationml," in *VDI Kongress AUTOMATION 2011*. Baden Baden, Jun 2011.

[33] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," no. 10, 1995.

[34] A. Liu, X. Yuan, and J. Yu, "Research and application of a hierarchical fault diagnosis system based on support vector machine," in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, vol. 2, Aug. 2007, pp. 59 –65.

[35] S. Enyedia, L. Miclea, H. Valean, I. Stoian, and G. Toderean, "Management of a multi-agent society used for monitoring and diagnosis in a hydroelectric power plant chain," in *Automation, Quality and Testing, Robotics, 2006 IEEE International Conference on*, vol. 2, May 2006, pp. 27 –32.

[36] C. Zhao, X. Zhao, Z. Tan, and X. Yan, "Research on multi-agent system model of diesel engine fault diagnosis by case-based reasoning," in *Innovative Computing, Information and Control, 2006. ICICIC '06. First International Conference on*, vol. 3, Sept. 2006, pp. 386 –389.

[37] D. Fischer, M. Boerner, J. Schmitt, and R. Isermann, "Fault detection for lateral and vertical vehicle dynamics," *Control Engineering Practice*, vol. 15, no. 3, pp. 315 – 324, 2007, selected Papers Presented at the Third IFAC Symposium on Mechatronic Systems (2004).

[38] L. Console, C. Picardi, and D. T. Dupré, "Temporal decision trees: Model-based diagnosis of dynamic systems on-board," 2003.

[39] B. Stein, "Model compilation and diagnosability of technical systems," in *Proceedings of the 3rd International Conference on Artificial Intelligence and Applications (AIA 03), Benalmádena, Spain*, 2003.

[40] W. L. P. Sperling, Ed., *Designing applications for an OSACA control*, 1997.

[41] F. Ubis, T. Kirkham, B. Matthews, J. Lastra, R. Harrison, V. Herrera, and A. Chowdrey, "The challenges along the road to the realisation of a factory automation lifecycle," in *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, Aug. 2008, pp. 559 –562.

[42] OPC Foundation, "OPC Unified Architecture Specification," Feb. 2009.

[43] Wolfgang Mahnke, Stefan-Helmut Leitner, Matthias Damm, *OPC Unified Architecture*. Springer-Verlag Berlin, 2009.

Optimierung der Rekuperation im Elektrofahrzeug durch Co-Simulationstechniken

S. Köhler†, J. Zimmermann†, H. Sahin†, O. Bringmann†
†FZI Karlsruhe, Germany
{koehler,zimmermann,hsahin,bringmann}@fzi.de

W. Rosenstiel†‡
‡University of Tuebingen, Germany
rosenstiel@informatik.uni-tuebingen.de

***Zusammenfassung*—Die begrenzte Energiekapazität aktueller Batterie-Elektrischer Fahrzeuge (BEV) führt zu neuen Anforderungen an Fahrerassistenzsysteme, mit deren Hilfe eine Reichweitensteigerung ermöglicht werden soll. Da dabei auch die Komplexität der Elektrischen/Elektronischen (E/E)-Architektur steigt, ist zur Reduktion der Entwicklungskosten eine möglichst frühe Analyse von Funktionalität und Zeitverhalten notwendig. In diesem Beitrag soll sowohl eine energieeffiziente Rekuperationsstrategie als auch eine Co-Simulationsumgebung zur Validierung dieser vorgestellt werden. Anhand eines Beispiels wird die Optimierung der Effizienz bestätigt sowie gezeigt, dass das Zeitverhalten der E/E-Architektur einen signifikanten Einfluss auf die Ergebnisse ausübt.**

I. Einleitung

Elektrofahrzeuge können, sofern sie mit Strom aus regenerativen Quellen geladen werden, einen großen Beitrag zur Reduktion des CO2-Ausstoßes leisten. Zum jetzigen Zeitpunkt können sich BEV dennoch nur sehr schwer durchsetzen. Neben den – im Vergleich zu Fahrzeugen mit Verbrennungsmotor – sehr hohen Kosten ist insbesondere die stark begrenzte Reichweite ein Hauptgrund für diese Tatsache. Trotz der hohen Effizienz des Elektromotors liegt die Reichweite deutlich unter der eines Verbrennerfahrzeugs. Dies liegt vor allem am Speichermedium für die genutzten Energie. Lithium-Ionen Akkumulatoren besitzen eine deutlich geringere Energiedichte als das Speichermedium Benzin (ca. Faktor 100). Dies führt selbst bei moderaten Reichweitenanprüchen dazu, dass Batterie-Packs im typischen Gewichtsbereich von bis zu 500 kg liegen. Aus diesem Grund ist es besonders wichtig, die zur Verfügung stehende Energie sehr effizient einzusetzen. Um dies zu erreichen soll der Fahrer durch Algorithmen – die auf Informationen von intelligenten Sensoren zurückgreifen – dabei unterstützt werden energieeffizient zu fahren.

Diese Algorithmen, aber auch Sicherheitssysteme innerhalb des Automobils wie Fahrdynamik-Regelung (ESP), Antiblockiersystem (ABS) oder Antischlupfregelung (ASR) können erst durch die Verwendung und Interaktion eingebetteter Systeme realisiert werden.

Die Landschaft existierender Simulationswerkzeuge ist durch eine große Heterogenität geprägt. Einige ermöglichen die Simulation von Elektrofahrzeugen, decken aber dabei sowohl das funktionale und zeitliche Verhalten der E/E-Architektur als auch das Verarbeiten von Sensordaten nur unzureichend genau ab. Zum Verifizieren von Maßnahmen zur Energieeffizienzsteigerung ist dieses Zusammenspiel und die Abbildung der gegenseitigen Abhängigkeiten von Bedeutung.

In den folgenden Abschnitten wird sowohl der Ansatz einer energieeffizienten Rekuperationsstrategie für ein BEV als auch eine Co-Simulationsumgebung zu deren Validierung unter Einbezug der E/E-Architektur vorgestellt. Dazu werden im Fahrzeugbereich gängige Simulations-Tools untereinander und mit SystemC gekoppelt.

Diese Arbeit wurde durch das FP7-Projekt OpEneR unter Fördernummer 285526 und durch das BMBF-Projekt RESCAR 2.0 unter Fördernummer 01M3195E unterstützt.

II. Stand der Technik

Zur Steigerung der Reichweite existieren für Verbrennerfahrzeuge Ansätze, die sich damit beschäftigen möglichst energieeffizientes Fahren zu ermöglichen. Ein Ansatz ist hierbei das sogenannte Freilaufen, bei dem die kinetische Energie des Fahrzeugs vollständig zu dessen Fortbewegung während des Abbremsvorgangs genutzt wird [1] [2]. Bei Hybrid- (HEV) und Elektrofahrzeugen besteht zusätzlich die Möglichkeit regenerativ zu bremsen. Bei der sogenannten Rekuperation erzeugt der Elektromotor im Generatorbetrieb ein Drehmoment, das zum Abbremsen genutzt wird. Ein Teil der dabei reduzierten kinetischen Energie des Fahrzeugs wird in elektrische Energie gewandelt und kann in die Batterie gespeist werden. Bei diesem Abbremsvorgang besteht die Möglichkeit, den Motor stets in einem möglichst effizienten Arbeitspunkt zu betreiben. Nach unserem derzeitigen Kenntnisstand gibt es hierzu noch keine Arbeiten. Bei einem HEV wird beim Betrieb der Antriebselemente das Prinzip der Lastpunktverschiebung angewandt. Ein Elektromotor arbeitet hier zusätzlich, um den Betriebspunkt des Verbrennungsmotors in einen effizienteren Drehmoment-Bereich zu verschieben. Wirkt der Elektromotor dabei als Generator, wird die erzeugte Energie zurück in die Batterie gespeist [3]. Auch der Einsatz von intelligenter Sensorik wird in diversen Projekten genutzt, um eine möglichst effiziente Fahrweise zu erreichen. Anwendung findet dies im Rahmen GPS gestützter ACC-Systeme bei Verbrennerfahrzeugen [4] [5].

Mit CosiMate [6] existiert eine Co-Simulationsumgebung, die es erlaubt, heterogene Simulationen (z.B. Matlab/Simulink, Modelsim, IBM Rational), aber auch C/C++-Code und damit SystemC wechselwirkend zu simulieren. Eine weitere Co-Simulationsumgebung für verbreitete Simulations-Tools ist TISC [7], welche ebenfalls eine Synchronisierung der gekoppelten Simulationsplattformen vornimmt und geteilte Variablen in einem gemeinsamen Speicherbereich ablegt. TISC ist dabei so aufgebaut, dass gekoppelte Simulations-Tools auf unterschiedlichen Rechnern ausgeführt werden können (Server-Client Prinzip). Gerade im Fahrzeugbereich sollte auch das Projekt MODELISAR [8] nicht unerwähnt bleiben. Mit seinem Functional Mockup Interface kann MODELISAR die zuvor genannten Co-Simulationsplattformen kombinieren. Nutzt man dies, so kann man die im Beitrag genutzten Simulations-Tools gemeinsam betreiben. Da dies jedoch über insgesamt 3 Co-Simulationsumgebungen geschehen würde, ist von einem unnötigen Overhead auszugehen.

III. Systemmodell

Im Rahmen der vorliegenden Arbeit wird ein Optimierungs-Algorithmus zur Reichweitensteigerung eines Elektrofahrzeugs betrachtet. Um die hierzu notwendige Erhöhung der Effizienz zu erreichen, soll auf umfassende Sensorik wie GPS, Radar und kamerabasierte Verkehrszeichenerkennung zurückgegriffen werden. Die dadurch gewonnenen a-priori Informationen können genutzt werden, um die Komponenten des Antriebsstrangs in einem optimalen Arbeitspunkt zu betreiben. Zur bestmöglichen Reaktion auf die durch die Sensorik erkannten Situationen muss das Verhalten des Fahrzeugs bekannt sein. Hierbei sind Detaillierungsgrade vom Zweispur- über Einspurmodell [9] bis zur Punktmasse möglich. Da bis zum jetzigen Zeitpunkt noch keine Untersuchung der Fahrzeugstabilität stattfindet und die Masseverteilung des Fahrzeugs derzeit nicht ermittelt ist, wird sie als gleichverteilt angesehen. Unter diesen Rahmenbedingungen ergibt sich das Einspurmodell als ausreichend.

A. Fahrzeugdynamik

Es wird davon ausgegangen, dass das vorgestellte Assistenzsystem nur in sicherheitsunkritischen Situationen zum Einsatz kommt, deshalb beschränkt sich die Modellierung auf das Longitudinalmodell des Fahrzeugs. Dabei wirken die Fahrwiderstände, die von Fahrzeug (Masse m, Rollreibungskoeffizient c_{rr}, Strömungswiderstandskoeffizient c_w, Bezugsfläche A_w,), Umwelt (Luftdichte ρ, Gravitationsbeschleunigung g) sowie Streckenprofil (Steigung α, Kurvenradius R) abhängen. Die Kräfte sollen im resultierenden Fahrwiderstand F_r aufsummiert dargestellt werden. Unter Berücksichtigung der durch den Antriebsstrang an den Reifen wirkenden Gesamtkraft F_t ergibt sich durch das zweite Newtonsche Axiom

$$m\, a_x = F_t - F_r. \tag{1}$$

Die Integration der Differentialgleichung

$$\frac{\mathrm{d}v_x}{\mathrm{d}s} = \frac{F_t - F_r}{m\, v_x} \tag{2}$$

ermöglicht das Ermitteln von Position und Geschwindigkeit des Fahrzeugs über die Strecke.

B. Fahrzeugkomponenten

Das den Betrachtungen zugrundeliegende Fahrzeug ist schematisch in der Abbildung 1 dargestellt.

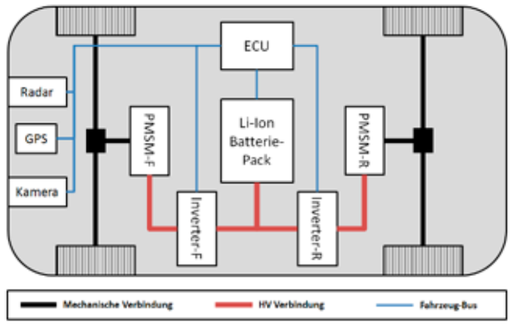

Abb. 1. Schematische Darstellung der Fahrzeugkomponenten

Als Antrieb und Generator dient an Vorder- und Hinterachse jeweils ein Permanentmagnet-Synchronmotor (PMSM) mit integriertem (konstantem) Getriebe. Die fusionierte Effizienzcharakteristik der Motoren zeigt Abbildung 2 über Drehzahl und Drehmoment aufgetragen.

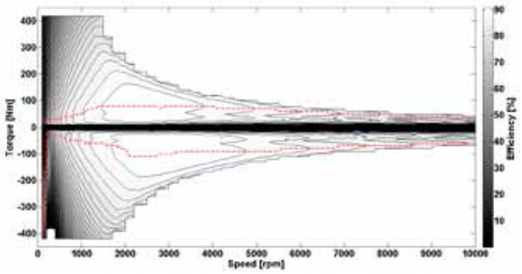

Abb. 2. Effizienzcharakteristik der PMSM

Weiterhin von großer Relevanz für die Gesamtperformanz des Antriebsstrangs ist die Effizienz der Batterie und des Inverters. Für diese und andere bisher nicht charakterisierte Komponenten (Getriebe etc.) soll eine akkumulierte, konstante Effizienz η_b angenommen werden. Wie im vorigen Abschnitt erwähnt, erfolgt die Modellierung der Fahrzeugdynamik durch das Einspurmodell. Damit gilt mit dem Getriebe des PMSM eine Übersetzung des Motordrehmoments auf die Reifen mit festem Übersetzungsverhältnis i[1]. Die Leistung P [W] ist proportional zum Drehmoment M [Nm] und der Drehzahl n [rpm] bzw. ω [rad/s]

$$P = \omega\, M = \frac{2\pi n}{60}\, M. \tag{3}$$

Für das an den Reifen anliegende Drehmoment M_t gilt dann bei Motorleistung P_m und Motordrehzahl n_m

$$M_t = i\, \frac{60 P_m}{2\pi n_m}. \tag{4}$$

Für die Effizienzbetrachtung unterscheiden sich die Fälle des positiven und negativen Drehmoments. Sollen die Motoren ein positives Drehmoment liefern, gilt für die Leistung, die der Batterie entnommen wird

$$P_{b,out} = \frac{P_m}{\eta_m\, \eta_b}. \tag{5}$$

Wird der Motor zum rekuperativen Bremsen genutzt, so wird die Leistung

$$P_{b,in} = \eta_m\, \eta_b\, P_m \tag{6}$$

in die Batterie gespeist.

C. E/E-Architektur

In Fahrzeugsystemen sind diverse datenabhängige Faktoren vorhanden (z.B. Buslatenzen, Funktionen, Leistungsaufnahme ECU). Im vorliegenden Dokument wird die Betrachtung auf die Berechnungszeiten der Verkehrszeichenerkennung begrenzt, da diese zu einer – für die im nachfolgenden Kapitel vorgestellte Rekuperationsstrategie – relevanten Verzögerung führen kann. Tabelle I zeigt die Berechnung der Verkehrszeichenerkennung für einen Beispielframe auf unterschiedlichen E/E-Architekturen und Implementierungen.

Tabelle I
Beispielvergleich verschiedener E/E-Architekturen

Alternativkonfigurationen Verkehrszeichenerkennung	Latenz
Software-Implementierung (1 Leon-Kerne)	1631ms
Software-Implementierung (2 Leon-Kerne)	901ms
Software-Implementierung (4 Leon-Kerne)	594ms
Hardware, Software-Implementierung (2 Leon-Kerne)	509ms

IV. Simulationsumgebung

A. Fahrzeugsimulation

Zur Simulation der Energiebilanz eines Elektrofahrzeugs werden die zeitgesteuerten Simulationsplattformen Matlab/Simulink und AVL Cruise kombiniert. Während Matlab/Simulink die Standardsoftware zur Modellierung von Fahrzeugkomponenten bei vielen OEMs ist, steht mit AVL Cruise ein auf Antriebsstrang-Komponenten spezialisiertes Simulations-Tool zur Verfügung. Beide Simulationsplattformen können Modelle der jeweils anderen über mitgelieferte Schnittstellen einbinden. Die gewählte Kombination ermöglicht dadurch eine möglichst aufwandsarme Modellbildung für Fahrzeuge, bietet jedoch gleichzeitig große Flexibilität.

B. Simulation der E/E-Architektur

Seit geraumer Zeit hat sich sowohl im industriellen als auch im Forschungsumfeld die Systemmodellierungssprache SystemC [10] [11] zu einem De-facto-Standard zur Modellierung und Simulation eingebetteter Systeme auf höheren Abstraktionsebenen entwickelt. SystemC ist eine statische C++-Bibliothek zur diskreten, ereignisgetriebenen Simulation von Hardware-/Software-Systemen, die zur Beherrschung der Modellkomplexität eine komponentenbasierte Modellierungsmethodik und einhergehend eine strikte Trennung zwischen Berechnung und Kommunikation verfolgt. Der SystemC IEEE Standard [12] definiert sowohl die Sprachkonzepte der C++-Bibliothek als auch den Simulationskern, der eine nebenläufige Ausführung der Systemfunktionalität und die Realisierung unterschiedlicher Abstraktionsebenen erlaubt. Ein SystemC-Modell ist dabei in *Module* unterteilt, die zur Repräsentation eines hierarchischen Verhaltens wiederum andere Module enthalten können. Weiterhin wird die Modul-Funktionalität in *Prozessen* dargestellt, die über *Ports* mit anderen Modulen interagieren können. Prozesse werden durch den Simulationskern quasi-parallel ausgeführt, wobei ein Prozess aktiv bei seiner Abarbeitung so lange unterbrochen werden kann, bis eine Reaktivierung ausgeführt wird. Kommunikationen werden in Schnittstellen, mit denen die jeweiligen Modul-Ports typisiert werden, deklariert und durch ein *Channel*-Konzept implementiert. Die darin enthaltenen Kommunikationsprimitive können sowohl eine Kommunikation basierend auf Signalen als auch komplexe Kommunikationsprotokolle umfassen.

Weiterhin existieren mehrere Ansätze zur modellgetriebenen Entwicklung von Hardware-/Software-Systemen [13] [14] – inklusive der automatischen Generierung einer SystemC-Simulationsplattform und Einbindung von zeitgenauen Busmodellen wie CAN, Flexray, usw. Dadurch werden die Entwicklungszeiten einer E/E-Architektur-Simulation in der Regel drastisch reduziert. Aus Platzgründen werden diese Ansätze hier nicht im Detail dargestellt, sondern stattdessen auf die zitierten Quellen verwiesen.

V. Rekuperationsstrategien

Auf Basis intelligenter Sensorik wie GPS zzgl. Kartendaten sowie Bilderkennungsverfahren kann für einen gegebenen Streckenabschnitt ein optimales Geschwindigkeitsprofil berechnet werden, welches sich an Sicherheits- und/oder Verbrauchskriterien orientiert. An Stellen bei denen dieses Geschwindigkeitsprofil die Reduktion der Fahrzeuggeschwindigkeit vorsieht, soll dies möglichst energieeffizient geschehen. Damit das erreicht wird, soll keine Energie durch die mechanischen Bremsen in unbrauchbare Wärme umgewandelt werden. Es wurde gezeigt, dass die effizienteste Methode das Freilaufen des Fahrzeugs ist [1]. Hierbei wird die gesamte kinetische Energie des Fahrzeugs genutzt, um den Fahrwiderständen entgegenzuwirken. Durch die geringen Beschleunigungen führt dies allerdings zu langen

[1] Wegen des Einspurmodells wird das Differenzialgetriebe vernachlässigt.

Bremswegen, wie an der Trajektorie *Freewheeling* in Abbildung 3 zu sehen ist.

Durch regeneratives Bremsen kann bei Abbremsvorgängen die überschüssige kinetische Energie des Fahrzeugs durch die PMSM rückgewonnen werden. Die Strategie sieht deshalb vor, dem Fahrer mitzuteilen, ab welcher Distanz allein durch Freilaufen die gewünschte Geschwindigkeit zur Zielposition erreicht werden kann. Reagiert der Fahrer zu diesem Zeitpunkt nicht, so muss das verlustbehaftete regenerative Bremsen eingesetzt werden. Da die PMSM wie im vorigen Abschnitt beschrieben einer gewissen Effizienz-Charakteristik unterliegen, gilt es eine möglichst effizienz-optimale Drehmoment-Trajektorie zu ermitteln. Geht man von einer hinreichend großen Distanz aus, so kann die ideale Trajektorie direkt aus der Effizienz-Charakteristik der PMSM in Abbildung 2 ausgelesen werden. Sie wird durch die eingezeichnete gestrichelte Linie markiert und beschreibt für jede Drehzahl jenes Drehmoment, bei dem die Motorenkombination mit maximaler Effizienz arbeitet. Mit den Gleichungen aus Abschnitt III kann hierzu unter Berücksichtigung der Sensordaten und den Informationen aus der digitalen Karte der Startpunkt für ein Freilaufen und eine Rekuperation bei maximaler Effizienz berechnet werden.

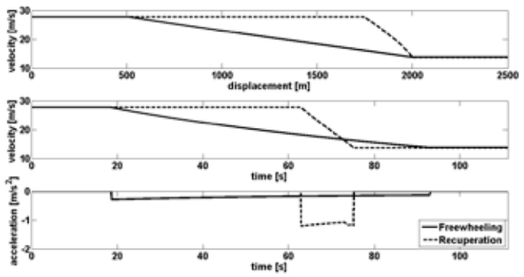

Abb. 3. Trajektorien für Freilaufen und Rekuperation bei max. Effizienz

Reagiert der Fahrer nicht rechtzeitig zur Freilauf-Empfehlung, sondern im nachfolgenden Streckenabschnitt – also vor dem Startpunkt der idealen Rekuperations-Trajektorie – wird eine Kombination beider angewandt. Das Fahrzeug wird dann freilaufen, bis die Rekuperations-Trajektorie geschnitten wird und ab diesem Punkt mit maximaler Effizienz rekuperieren (vgl. Abb. 4).

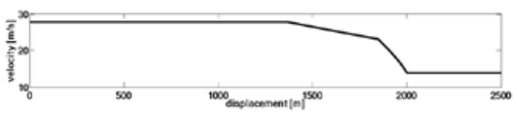

Abb. 4. Kombination Freilaufen und Rekuperation bei max. Effizienz

Wird allerdings zu spät reagiert und kann die Fahrzeuggeschwindigkeit mit dem Drehmoment der maximalen Effizienz nicht auf den geforderten Wert reduziert werden, so ist die zu wählende Drehmoment-Trajektorie nicht eindeutig. Es ergibt sich ein Problem der optimalen Steuerung mit dem Steuervektor M_t und den zu minimierenden Kosten

$$\begin{aligned} \min_{M_t} L &= \int_{s_0}^{s_f} \left[1 - \eta(n, M_t)\right] \frac{|M_t|}{r} \mathrm{d}s \\ &\text{s.t.} \\ \frac{\mathrm{d}v_x}{\mathrm{d}s} &= \frac{M_t/r - F_r}{m\, v_x} \\ v(s_0) &= v_0 \\ v(s_f) &= v_f \\ M_{t,max}(n) \leq M_t &\leq 0 \end{aligned} \tag{7}$$

VI. Co-Simulation

In diesem Abschnitt soll eine Co-Simulationsplattform vorgestellt werden, mit der energieeffiziente Fahr- und Betriebsstrategien im Zusammenspiel einer gesamtheitlichen Repräsentation des Fahrzeugs simuliert und bewertet werden können. Dies ist sowohl im Hinblick auf eine umfassende Energieverbrauchsanalyse als auch zur Verifikation des funktionalen Verhaltens und der damit einhergehenden Prüfung der funktionellen Sicherheit unumgänglich.

A. Aufbau der Co-Simulationsplattform

Wie bereits zuvor erwähnt, zeichnen sich aktuelle Simulationsumgebungen durch eine große Heterogenität aus, sodass in der Regel eine spezifische Simulation für die jeweilige Anwendungsdomäne existiert. Um eine Interaktion dieser verschiedenen Simulationsplattformen zu ermöglichen, wurde eine generische Co-Simulationsplattform, wie in Abbildung 5 dargestellt, entwickelt.

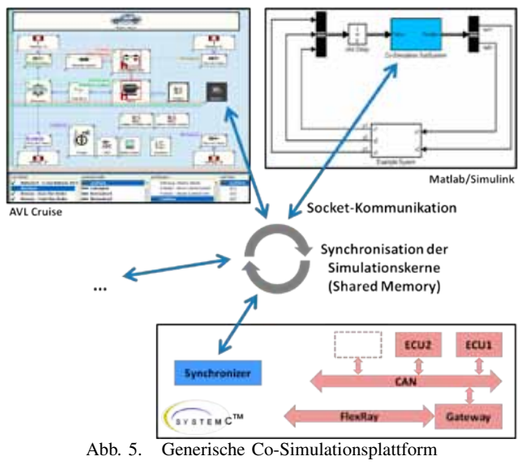

Abb. 5. Generische Co-Simulationsplattform

Eine Hauptkomponente zur Synchronisation der Simulationskerne ist die Kommunikation über einen gemeinsamen Speicherbereich, in dem die jeweils benötigten Simulationswerte abgelegt werden, um sie den Simulationsumgebungen nach der Synchronisation zum richtigen Zeitpunkt zur Verfügung stellen zu können. Diese Komponente ist auch für die Synchronisation zuständig, die im nachfolgenden Abschnitt VI-B beschrieben wird.

Die Kommunikation zwischen gemeinsamem Speicher und den jeweiligen Simulationsumgebungen erfolgt über Socket-Verbindungen, die im ISO-Standard plattformunabhängig spezifiziert sind. Somit ist es auch möglich, die Simulationsumgebungen auf unterschiedlichen Plattformen und auf unterschiedlichen Simulationsrechnern parallel auszuführen.

B. Synchronisation

Um eine hohe Simulationsgeschwindigkeit zu garantieren, wird eine Synchronisation der Simulationskerne nur dann durchgeführt, wenn eine Interaktion zwischen den einzelnen Simulationsumgebungen unbedingt notwendig ist. Weiterhin werden die Simulationsumgebungen mit einer variablen Schrittweite durchgeführt, sodass auf eine periodische Synchronisation nach der kleinsten gemeinsamen Schrittweite verzichtet werden kann.

1) Bestimmung des nächsten Simulationsschritts: Entscheidend für die Möglichkeit der Co-Simulation durch Synchronisierung der Simulationskerne ist die Fähigkeit, den nächsten Simulationsschritts akkurat vorhersagen zu können. Da der SystemC-Simulationskern ereignisbasiert arbeitet, existiert innerhalb des Simulationskerns eine Liste mit ablaufbereiten Prozessen und deren nächste Aktivierung. Nach Beendigung eines Simulationszyklus kann also die nächste Simulationsschrittweite durch einfache Differenzbildung von nächster Simulationsschrittweite und aktueller Simulationszeit bestimmt werden. Der Vergleich mit den aktuellen Simulationszeiten der zeitgesteuerten Simulationsumgebungen ermöglicht die Bestimmung des als nächstes auszuführenden Simulationsprozesses.

2) Allgemeiner Ablauf der Co-Simulation: Der genaue Ablauf der Co-Simulation ist in Abbildung 6 dargestellt. Zu berücksichtigen ist hierbei, dass aus Übersichtsgründen die direkte Verbindung zwischen SystemC und AVL Cruise *oder* Matlab/Simulink dargestellt ist. Sind mehrere zeitgesteuerte Simulationsumgebungen vorhanden, so entstehen zusätzliche Abhängigkeiten.

Wie bereits erwähnt, wird eine Synchronisation der einzelnen Simulationen nur durchgeführt, wenn sie unbedingt notwendig ist, d.h. Simulationsumgebungen interagieren. Im Allgemeinen (in Abbildung 6 durch Fall 1 gekennzeichnet) wird eine Synchronisation dann durchgeführt, wenn der nächste Simulationszeitpunkt der ereignisbasierten Simulation über die aktuelle Simulationszeit der zeitgesteuerten Simulation hinausgehen würde. Die Simulation kann also auf der ereignisbasierten oder der zeitgesteuerten Simulationsumgebung aus mehreren Simulationsschritten bestehen, ohne dass dazwischen eine Synchronisation stattfinden muss.

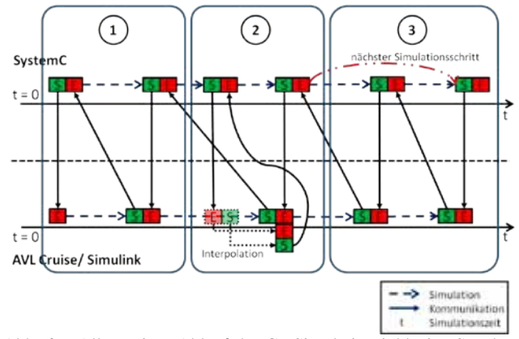

Abb. 6. Allgemeiner Ablauf der Co-Simulation inklusive Synchronisation der Simulationskerne

Durch diesen Ablauf kann es allerdings vorkommen, dass die ereignisbasierte Simulationsumgebung zur Durchführung der eigenen Simulation zum Zeitpunkt t_h auf Simulationswerte zurückgreifen muss, die zwischen zwei Simulationsschritten t_{i-1} und t_i der zeitgesteuerten Simulation liegen (Fall 2). In diesem Fall wird der zum Zeitpunkt t_h ($t_{i-1} \leq t_h \leq t_i$) geltende Wert durch lineare Interpolation nach Gleichung 8 ermittelt.

$$wert_{t_h} = wert_{t_{i-1}} + \frac{wert_{t_i} - wert_{t_{i-1}}}{t_i - t_{i-1}} \left(t_h - t_{i-1}\right) \tag{8}$$

Hierbei ist zu erwähnen, dass die Genauigkeit der durch die Interpolation errechneten Werte sehr hoch ist, da in zeitgesteuerten Simulationsumgebungen mit variabler Schrittweite der Zeitschritt so klein gewählt wird, dass von einer linearen Verbindung ausgegangen werden kann.

Wird bei einer zeitgesteuerten Simulation ein Ereignis ausgelöst (z.B. Erreichen von definierten Wegpunkten auf der simulierten Teststrecke, Signalisierung der ESP-Sensorik, usw.), muss dieses unter Umständen der ereignisbasierten Simulationsumgebung mitgeteilt werden. Der nächste Zeitschritt wird daraufhin von der ursprünglich angenommenen auf die neue Schrittweite modifiziert. Für die Berechnung der aktuellen Simulationszeit bedeutet dies, dass nur die daraus resultierende Differenz in der Zeit voranschreitet und somit ein zusätzlicher Zeitschritt eingeführt wird (Fall 3 in Abbildung 6).

VII. Ergebnisse

Das mit der Optimierung aus Abschnitt V ausgestattete Fahrzeugmodell wurde im ersten Schritt ohne die Co-Simulation betrachtet. Hierbei ergeben sich für eine Geschwindigkeitsreduktion von $v_0 = 100\,\mathrm{km/h}$ auf $v_f = 50\,\mathrm{km/h}$ Verbesserungen der Gesamteffizienz der PMSM gegenüber den in Abbildung 7 ebenfalls aufgetragenen alternativen Trajektorien.

Die Alternative *Time Efficient* ergibt sich unter Berücksichtigung der Zeit. Es wird zwar rein rekuperativ, jedoch mit möglichst hohem Drehmoment abgebremst.

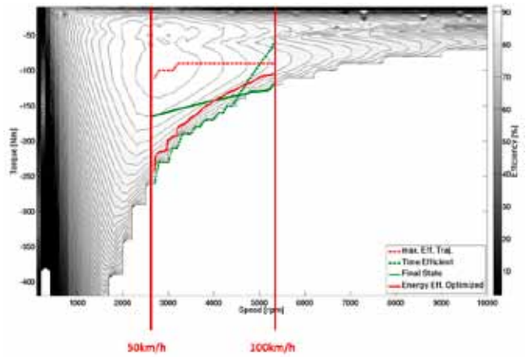

Abb. 7. Vergleich zulässiger Rekuperations-Trajektorien

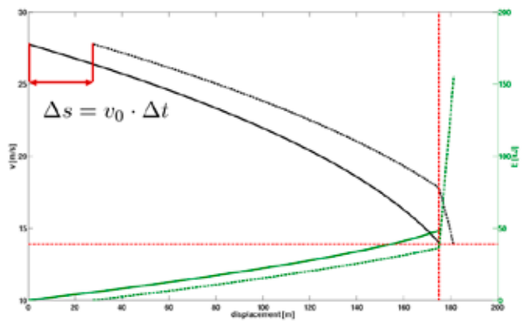

Abb. 8. Darstellung des Energieverlusts durch Verzögerung

Für Alternative *Final State* werden lediglich die Rahmenbedingungen – also ausreichende Geschwindigkeitsreduktion – miteinbezogen. Tabelle II zeigt die Verluste E_{loss} bei Nutzung der unterschiedlichen Trajektorien, ΔE die zusätzlich verlorene Energie bezogen auf die maximal erreichbare Effizienz. Weiterhin ist die Gesamteffizienz η_{tot} des Rekuperationsvorgangs angegeben.

Tabelle II
ERGEBNISSE DER SIMULATION OHNE BERÜCKSICHTIGUNG DER E/E-ARCHITEKTUR

Alternative	E_{loss} [kJ]	ΔE [kJ]	η_{tot} [%]
Max. Eff.	44.77	0.00	92.1
Energy Eff. Optimized.	53.21	8.44	89.1
Final State	58.00	13.24	88.2
Time Efficient	61.64	16.87	87.3

Nachdem die erfolgreiche Effizienzsteigerung durch die Optimierung der Rekuperation gezeigt wurde, wird nachfolgend die Zeitberücksichtigung durch die Co-Simulation mit eingebracht. Dafür wurde aus Tabelle I die mittlere Leistungsklasse mit einer Verzögerung von $\Delta t \approx 1\,\mathrm{s}$ gewählt. Das Ergebnis soll aus Platzgründen nur für den Fall der Rekuperation mit optimierter Drehmoment-Trajektorie dargestellt werden. Der Vergleich der zeitbehafteten (gestrichelt) mit der verzögerungsfreien Simulation ist in Abbildung 8 zu sehen. Im Diagramm ist sowohl Fahrzeuggeschwindigkeit als auch die während des Bremsvorgangs verlorene Energie dargestellt. Da die Zeitverzögerung zu einem fehlerhaften Startpunkt der Rekuperation führt, muss der Fahrer eingreifen. Es wird nicht mehr rein rekuperativ gebremst, sondern kinetische Energie durch die Bremsbacken in Wärme umgewandelt. Man erkennt, dass mit mehr als $150\,\mathrm{kJ}$ ca. dreimal soviel Energie unbrauchbar wird als bei verzögerungsfreiem Szenario.

Es zeigt sich dadurch, dass die Betrachtung der E/E-Architektur auch für den Bereich der Energieeffizienz relevant ist. Vielmehr noch ist dies der Fall, wenn es um Algorithmen im sicherheitskritischen Bereich handelt. Auch hierfür kann die Co-Simulation eingesetzt werden und Schwachstellen in der Konfiguration der E/E-Architektur in einer möglichst frühen Entwicklungsphase aufdecken.

VIII. DANKSAGUNG

Diese Arbeit wurde mit Mitteln des Ministeriums für Wissenschaft, Forschung und Kunst Baden-Württemberg im Rahmen des kooperativen Promotionskollegs EAES gefördert.

LITERATUR

[1] T. Bar, R. Kohlhaas, J. Zollner, and K. Scholl, "Anticipatory driving assistance for energy efficient driving," in *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*, 29 2011-july 1 2011, pp. 1 –6.

[2] D. Nienhueser, T. Baer, R. Kohlhaas, T. Schamm, J. Zimmermann, T. Gumpp, M. Strand, O. Bringmann, and J. Zoellner, "Energy efficient driving and operation strategie based on situation awareness and reasoning," in *it - Methods and Applications of Informatics and Information Technology*, 2012.

[3] H. Woehl-Bruhn, *Synchronmaschine mit eingebetteten Magneten und neuartiger variabler Erregung fuer Hybridantriebe*. Cuvillier Verlag Goettingen, 2009.

[4] E. Hellstroem, "Look-ahead control of heavy vehicles," Ph.D. dissertation, Linkoeping studies in science and technology, 2010.

[5] autoplenum.de, "Porsche ACC InnoDrive," http://www.porsche.com.

[6] ChiasTek, "CosiMate," http://www.chiastek.com/.

[7] TLK-Thermo GmbH, "TISC Suite," http://www.tlk-thermo.com/.

[8] "Modelisar," http://www.modelisar.com.

[9] B. Heissing, M. Ersoy, and S. Gies, *Fahrwerkhandbuch*. Vieweg+Teubner Verlag, 2011.

[10] Open SystemC Initiative (OSCI), "SystemC," http://www.systemc.org.

[11] T. Groetker, S. Liao, G. Martin, and S. Swan, *System Design with SystemC*. Springer, 2002.

[12] *IEEE Standard SystemC Language Reference Manual*, IEEE Computer Society, 2006.

[13] J. Zimmermann, M. Pressler, A. Viehl, O. Bringmann, and W. Rosenstiel, "Model-based virtual prototyping for early automotive software systems evaluation," in *Proceedings of the 1st Workshop on Model Based Engineering for Embedded Systems Design at DATE*, 2010.

[14] J. Zimmermann, M. Kuester, O. Bringmann, and W. Rosenstiel, "Model-based generation of a fast and accurate virtual execution platform for software-intensive real-time embedded systems," in *Proceedings of the 17th Workshop on Synthesis And System Integration of Mixed Information Technologies*, 2012.

Digitaler Hard- und Softwareentwurf für Cyber-Physical Systems

Jitter Considerations for Worst-Case Performance Generation in Digital Controller Design

Tobias Bund, Steffen Moser, Steffen Kollmann and Frank Slomka
Institute of Embedded Systems/Real-Time Systems
Ulm University
Email: {tobias.bund | steffen.moser | steffen.kollmann | frank.slomka}@uni-ulm.de

***Abstract*—One domain of cyber-physical systems are distributed control systems. The requirements for a control system are correctness, stability and control performance. One issue when designing such a controller is that the timing behavior of the system architecture has a direct impact on the control performance. In this paper, we present an approach which allows us to consider the timing during the controller design flow. Further we propose an approach which allows us to include the maximal occurring jitter in a control loop. Based on this, we propose a new co-design method of real-time and control systems. The controller is designed based on the results of the average delay estimated by the real-time simulation to reach best performance in most cases. It is then verified with the worst- and best-case end-to-end delay bounds calculated by real-time analysis methods. In the last step, the worst control performance regarding the jitter is evaluated by constructing a bad-delay sequence. This approach combines real-time simulation and analysis for best controller design and verification under worst-case timing.**

I. Introduction

As the requirements of embedded systems are increasing, we can see more and more complex architectures consisting of many inter-networked electronic control units (ECUs). Taking the automotive industry for example, we can find up to seventy ECUs in a modern upper-class vehicle. Each ECU consists of various subcomponents like general and application-specific processing units, communication networks and memory architectures. An ECU can be a host for multiple tasks which bring the challenge of finding a suitable task schedule. All these degrees of freedom result in a large design space whereof the system engineer has to choose a solution which must fulfill all requirements that have been specified for the system.

One possibility to avoid problems when dealing with complex architectures and timing issues are to design an oversized platform. In oversized platforms, consisting of more communication and computation resources, there is in general less influence among tasks regarding their timing behavior. In most embedded systems, especially in the automotive domain, there is a trend in reducing costs and energy by resource sharing. This results in a more efficient usage of the available resources and bandwidths, but introduces a non-determinism in the response times of tasks. Additionally, there are future trends in embedded systems like multi-core architectures that lead to timing effects, caused by task migration, that need to be considered.

One commonly required task in embedded systems is controlling a physical system. This results in real-time constraints, which, if violated, can not only have a negative impact on the control performance but may also let the system fails completely. In embedded systems it is not uncommon that tasks of a controller, like gathering sensors values, calculating the actuating signal and driving the actuators are distributed over more than one ECU. This is, for example, the case if one sensor value is needed for more than one technical process. For safety-critical systems it is necessary to know that the system behaves at any time as classified. For example, in a car suspension control system a poor controller performance has direct influence on car maneuverability and therefore car stability.

A goal in the design of distributed closed-loop control systems is to achieve that the system remains stable and stays in the desired performance bounds for all possible disturbances and timing effects. Hence it is necessary to consider the underlying architecture in the controller design. So the relevant timing effects, the average and worst-case delays and the jitter of the delay have to be extracted. In the literature the jitter is mainly described statistically, but in this paper, a special attention is devoted the worst-case jitter behavior.

Finally a new controller design flow is presented that combines the classical controller design with real-time simulation and analysis, extended with worst-case jitter behavior. The main idea behind this design flow is to design the controller with the knowledge of the average delay that appears in the control loop. In the next step, the design is verified against the worst-case conditions.

The structure of the paper is as follow: In the next section we present the problem of jitter in a closed-loop control system and the work that has been done in this field of research. Section three describes our system model, containing the controller and the plant as well as the model, needed for real-time analysis. Based on the results of real-time analysis and simulation, we define the model of timing effects, that are relevant for our method. The next section discusses the influence of jitter on closed-loop control systems. In the fifth section, we present a design flow for an embedded controller by exploiting the described jitter transformation. Section six illustrates our approach in an example, followed by a conclusion.

II. Problem Formulation and Related Work

In complex cyber-physical systems it is very common to have tasks which share resources for computation as well as for communication. This creates dependencies among tasks which may lead to delays when tasks cannot be executed or messages cannot be sent while the resources are busy. These delays can occur in a constant and in a varying manner during the run-time of the system. Differences among two consecutive time-varying delays are called jitter.

In the following, we will mainly focus on the jitter effects on control loops. We assume that the control loop is distributed over the system and uses shared resources which lead to both, constant and time-varying delays in the communication with the controller. In controller design, two main criteria are important: stability and control performance. We will focus on the control performance in the following.

Control performance can be defined in several ways, the most common definitions are:

- as maximum overshoot of the step response
- as phase margin
- as integral of absolute error (IAE) or integral of squared error (ISE)
- as settling time in the step response

The control performance can be seen as a meter to evaluate the quality of a controller. While still being stable, a controller can show a poor performance which means, for example, too much stress for a mechanical system, too high energy consumption or even a safety risk. Both, stability and performance are influenced by the constant delay times and the jitter which occurs in the system. In many applications a poor control performance is not acceptable, so there is a strong need for a method which allows to derive the worst performance that can be expected for a given dead time and jitter.

A control loop which suffers only from a constant delay can be handled much easier than jitter. A constant delay, also called dead time in the domain of control theory, leads to a degradation of the control performance, but it can easily be considered in the design step of a controller because the system remains time-invariant. Therefore, the methods for controller design remain valid. Thus, there exist some well-founded approaches that offer possibilities to integrate the constant delay into stability and performance considerations, when developing a controller application. One approache which allow to cope with dead time is the Nyquist stability criterion [1], a simple graphical stability criterion, where the gain and phase of a frequency response of the open control loop are plotted. Out of this Nyquist plot, a phase margin can be read that provides an amount of dead time which can be added to a closed-loop control system before it becomes unstable.

Based on the Nyquist criterion and the small gain theory, Kao and Lincoln defined the jitter margin which is a simple stability criterion for systems with time-varying delays in [2]. Cervin et al. further improved this stability criterion in [3] by splitting the time-varying delay in a constant delay and jitter, resulting in a less pessimistic test. But it is, as stated above, not always sufficient to know that the system behaves stable in all occurrences of dead time and jitter.

An approach to consider the effects of jitter to the control performance is presented by Lincoln and Cervin in [4]. They developed the Matlab toolbox Jitterbug which is based on the ISE performance criterion. The closed-loop digital control system with delays is modeled as a Markov Jump Linear System. The evolution of this system with a time-grain leads to a corresponding covariance of the state, from which the costs can be calculated. Due to the stochastic manner, this approach is not sufficient for deriving the worst-case performance and therefore does not help to analyze the signal behavior in the worst-case scenario.

Alternative approaches are jitter compensation techniques as published by Lincoln in [5], where additional information in the form of time stamps has to be propagated through the system, leading to a higher utilization. Another approach is the controller parameter adaption, based on the estimated jitter. A method to avoid jitter effects in a control loop is the usage of a time-triggered communication method, as used by Goswami et al in [6], where sample and actuate only act on constant sample times with a delay of one sampling period. For systems with limited resources, where the sensor value can not be sampled at high rates, a delay of one sample time can lead to unacceptable control performances. Additionally, a clock is needed in the actuator that needs to be synchronized.

The above described approaches do not allow to derive the jitter-caused worst-case performance of the controller or they require additional effort to compensate the jitter. Our approach is to construct a delay sequence for given bounds with knowledge of the control signal sequence. This "bad-delay-sequence" causes the worst possible control performance. In the next section, we will introduce a system model which is the basis for a method that will provide a solution for this problem.

III. System Model

To analyze the performance, behavior and stability of a closed-loop control system, we need to model the system. A common platform for control modeling and simulation is Matlab/Simulink. The plant that needs to be controlled consists of continuous blocks, which represent a differential equation. Another way to model a physical system would be in the form of a transfer function in laplace domain.

The architecture surrounding the plant is modeled by blocks out of the SimEvents [7] toolbox. This is due to the event-based simulation engine SimEvents provides, following the discrete event-based behavior of networking elements and computation units. SimEvents blocks contain queuing, routing and serving elements to model delays according to the effects in a distributed architecture or in a multi-core environment. The controller itself is built out of discrete blocks, as it is implemented on a computation unit. In sum, the described model guarantees a flexible and transparent way to simulate

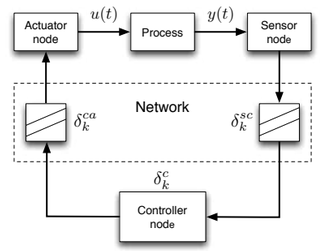

Fig. 1. Model of a distributed closed-loop control system with network delays (δ_k^{ca}) and (δ_k^{sc}) and computational delay (δ_k^c) based on [10].

the system with maximum relation to real behavior. From this model, we can derive a graph, representing the system model. A more detailed description about this approach can be found in [8].

Real-time systems are modeled, including tasks, holding a best-case and a worst-case execution time. These tasks are mapped to a resource, where their execution order is determined by their priority. A resource can be a processor or a network with a limited capacity. The tasks are linked to event chains according to their logical execution order and stimulated by sources, specifying an amount of events per time units as shown in [9]. It is also possible to generate a graph representation of the real-time model (a task graph) as given in [8].

In [8], also a way is described to link a functional Matlab/Simulink model to the real-time analysis by using a bijective mapping. This is sufficient in most cases, as it regards only constant delays in the control loop. As we will see in this paper, we have to expand this to simulation results and to the amount of jitter that can occur.

Figure 1 shows an example of a closed-loop control system with several sources of delays (δ_k), where (δ_k) may vary within every k-th sample. In the presented system model, the sensor samples the data ($y(t)$) from the plant with a constant period (h).

Thereby, we assume an ideal sensor with an ideal clock (no sampling jitter and no clock drift) that samples the sensor value at fix sample times ($t_k = k \cdot h$) with $k \in \mathbb{N}$ and a constant sample period ($h = t_{k+1} - t_k$). The following blocks are event activated. This means, whenever a block finishes its execution, or the time allocated to this block expires, the next block starts its execution. At the end of the chain, an actuator transforms the control sequence (u_k) in a stepwise constant continuous signal (sample-and-hold). As mentioned above, the delays (δ_k) are caused by the execution times and the blocking times of the controller task, plus the delays in the network communication.

The next section describes how to calculate valid response-time bounds based on real-time verification methods.

A. Calculating Time Delays

As mentioned, a time delay in a closed-loop control system has a direct impact on the control performance. In embedded systems these time delays are mainly induced by tasks (τ_i) which do not hold resources exclusively. Therefore, tasks or messages can be blocked or interrupted by other tasks or messages which lead then to time delays which are denoted as response times ($r(\tau_i)$).

In early design phases two possibilities are available to determine these times. The first one is to simulate the system and to extract from the trace files the response times ($r(\tau_i)$). Another possibility is to perform a real-time analysis delivering absolute bounds for the worst-case ($r^+(\tau_i)$) and best-case ($r^-(\tau_i)$) response times.

The simulation has the advantage that the average behavior of the system is considered. From this the average response and jitter times of a closed-loop control system can be derived. The great disadvantage of the technique is that corner cases are not necessarily considered, since a simulation has the typical coverage problems. Therefore analytical methods have to be used, able to calculate bounds for the worst-case and best-case response times in the system. One popular method is the real-time calculus [9]. The idea is to calculate bounds for the time behavior based on the min/plus and max/plus algebra.

For this paper the INCHRON Tool-Suite 2.3.0 [11] is used which enables the possibility to apply both techniques: simulation and validation. Consequently the condition $r^-(\tau_i) \leq r(\tau_i) \leq r^+(\tau_i)$ holds and allows it to explore the whole time behavior of a system.

As we will show in section V this condition can be used by a Matlab/Simulink model to analyze different scenarios. Since both approaches cover the possibility to calculate the end-to-end delay in a closed-loop control system, it is possible to consider the maximum, minimum and average end-to-end delay. Hence, one scenario could be to explore the closed loop with maximum or minimum delay. Another scenario is to consider a variable delay between maximum and minimum end-to-end delay with arbitrary probability distribution functions and therefore the effect of jitter.

But the question arises, how to create the worst-case performance for a given controller using the results of the real-time tools. As we will show in the following sections, the worst-case performance regarding the jitter can be created by a mixture of timing, functional behavior and knowledge of the chosen performance criteria and therefore the pure real-time tools results of a system are not sufficient.

When dealing with jitter, we have to differentiate between sampling jitter and sampling-actuating jitter. Sampling jitter is caused by different values for the sampling period, which does not occur in our system model. We only focus on the sampling-actuating jitter, that is caused by different values for the time delay. For controller applications it is adequate, to identify the response time of tasks in the event-chain of the control loop. Thus, there exist multiple jitter sources in the signal flow of a distributed controller. Assuming, the control loop is not nested or chained in any way, the jitter sources can be summed

together to one jitter in the closed loop. The maximum jitter of a task (τ_i) is defined as $j_i = r^+(\tau_i) - r^-(\tau_i)$. The summed maximum and minimum end-to-end delays can be described as $\delta_{min} = \sum_i r^-(\tau_i)$ and $\delta_{max} = \sum_i r^+(\tau_i)$ for all tasks (τ_i) in the control loop event chain. Thus the maximum round-trip jitter, that can occur is $j_{max} = \delta_{max} - \delta_{min}$.

IV. Influence of Jitter on Closed-Loop Control

In this section, we present a method capable of identifying the delay sequence that causes the worst performance degradation. We focus in our consideration on the maximum overshoot on a step response as performance criteria. This is a suitable criteria, because it is intuitive to identify in the plot of the control signal and relevant for many real-life systems like car suspension control. Thereby we assume a sufficiently small maximum end-to-end delay ($\delta_{max} < h$), which means no sampled value gets lost.

To identify the maximum overshoot of a closed-loop control system when influenced by a reference value or a specific disturbance, we need three kind of information from our system. First of all we need the actuating signal sequence (u_k) that affects the system. This sequence can be determined out of a simulation or a measurement of the controlled system. Further we need to know the behavior of the system on an impulse on the actuating signal. The sequence of the so called impulse response is denoted as (ϕ_k). Finally the time step ($m \cdot h$), when the overshoot occurs must be known.

As the end-to-end delay from sensor to actuator can vary between (δ_{min}) and (δ_{max}), the time when the actuator begins to affects the process lies between ($t_k + \delta_{min}$) and ($t_k + \delta_{max}$). This means, a time varying end-to-end delay modifies the duration, on which an actuating signal (u_k) affects the process. The maximum duration on the actuator is given as ($h + j_{max}$) and the minimum as ($h - j_{max}$). Figure 2 displays an example, where the actuating signal is extended to the duration of ($h + j_{max}$). As we can see, the maximum additional impulse on the system is a function of the maximum possible jitter and the step size between two consecutive actuating signals, e.g. (u_k) and (u_{k+1}).

In our approach, we take advantage of this influence on the duration of the actuating signal to manipulate the process in a manner to construct worst performance or the highest overshoot respectively. The result is of course an improbable but nevertheless possible scenario.

To reach worst performance, we modify the duration of the actuating signal with the maximum amount of absolute jitter that can occur, resulting in the most possible expansion or reduction of the actuating signal duration. When expanding the actuating signal by (j_{max}), one either can retain the duration of the next actuating signal as the sample period (h), or reduce it by (j_{max}).

The choice, which sample to delay and which to shorten by the maximum jitter depends on two factors. The first one is the difference between two consecutive actuating signals defined as $\tilde{u}_k = u_{k+1} - u_k$. The second factor is the impact of the error, made by jitter in the future, when the overshoot appears. This

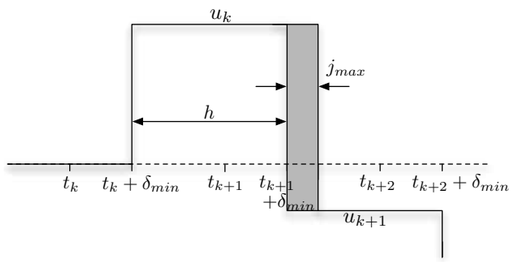

Fig. 2. Additional impulse on the system, caused by the additional delay of (j_{max}).

is done by the above described impulse response. Since the error made by the maximum jitter (grey area in figure 2) has a sufficiently short duration in relation to the sample period, it can be seen as an impulse on the system. This impulse is weighted by ($m - k$)-th element of the above described impulse response sequence (ϕ_{m-k}), to get the influence from an expansion on k-th actuating signal on the overshoot at m-th sample time.

If this procedure is done for all possible sample times up to ($m \cdot h$), the result is a sequence that identifies the impact of (j_{max}) on the overshoot for all sample times. By expanding the actuating signal duration on the local maximums of this sequence by the maximum jitter and shorten the duration for the local minimums, the bad delay sequence is defined.

V. Controller Design Flow

In this section, we propose a design flow for an embedded controller by considering time-varying delays based on the idea presented in section IV.

According to figure 3, the system engineer defines the platform design of the distributed system at the design entrance. On the other side, the task architecture, that defines the mapping between functionality and software tasks is derived from the controller design and other functions in the system. By mapping the task architecture to the hardware platform, the system architecture is specified. In this step, the scheduling is assigned to the tasks.

In the next step, suitable information about the occurring end-to-end delays of the messages that are exchanged by the components of the distributed controller has to be acquired. These delays are variable during the run-time of the system within an interval of best-case and worst-case end-to-end-delay.

To get the best control performance over the run-time of the system, the designer has to optimize the controller and to parametrize its settings in a way that holds best performance at the average end-to-end delay that occurs in the system. Therefore, the average end-to-end delay has to be known in advance of the controller design. It can be obtained from a simulation of the system's real-time behavior which has been described in section III-A.

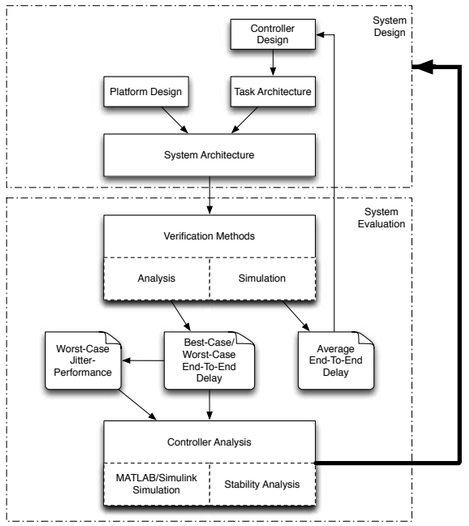

Fig. 3. Proposed new design flow for an embedded controller to consider the effect of time-varying delays to control performance.

While this leads to a controller that delivers its optimum performance in the average case, we must consider both, stability and performance for delay times higher or lower than the average. But this is not enough: We have also to consider the influence of the end-to-end delay jitter.

At the first step, we focus on the minimum and maximum end-to-end delays which can occur in the system. The real-time simulation we used in the previous step to gather the average end-to-end delay shows the coverage problem which is quite common for simulations, which means that simulations do not necessarily find the best and worst case end-to-end delays. This makes clear that we cannot rely on the simulation only. Therefore we make use of deterministic real-time analysis to get the absolutely best- and the worst-case end-to-end delays which can occur in the system. By using this information, we simulate our controller to check if it holds the stability and performance requirements even for the absolute minimum and maximum end-to-end delays.

As previously discussed, not only constant end-to-end delays affect the stability and performance of a controller, they rather suffer much stronger from time-varying end-to-end delays jittering between the maximum and minimum value. Based on the method we present in section IV, we identify the performance degradation due to jitter.

If the controller analysis does not hold the requirements, the system design, including the controller, platform and task architecture and even the scheduling have to be adjusted, which leads to a re-design phase. If system specifications are violated, the system design needs to be redesigned.

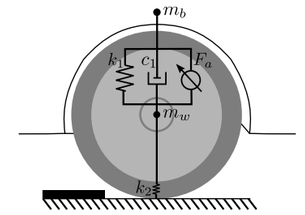

Fig. 4. Model of active car suspension based on [12].

VI. Example

In our example, we demonstrate our approach on an active car suspension control. An active car suspension control is implemented for comfort and for car stability reasons. To affect the chassis dynamic, a linear electric motor is placed between wheel and chassis. In [12] a simplified model of a suspension control is proposed and the mathematic equations of motion are derived. The model consists of a spring-mass-damper system and a linear motor, modeled as a force (F_a) between chassis and wheel (see figure 4). The wheel itself is modelled as a spring to the ground. As the model covers only a quarter of the car, the mass (m_b) denotes a quarter of the car's mass and (m_w) the mass of one wheel. The dynamics can be described in the form of a linear state-space model

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1}$$

with parameters

$$A = \begin{pmatrix} 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1}{m_b} & 0 & -\frac{c_1}{m_b} & \frac{c_1}{m_b} \\ \frac{k_1}{m_w} & -\frac{k_2}{m_w} & \frac{c_1}{m_w} & -\frac{c_1}{m_w} \end{pmatrix},$$

$$B = \begin{pmatrix} 0 & 0 & \frac{1}{m_b} & -\frac{1}{m_w} \end{pmatrix}^T.$$

Thereby the system input is denoted as (u) and the state vector as (x). A detailed description of the states can be found in [12].

The controller is designed as a digital control. Therefore equation 1 is discretized by the zero-order hold model, analogous to a digital-to-analog converter. The control system is then in the form of the following difference equation

$$x(k+1) = Ax(k) + Bu(k) \tag{2}$$

with state vector $x(k)$ including n state variables. The according state feedback

$$u(k) = -Kx(k) \tag{3}$$

is calculated by pole placement.

Our control objective is to minimize the difference between road and wheel ($z_w - z_r$) after a disturbance on the road in form of a stepwise displacement, as displayed in figure 4. The smaller the difference ($z_w - z_r$), the better the car handling in the case of a disturbance. Hence, car suspension control is a safety-critical system in the automotive domain.

$\overline{\delta}$	δ_{min}	δ_{max}	j_{max}
$0.636\,mm$	$0.757\,mm$	$3.790\,mm$	$4.974\,mm$

TABLE I
OVERSHOOT CAUSED BY DELAY

Based on the controller design with a sample time of $10\,ms$ and the surrounding functionality, a task architecture is derived. The platform architecture consists of three ECUs, interconnected by a CAN bus. In our case, the messages assigned to the control loop have the lowest priority on the CAN bus. Hence, a system architecture is defined with a bus speed of $500\,kbit/s$ resulting in an overall utilization of $25.88\,\%$.

The described real-time system was simulated for a duration of $3600\,s$, leveling off a distribution of the end-to-end delay. The average end-to-end delay ($\overline{\delta}$) can be calculated out of the histogram as an arithmetic mean value.

Based on the average end-to-end delay $\overline{\delta} = 1786\,\mu s$, an improved controller with optimized parameter setting is designed. This is done by an improved model of the system, including the average end-to-end delay ($\overline{\delta}$). A consequence, when considering $\overline{\delta} < h$ in the controller design is that the state vector ($x(k+1)$) does not only depend on the actual actuator signal ($u(k)$), but also on the previous one ($u(k-1)$). This is done by expanding the state space vector with the additional state variable

$$x_{n+1}(k) = u(k-1). \tag{4}$$

We obtain the new differential equation

$$\begin{pmatrix} x(k+1) \\ x_{n+1}(k+1) \end{pmatrix} = \begin{pmatrix} A & b_{-1} \\ 0' & 0 \end{pmatrix} \begin{pmatrix} x(k) \\ x_{n+1}(k) \end{pmatrix} + \begin{pmatrix} b_0 \\ 1 \end{pmatrix} u_k \tag{5}$$

where the coefficients (b_{-1}) and (b_0) are calculated, as described in [13] and depend on the amount of delay (δ).

For the ascertained average end-to-end delay and a road-displacement of $10\,mm$, the maximum overshoot of ($z_w - z_r$) is $0.636\,mm$.

According to that, the controller design is verified against the highest possible end-to-end delay $\delta_{max} = 8790\,\mu s$ that possibly can occur. This delay is calculated from real-time analysis methods, described in section III-A, resulting in a step response with an overshoot of $3.79\,mm$. The real-time analysis also identifies an minimum end-to-end delay that causes an overshoot of $0.757\,mm$.

Finally an even worse performance, referring to the maximum overshoot, can be constructed by assuming a jitter alternating between minimum and maximum end-to-end delay. This is done out of the methods described in section IV. The overshoot of the system, when it is delayed with the bad-delay sequence, is $4.974\,mm$.

When the overshoot exceeds our specification, we need to redesign our system by adapting our controller, modifying our platform or assigning a new scheduling.

VII. CONCLUSION

In this paper we presented an approach for a digital controller design by using results from a real-time simulation to determine the average end-to-end delay to design a controller with best performance in most cases. Afterwards, the controller design is verified against the best-case and worst-case end-to-end delays, calculated by real-time analysis methods, to check if the control performances in these points of operation are satisfying. We could also construct an even worse performance by combining real-time analysis results with system characteristics.

To improve the results and push the worst-case performance to a more optimistic and valid direction, the maximum step between two consecutive delays has to be regarded. Or, in other words, is it possible, that a maximum end-to-end delay (δ_{max}) can follow the minimum end-to-end delay (δ_{min}) or vice versa?

One further challenge would be to develop an analysis method that allows to check whether a specific system is more sensitive to a constant delay or rather to jitter. One possibility to reduce the jitter sensitivity would be to increase the sample rate of the controller. This would cause in general smaller differences between two consecutive actuating signals and therefore would reduce the influence of jitter. However, a higher sample rate would possibly result in higher response times. This tradeoff needs to be studied in further research.

REFERENCES

[1] H. Nyquist, "Regeneration Theory," *Bell System Technical Journal*, vol. 11, no. 1, pp. 126–147, 1932.

[2] C.-Y. Kao and B. Lincoln, "Simple Stability Criteria for Systems with Time-Varying Delays," *Automatica*, vol. 40, 2004.

[3] A. Cervin, B. Lincoln, J. Eker, K.-E. Årzén, and G. Buttazzo, "The Jitter Margin and Its Application in the Design of Real-Time Control Systems," in *10th International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA)*, Göteborg, Sweden, Aug. 2004.

[4] B. Lincoln and A. Cervin, "Jitterbug: A Tool for Analysis of Real-Time Control Performance," in *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, Dec. 2002.

[5] B. Lincoln, "Jitter Compensation in Digital Control Systems," in *American Control Conference, 2002*, Anchorage, AK, USA, May 2002.

[6] D. Goswami, R. Schneider, and S. Chakraborty, "Co-design of Cyber-Physical Systems via Controllers with Flexible Delay Constraints," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Yokohama, Japan, 2011.

[7] "SimEvents," http://www.mathworks.com/products/simevents/.

[8] T. Bund, S. Moser, S. Kollmann, and F. Slomka, "Guaranteed Bounds for the Control Performance Evaluation in Distributed System Architectures," in *Proceedings of the International Conference on Real-Time and Embedded Systems (RTES 2010)*, Singapore, Sep. 2010.

[9] E. Wandeler, "Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems," Ph.D. dissertation, ETH Zurich, September 2006.

[10] J. Nilsson, "Real-Time Control Systems with Delays," 1998.

[11] "Inchron Tool-Suite 2.3.0," http://www.inchron.com.

[12] K. Hyniova, A. Stribrsky, J. Honcu, and A. Kruczek, "Active Suspension System with Linear Electric Motor," *WSEAS TRANSACTIONS on SYSTEMS*, vol. 8, pp. 278–287, 2009.

[13] M. S. Branicky, S. M. Phillips, and W. Zhang, "Stability of Networked Control Systems: Explicit Analysis of Delay," in *Proceedings of the American Control Conference*, Chicago, Illinois, Jun. 2000.

Genauigkeit, Robustheit und Powerprofiling für Cyber Physical Systems

Joseph Wenninger
and Florian Schupfer
and Jan Haase
and Christoph Grimm
Institute of Computer Technology
Vienna University of Technology
Vienna, Austria
Email: {wenninger|schupfer|haase|grimm}@ict.tuwien.ac.at

***Zusammenfassung*—Die Vernetzung heterogener Systeme stellt neue Anforderungen an Entwurfsmethoden und Entwurfswerkzeuge. Diesen Anforderungen werden bislang weder existierende Werkzeuge noch Methoden gerecht. Dieser Beitrag diskutiert die Probleme und gibt einen Überblick über das in Entwicklung befindliche Framework zum Entwurf von 'Cyber Physical Systems': SYCYPHOS. SYCYPHOS [1] unterstützt die Modellierung verteilter und heterogener Systeme und – auf der Basis von funktionalen Modellen – Power Estimation und Methoden zur Analyse der Robustheit.**

I. EINLEITUNG

Eingebettete Systeme, bestehend aus Hardware (HW) und Software (SW) können heute systematisch und 'top-down' entworfen werden. Der Entwurf wird durch Werkzeuge auf unterschiedlichen Abstraktionsebenen sehr gut unterstützt.

Fortschritte bei der Integration ermöglichen neben der Integration von mehr digitalen Prozessoren auch die Integration von analogen Sensorinterfaces und Funkschnittstellen. Die Möglichkeit, eingebettete Systeme über Funk zu vernetzen schafft neue Möglichkeiten. Für Systeme, in denen ein physikalischer Teil durch einen 'Cyber' Teil vernetzt wird, hat sich der Begriff 'Cyber Physical System' etabliert [2]. Beispiele für 'Cyber Physical Systems' (CPS) sind zum Beispiel komplexe Regelsysteme in Smart Grids, Robotersysteme in komplexen flexiblen Automatisierungssystemen, aber auch in- und inter-Automobil Kommunikationsnetzwerke und die darauf aufbauenden Funktionen.

Ein einfaches Beispiel für die Heterogenität und Komplexität in CPS ist in Abb. 1 dargestellt: Ein Reifendruckmesssystem (TPMS), das in einem Reifen eingebettet den Reifendruck misst und ihn mit unterschiedlicher Priorität in das Kommunikationsnetz des Autos einspeist. Das Design umfasst einen Sensor, Sensordatenaufbereitung, einen Mikrocontroller, Funkschnittstellen, sowie Software (Betriebssystem, Kommunikationsstack). Hierbei werden hohe Anforderungen an die Zuverlässigkeit (Echtzeit, Sicherheit, Dependability) gestellt, wobei gleichzeitig die Stromaufnahme sehr niedrig sein muss – Herausforderungen, die nur durch Methoden wie optimierte Protokolle, Sensordatenfusion sowie im nicht unwahrscheinlichen Fall von Fehlern z.B. bei der Kommunikation robustem Verhalten des Gesamtsystems.

Der Entwurf von Cyber-Physical Systems (CPS) ist eine Herausforderung, die bislang nicht hinreichend gelöst ist. Problematisch aus Sicht der Entwurfsmethodik sind insbesondere

1) Keine gemeinsame Simulation von Netzwerk *und* Hardware/Software Design.
2) Späte Verifikation von Eigenschaften wie Stromaufnahme, Genauigkeit, Zuverlässigkeit.

Dies kann zu einem hohem Risiko beim Entwurf solcher Systeme bzw. zu überlangen Projektlaufzeiten führen.

Die Designmethodologie und das daraus entstehende Entwurfsframework SYCYPHOS wurden bereits in [3] vorgestellt und generell abgehandelt. In diesem Beitrag wird auf zwei große Teilaspekte des SYCYPHOS-Frameworks detailierter eingegangen.

1) Profiling von Genauigkeit/Robustheit
2) Profiling der Stromaufnahme auf CPS-Level.

II. PROFILING VON GENAUIGKEIT, ROBUSTHEIT

Um die Auswirkungen von einzelnen Störungen entlang eines komplexen Signalverarbeitungspfads zu beobachen und zu analysieren wird Affine Arithmetik [4] verwendet. Hierbei werden Parameterabweichungen

Abb. 1. Ein zukünftiges TPMS System mit Kommunikation innerhalb eines Automobiles als Beispiel für ein Cyber Physical System. Das Gesamtsystem besteht aus mehreren Sensorknoten, sowie Knoten, die nur das Routing übernehmen. Die Sensoren sind direkt mitsamt der Batterie in den Reifengummi integriert. Aufgrund von Beschleunigungen, Luftdruck und Temperaturen schließen sie auf den Zustand des Reifens. Über im Auto verteilte Logik, sowie eine etwaige Zentraleinheit werden Gefahrensituationen erkannt und einerseits der Fahrer informiert, andererseits im Extremfall das Auto durch dosiertes Bremsen und Gegenlenken in einen schadensbegrenzenden Zustand versetzt. Die Kommunikation erfolgt drahtlos um Gewicht zu sparen und das System ist selbstorganisierend um Energie zu sparen und auch trotz externer Störquellen zuverlässig zu funktionieren.

vom nominellen Wert als dem idealen Wert überlagerte Intervalle modelliert. Diese Intervalle werden *symbolisch* dargestellt und bleiben während der Simulation als symbolische Darstellung erhalten, d.h. Ausgabe der Simulation ist für jeden Wert neben dem erwartetem nominellen Wert eine symbolische Darstellung der Abweichungen und ihrer Ursachen. Die Algorithmen zur Semi-Symbolischen Simulation mit Affiner Arithmetik sind in [5], [6], [7] genauer beschrieben.

Terme der Affine Arithmetik sind definiert als ein nomineller Wert x_0 überlagert mit einer Menge $\mathcal{N}_{\tilde{x}}$ von skalierten Abweichungssymbolen $x_i\epsilon_i$, welche die Bereiche verursacht von den einzelnen Parameterabweichungen darstellen:

$$\tilde{x} = x_0 + \sum_{i \in \mathcal{N}_{\tilde{x}}} x_i \epsilon_i \qquad \epsilon_i \in [-1, 1] \tag{1}$$

Mathematische Operationen angewendet auf die Affinen Formen dargestellt in Gleichung 1 können in *affine operations*, welche exakte Lösungen ergeben und *non-affine operations*, die sich durch eine pessimistische aber sichere Abschätzung ergeben, unterschieden werden.

Der Vorteil einer Semi-Symbolischen Simulation gegenüber numerischen Simulationen ist die Rückverfolgbarkeit einzelner Abweichungen vom Simulationsresultat zurück zu ihrem Ursprung. Damit werden nicht nur Aussagen über die absolute Einhaltung von Spezifikationen möglich, sondern auch eine Analyse der Auswirkungen der einzelnen Abweichungen auf dieses Simulationsresultat.

Abb. 2 zeigt ein einfaches Beispiel für das Profiling der Genauigkeit und Robustheit eines einfachen Receivers. Vier Parameterschwankungen wurden berücksichtigt. Eine Offset Abweichung berücksichtigt mögliche Schwankungen im Potential des Systems, Abweichungen vom nominellen Wert der Verstärkung der beiden Filter werden durch zwei weitere Bereiche modelliert. Eine Abweichungen des Signalverlaufs von der idealen Sinusschwingung werden als Ungenauigkeit des *Local Oscillators* hinzugefügt. Eine Simulation dieses bereichsbasierten Modells liefert mit dem angegebenen AM modulierten Testsignal ein Simulationsresultat welches in seinen Teilintervallen die Auswirkungen der Abweichungen auf das Systemverhalten repräsentiert.

Aus den (teils symbolischen) Ausgaben können Analysetools anschließend das SNR des Empfangssignals bestimmen. Bei unzureichender Signalqualität könnte man diejenige der vier Abweichungen bestimmen, die das größte Potential für eine Signalverbesserung liefert und diese optimieren.

A. Regressiontest und Reports

Die Komplexität und Heterogenität von CPS erfordert neben Werkzeugen zur Modellierung/Simulation und Analyse insbesondere auch systemumfassende Werkzeuge zur Verifikation. Für Software und digitale Schaltungen sind Regresssimulationen schon weit verbreitet und diverse Sprachen wie PSL [8] (Property Specification Language) werden eingesetzt um den benötigten Aufwand bei der Verifikation durch automatisierte

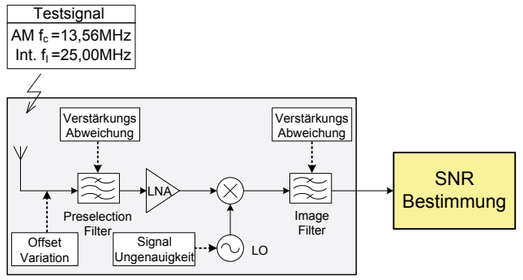

Abb. 2. Semi-symbolische Simulation: Struktur eines Empfängers, mit den für die Robustheits- und Genauigkeitsabschätzung relevanten Blöcken, die über die Bereichsarithmetik der affinen Arithmetik evaluiert werden.

Fehlererkennung zu verringern. Bei der Verifikation von analogen Schaltungen oder heterogenen Systemen wie CPS sind insbesondere Eigenschaften 'analoger' Signale zu spezifizieren, und zwar in analogen Schaltungen, auf Embedded System Level und in physikalischen Systemteilen. Viele Eigenschaften von analogen Signalen werden immer noch manuell überprüft.

SYCYPHOS unterstützt den Regressiontest durch ein Offline-Analysetool. Der Ablauf der Verifikation ist in Abb. 3 dargestellt. Aus der Spezifikation müssen die Eigenschaften, welche überprüft werden sollen, extrahiert werden. Im Rahmen der Spezifikation werden Eigenschaften (z.B. Einschwingzeiten, Genauigkeit, aber auch Stromaufnahme) spezifiziert, die mit Hilfe von Templates und einer einfachen Sprache beschrieben werden. Die Sprache umfasst derzeit arithmetische und logische Verknüpfungen von Signalen, sowie verschiedene Toleranzschemata. Das Analysetool liest derzeit VCD-Files; die Anbindung an Power Profiling ist leider noch nicht implementiert.

Das Offline-Analysewerkzeug wird nach den Simulationsläufen gestartet. Das Tool liefert den aktuellen Status der spezifizierten Eigenschaften und generiert eine HTML-Seite als Report, der Ergebnisse zusammenfasst. Bei erkannten Fehlern werden die genauen Zeitpunkte bzw. Start und Endzeit, für die Dauer während der die Spezifikation verletzt wurde, angegeben. Dadurch können die Fehler in der Waveform leicht lokalisiert werden. Abbildung Abb. 4 zeigt ein Beispiel für solch einen Report.

III. Power Profiling

In SYCYPHOS ermöglicht Power Profiling eine simulationsgestützte, zunächst eher qualitative Schätzung und

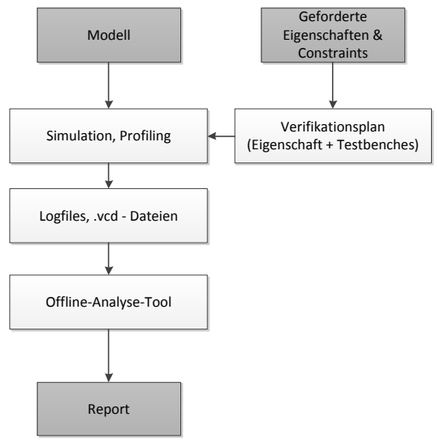

Abb. 3. Verifikations- und Analysetool für Regressiontest. Dargestellt ist der Ablauf bei einemWerkzeuggestützten Regressionstest. Als erstes müssen das zu untersuchende Modell entwickelt werden und die geforderten Randbedingungen beschrieben werden. Während der Simulation werden die gewünschten Eigenschaften (Properties) mitprotokolliert. Nach dem Simulationslauf wird das Analysetool auf die Logdaten angewandt. Durch diese Anwendung wird der Verifikationsreport Abb. 4 generiert.

Analyse des Energieverbrauchs auf CPS-Level. Hierzu wird einzelnen Aktivitäten (z.B. Messen, Kommunizieren) in funktionalen Modellen oder dem Benutzen von Ressourcen in Architekturmodellen ein (geschätzter oder gemessener) Energieverbrauch zugeordnet. Während der Simulation werden Aktivitäten und das Benutzen von Ressourcen in ein Logfile protokolliert (Abb. 5, vgl. [9], [10], [11], [12]).

Derzeit implementiert ist insbesondere auch ein nachrichtenbezogenes Energietracking. Hierbei kann innerhalb des 'Air' Objekts mittels einer Payload Extension beim Transport von Information (=Transactions) der Transaktion die Information mitgegeben werden, wie viel Energie der Netzwerkknoten zur Erzeugung oder zur Weiterleitung benötigt hat.

Wenn die übermittelte Nachricht ihr Ziel erreicht hat, kann der akkumulierte Energieverbrauch genauso wie der Energieverbrauch pro Knoten ausgewertet werden; gleiches gilt für 'verlorene', nicht übermittelte Nachrichten. Beim nachrichtenbezogenen Power Profiling werden die Metadaten den virtuellen gesendeten Packeten als Erweiterung angehängt. Hierfür werden die Generic Payload Extensions von TLM 2.0 [13], [14] verwendet. In Abb. 6 ist der verwendete Ansatz dargestellt.

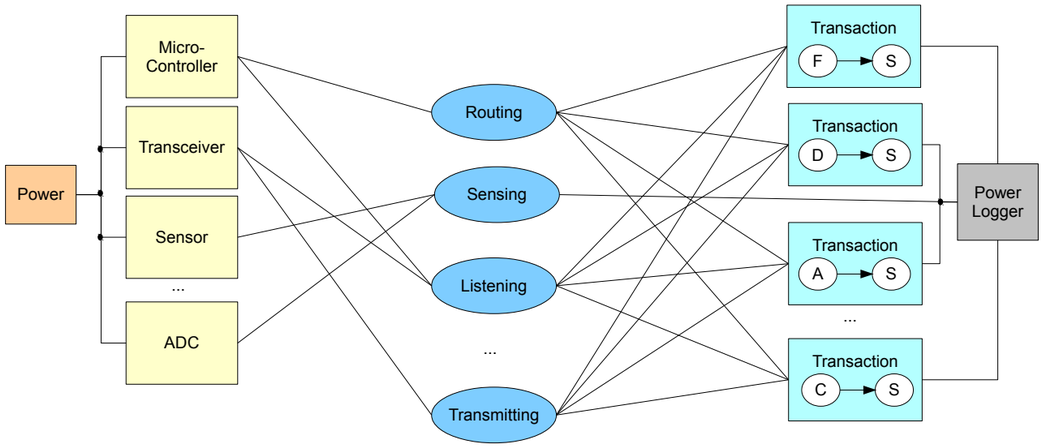

Abb. 5. Multilevel Powerprofiling: Die linke Hälfte der Abbildung bezieht sich auf die Knotenebene (Sensor-, Aktor und Kommunikationsknoten). Auf dieser Ebene wird Energieverbrauch für die Erstellung bzw. Verarbeitung von Kommuniktationspaketen ermittelt. Die Ergebnisse können sowohl auf der Knoteneben mitprotokolliert werden, um in der Überprüfungsphase Optimierungspotentiale erkennen zu können, als auch an die Netzwerkebene weitergeleitet werden. Der rechte Teil der Grafik beschreibt die Netzwerkebene. Hierbei werden Ende-zu-Ende Kommunikationen (Kommunikationen von einem Quellknoten zu einem Empfangsknoten, mit eventuellen Zwischenhops) betrachtet. Die Kommunikationen werden dabei zu Transaktionen zusammengefasst. Das Anfügen des Knotenlevel-Energieverbrauches an die Transaktionen erlaubt es, die Kosten einzelner Datenpakete auf der Gesamtsystemebene zu evaluieren.

Verificationreport

Date: 05.03.2012 Time: 11:41

Status: Failed

Number of testcases: 4
Number of failed testcases: 1

Testcase	Status
Property 1	OK
Property 2 ERROR between 55 µs and 57 µs	Failed
Property 3	OK
Property 4	OK

Abb. 4. Beispiel für einen Report nach Regressionstest. Durch die einfache Struktur klar ersichtlich ist, welche Überprüfungen funktioniert haben und welche fehlgeschlagen sind, weil z.B. Toleranzbereiche verletzt worden sind. Im Report wird auch angezeigt, in welchem Bereich der simulierten Zeit der Verifikationsfehler aufgetreten ist.

Die SYCYPHOS-Implementierung des Powerprofilings beruht auf Kenntnissen, die im Rahmen des PAWiS Projektes ([15], [16]) gewonnen worden waren. PAWiS

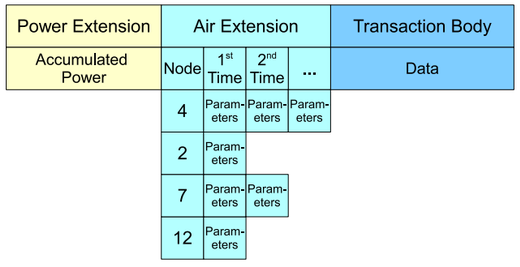

Abb. 6. Annotation von Nachrichten für das Power Profiling. Den simulierten Nutzdaten (Payload) werden über TLM 2.0 - Generic Payload Extensions zusätzliche, nur für die Simulation wichtige Informationen beigefügt. Dadurch wird das Nachverfolgen und Aufsummieren des Energieverbrauches über das gesamte vernetzte System für alle Kommunikations-Hops möglich.

basierte auf OMNeT++ [17], [18]. Die Konzepte konnten einfach auf SystemC und TLM portiert werden und erwiesen sich als sehr flexibel für Erweiterungen hinsichtlich Multilevel-Modellierungen. Die SYCYPHOS-Implementierung wurde in den Projekten SNOPS und SmartCoDe [19] verwendet. Ähnliche Ansätze werden auch im Projekt GREENHome-SP verwendet und werden später nach SYCYPHOS zurückfließen.

IV. Zusammenfassung und Ausblick

Der Beitrag hat einen Überblick über zwei große Teilaspekte des in der Entwicklung befindlichen Framework zum Entwurf von Cyber Physical Systems gegeben.

Wesentliche Innovationen in SYCYPHOS sind die Kombination des Entwurfs auf Embedded System Level mit einer Simulation auf Netzwerkebene. Durch Profiling können so Optimierungen, etwa der Firmware für Powermanagement oder Methoden zur Sensordatenfusion durchgeführt werden.

a) Stand der Implementierung: Die beschriebenen Teile von SYCYPHOS sind im Rahmen verschiedener Projekte weitgehend implementiert worden. Sie wurden auch zum Entwurf z.B. des TPMS Systems eingesetzt. Allerdings steht insbesondere die Integration der Einzelteile zu einem Framework für den durchgängigen Entwurf von CPS noch aus. Bibliotheken und Profilingmethoden sind derzeit noch nicht vollständig integriert, so dass derzeit für die einzelnen Entwurfsschritte (Modellierung, Profiling) teils unterschiedliche Modelle verwendet werden müssen.

b) Aktuelle und zukünftige Arbeiten: Schwerpunkt der aktuellen Arbeit ist die weitere Integration der Bibliotheken und der Verfahren zum Profiling sowie die Ergänzung der Bibliotheken um weitere Komponenten. Das in Entwicklung befindliche Gesamtframework soll damit innerhalb der nächsten ein bis zwei Jahre den Status erreichen, dass es von der wissenschaftlichen Community sowie von der Industrie produktiv eingesetzt werden kann.

Danksagungen

Die Forschung im Bereich der Affinen Arithmetik wurde durch den Wiener Wissenschafts-, Forschungs- und Technologiefonds (WWTF) im Rahmen des Projektes ICT08-012 gefördert.

Die Forschung im Bereich des Powerprofilings und der Optimierung auf Netzwerkebene wurde im Rahmen des SNOPS Projektes durchgeführt. SNOPS wurde durch den Staat Österreich im Rahmen von FIT-IT (Forschungsbeihilfennummer: 815069/13511) gefördert und war ein Subprojekt des ITEA2 Projektes GEODES (Forschungsbeihilfennummer: 07013).

GREENHome-SP ist ein Projekt, das vom Forschungszentrum Telekommunikation Wien (FTW) als strategisches Subprojekt zum Projekt GREENHome gefördert wird.

Literatur

[1] (2012, Mar.) Webseite des SYCYPHOS-Frameworks. [Online]. Available: http://sycyphos.ict.tuwien.ac.at

[2] E. A. Lee, "Cyber physical systems: Design challenges," in *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, May 2008.

[3] J. Wenninger, K. Gravogl, F. Schupfer, J. Haase, and C. Grimm, "SYCYPHOS: Ein Framework zum Entwurf von Cyber Physical Systems," in *Zuverlässigkeit und Entwurf - 5. GI/GMM/ITG-Fachtagung vom 27. bis 29. September 2011 in Hamburg-Harburg*, 2011.

[4] L. H. de Figueiredo and F. Stolffi, *Self-Validated Numerical Methods and Applications*, ser. Brazilian Mathematics Colloquium monographs. IMPA/CNPq, Rio de Janeiro, Brazil, 1997.

[5] C. Grimm, W. Heupke, and K. Waldschmidt, "Analysis of Mixed-Signal Systems with Affine Arithmetic," *IEEE Transactions on Computer Aided Design of Circuits and Systems*, vol. 24, no. 1, pp. 118– 123, 2005.

[6] D. Grabowski, C. Grimm, and E. Barke, "Semi-Symbolic Modeling and Simulation of Circuits and Systems," in *International Symposium on Circuits and Systems 2006 (ISCAS 2006)*. Kos, Greece: IEEE Press, May 2006.

[7] D. Grabowski, M. Olbrich, and E. Barke, "Analog circuit simulation using range arithmetics," in *ASP-DAC '08: Proceedings of the 2008 Asia and South Pacific Design Automation Conference*. Seoul, Korea: IEEE Computer Society Press, 2008, pp. 762–767.

[8] *IEEE Standard for Property Specification Language (PSL)*, IEEE Std 1850-2010, 2010.

[9] J. Haase, J. Molina, and C. Grimm, "High level energy consumption estimation and profiling for optimizing wireless sensor networks," in *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, july 2010, pp. 537 –542.

[10] J. Moreno, J. Haase, and C. Grimm, "Energy Consumption Estimation and Profiling in Wireless Sensor Networks," in *ARCS'10 Workshop Proceedings*, 2010, pp. 259 – 264.

[11] J. Haase, M. Damm, J. Glaser, J. Moreno, and C. Grimm, "Systemc-based power simulation of wireless sensor networks," in *Proceedings of the Forum of Design Languages*, 2009.

[12] J. Wenninger, M. Damm, J. Moreno, J. Haase, and C. Grimm, "Multilevel Sensor Node Simulation within a TLM-like Network Simulation Framework," in *ARCS'10 Workshop Proceedings*, 2010, pp. 211 – 216.

[13] J. Aynsley, "OSCI TLM-2.0 Language Reference Manual," Open SystemC Initiative, Tech. Rep., 2009.

[14] ——, "OSCI TLM2 User Manual," Open SystemC Initiative, Tech. Rep., 2008.

[15] S. Mahlknecht, J. Glaser, M. Rötzer, and T. Herndl, "Systems Architecture for Power Aware Wireless Sensor Network Nodes," FIT-IT Project Proposal "PAWiS", 4. Apr. 2005.

[16] D. Weber, J. Glaser, and S. Mahlknecht, "Discrete event simulation framework for power aware wireless sensor networks," vol. 1, June 2007, pp. 335–340.

[17] A. Varga, "The OMNeT++ discrete event simulation system," in *European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, 2001.

[18] ——, *OMNeT++ Discrete Event Simulation Sytem User Manual*, 29. Mar. 2005.

[19] S. Mahlknecht, M. Damm, J. Moreno, and E. Holleis, "Executable specification of a SmartCoDe node," ICT-2009-247473 - Deliverable D-2.2, 12. Dec. 2010.

Betriebssysteme für cyber-physische Systeme – Wie weit reichen die klassischen Abstraktionen?

Matthias Werner, Andreas Löscher und Mario Haustein
Professur Betriebssysteme
TU Chemnitz
Email: {mwerner|loesa|hamari}@cs.tu-chemnitz.de

Jan Richling
FG Kommunikations- und Betriebssysteme
TU Berlin
Email: richling@cs.tu-berlin.de

***Zusammenfassung*—Schon vor Jahren wurde die Forderung nach spezifischen Betriebssystemen für cyber-physische Systeme aufgestellt. Wir untersuchen in diesem Beitrag, inwieweit sich die klassischen Abstraktionen auf CPS anwenden lassen, oder wie sie neu interpretiert werden müssen.**

I. Einführung

Schon vor einigen Jahren wurde die Forderung nach spezifischen Betriebssystemen (BS) oder der Definition neuer Betriebssystemschnittstellen für cyber-physische Systeme (*cyber-physical systems*, CPS) erhoben, vgl. z. B. [1], [2]. In [3] wird behauptet, dass es wirklich erstaunlich wäre, wenn die klassischen Schnittstellen zum BS noch tauglich wären.[1]

Es stellt sich die Frage, ob CPS überhaupt spezielle BS benötigen. Um diese zu beantworten, betrachten wir die etwas allgemeinere Frage: Brauchen wir überhaupt unbedingt BS? Die Antwort darauf ist: Nein, *unbedingt* brauchen wir BS nicht. Aber: In der Praxis hat sich ihr Einsatz als überaus vorteilhaft erwiesen. Sie entheben den Anwendungsprogrammierer davon, sich immer wieder um die gleichen, lästigen (und dazu fehleranfälligen) Aufgaben zu kümmern. Es ist die feste Überzeugung der Autoren, dass diese Vorteile auch bei der neuen Klasse von CPS zu tragen kommen. Es wäre also von Interesse, spezielle BS für CPS zu entwickeln.

In der bisherigen Praxis der CPS-Forschung werden (ggf. um eine Middleware wie in [4] ergänzte) existierende BS, meist Echtzeitbetriebssysteme (z.T. aber auch Standardbetriebssysteme, vgl. [5]) benutzt, nicht ohne häufig die mangelnde Tauglichkeit der etablierten Abstraktionen zu beklagen. Ein Forschungsansatz zu einem dedizierten CPS-Betriebssystem ist den Autoren jedoch nicht bekannt. Das mag daran liegen, dass es sehr schwer erscheint, die in einem halben Jahrhundert in dem Gebiet der BS gewachsenen Konzepte und etablierten Abstraktionen auf einen Schlag zu überwinden und durch etwas Neues zu ersetzen.

In dieser Veröffentlichung werden daher die „klassischen“ BS-Abstraktionen und -Konzepte auf ihre Tauglichkeit in CPS untersucht. Wenn notwendig, wird versucht, vorhandene Konzepte neu zu interpretieren und ihr Verständnis in Hinblick auf CPS zu modifizieren. Dadurch sollen Anforderungen an konkrete BS entwickelt werden, die CPS besser als herkömmliche BS unterstützen.

Der Rest dieser Veröffentlichung ist folgendermaßen strukturiert. In Abschnitt II werden die einzelnen Schichten einer CPS-Architektur vom Standpunkt des BS diskutiert. Darauf aufbauend betrachtet Abschnitt III typische klassische Abstraktionen und Konzepte von BS und analysiert ihre Einsatzfähigkeit in einem CPS. In Abschnitt IV wird ein Resümee gezogen und kurz Beispiele für erweiterte Konzepte diskutiert.

II. Schichten

A. Betriebssysteme

Ein BS hat im wesentlichen zwei Funktionen (vgl. z. B. [6], [7]): Zum einen bietet ein BS eine *abstrakte Maschine*, auf der eine Anwendung ausgeführt wird. Reale Eigenschaften auf niedriger Ebene werden versteckt und auf abstraktere Konzepte abgebildet.[2] Zum anderen ist ein BS ein *Ressourcenmanager*, der die Komponenten des Systems verwaltet und ihre Nutzung koordiniert. Ein Beispiel für ersteres ist der Zugriff auf persistente Medien wie Flash-Speicher oder Festplatten, der durch das

[1] [3] im Original: „Is the conceptual boundary between the operating system and the programming language (a boundary established in the 1960's) still the right one? It would be truly amazing if it were.“

[2] [7] spricht davon, dass durch ein BS „hässliche“ Schnittstellen in „schöne“ Schnittstellen verwandelt werden.

Konzept des *Files* abstrahiert wird, so dass technische Details wie Adressierungsmethoden auf dem physischen Medium transparent werden. Eine Ressource im Sinn des Ressourcenmanagements ist z. B. die Rechenzeit auf dem (Haupt-)Prozessor eines Computers, die durch den Scheduler verwaltet wird.

Mit diesen beiden Funktionen unterstützt ein BS das End-to-End-Designprinzip [8], [9], das häufig durch ein *Sanduhrendesign* implementiert wird. Dabei steht oben eine große Klasse von Anwendungen, die über eine generische und eng begrenzte Schnittstelle (BS) auf einer größeren Klasse von Hardwarearchitekturen arbeiten. Damit wir also das in der Mitte stehende BS evaluieren können, empfiehlt es sich, zunächst die Schicht der Anwendungen und der unterliegenden Hardware[3] zu betrachten.

B. Anwendungen

CPS sind reaktive, verteilte Systeme. Deshalb werden sich die Konzepte entsprechender Programmiermodelle in CSP-Anwendungen wiederfinden und sollten somit auch von einem BS unterstützt werden. Auf den ersten Blick scheint zwar die verwendete Programmiersprache unabhängig vom BS zu sein, jedoch zeigt ein genauerer Blick, dass BS sogar im starken Maße durch die unterstützten Programmiersprachen – oder genauer: durch die genutzten Konzepte – bestimmt werden. Sprach- und BS-Konzepte stehen folglich in enger Wechselwirkung.

Der typische Ansatz, um reaktive System zu entwerfen, nutzt graphische Modellierungssprachen (z.B. Simulink [10] oder Ptolemy II [11]), die auf diskreten Ereignissystemen, kontinuierlichen Übertragungssystemen bzw. Hybriden dieser beiden basieren. Damit wird ein Programmiermodell genutzt, das bereits vor dem Aufkommen des modellgetriebenen Designs existierte, nämlich die Datenflussorientierung. Beispiele für Datenflusssprachen sind Lucid [12] oder Chuck [13]. Letztere realisiert ein weiteres Konzept, das für eine CPS-Anwendung unterstützt werden muss: Echtzeit. Die Echtzeitkonzepte anderer Sprachen wie z. B. PEARL [14] basieren ohnehin stark auf klassischen BS-Konzepten.

Eine weitere wesentliche Eigenschaft von CPS-Anwendungen ist die Verteiltheit. Falls eine Programmiersprache überhaupt Verteiltheit unterstützt, ist es meist eine Aufgabe des Laufzeitsystems (und damit auch des BS), eine gewisse Ortstransparenz zu erreichen. Dies wird dadurch erschwert, dass aufgrund der physischen Eigenschaften einer CPS-Anwendung häufig ein gewisses *Ortsbewusstsein* (*awareness*) notwendig ist. Wie dieser Widerspruch zwischen Transparenz und Bewusstsein gelöst werden kann, wird stark durch die vom BS angebotenen Abstraktionen beeinflusst.

[3]Man beachte, dass der Begriff der „Hardware“ hier für CPS in einem wesentlich weiteren Sinn gebraucht wird, als es für klassische Systeme üblich ist.

C. Hardware

Die meisten Konzepte heutiger BS stammen aus den 50er bis 70er Jahren des vorherigen Jahrhunderts, also aus einer Zeit, in der die Rechner typischerweise der von-Neumann-Architektur [15] entsprachen. Heute gibt es kaum noch Rechner, die strikt diesen Prinzipien folgen: Mit der *out of order execution* sind Elemente einer Datenflussarchitektur aufgenommen worden, Multicore-Rechner unterlaufen die Annahme der sequentiellen Programmausführung und verteilte Systeme, Caches und NUMA (*Non-Uniform Memory Architecture*) lösen den einheitlichen Speicher auf.

Trotzdem „funktionieren“ die Abstraktionen des von-Neumann-Rechners weiterhin relativ gut. In bestimmten Aspekten wurden die Konzepte und Abstraktionen erweitert und den neuen Gegebenheiten angepasst.

Entsprechend dem von-Neumann-Konzept wird ein Rechensystem in Prozessor(en), Speicher und Ein-/Ausgabe unterteilt, wobei letztere die Interaktionen des Rechners mit seiner Umgebung kapselt, also in gewisser Weise die Grenze des Rechners gegen die Umgebung definiert. Bei einem CPS ist eine Abgrenzung *an dieser Stelle* aber nicht sinnvoll, da es gerade das Ziel ist, im Zusammenwirken des Rechners und physischer Elemente Wirkungen zu erreichen. Folglich muss die Grenze der ausführenden Einheit nach außen verschoben werden, will man das prinzipielle Modell erhalten. Die physischen Elemente gehören damit explizit zum System und können als Ausführungseinheiten betrachtet werden, anstatt nur durch Ein-/Ausgabeoperationen manipulierbar zu sein.

Die Idee, physische Elemente als Ausführungseinheiten zu betrachten, ist keineswegs neu: Es gibt historische Rechnersysteme, die man als CPS auffassen kann: sogenannte Hybridrechner (z.B. der Dornier 960 [16]), die aus einer Kombination von Analog- und Digitalrechnern bestanden. Die Entwicklung dieser Hybridrechner endete in den 70er Jahren des vorherigen Jahrhunderts – Analog- und Hybridrechner erwiesen sich als zu unflexibel, um ihren Geschwindigkeitsvorteil im größeren Maßstab umsetzen zu können.

Bei Analog- und Hybridrechnern geht es darum, die Rechenfähigkeit physikalisch operierender Elemente zu nutzen, um Ergebnisse von Rechenproblemen (in der Regel Differentialgleichungen) zu finden. In CPS geht es darum, bestimmte Zustände bzw. ein bestimmtes Verhalten der physischen Elemente zu erreichen. [17] listet eine Anzahl nutzbarer Analogelemente, wie sie auch in CPS-Anwendungen zu finden sind. Bei Hybridrechnern funktionierten diese Elemente ausschließlich elektrisch, während bei CPS im Prinzip jede Art physischer Signale möglich ist.

Eine weitere Abweichung zum von-Neumann-Modell stellt der Speicher dar. Klassischerweise umfasst er den physischen Programm- und Datenspeicher des Rechners, von dem angenommen wird, dass er ein einheitlicher Speicher ist. Letzteres wird in vielen modernen Systemen durch das Vorhandensein von Caches und Architekturen wie NUMA bereits aufgeweicht. Für ein CPS bietet es sich an, analog zu der Betrachtung der Ausführungseinheiten auch die Grenze des Speichers deutlich nach außen zu verschieben: Physische Elemente können nicht nur Ausführungseinheiten sein, sondern haben auch eine speichernde Funktion. Diese repräsentiert sich in physikalischen und chemischen Eigenschaften des Elements und kann im System direkt oder indirekt im Rahmen von Operationen verwendet („gelesen“) werden. So ist der Bewegungsimpuls eines rollenden Balls ebenso eine gespeicherte Größe wie es die Temperatur eines Gegenstands oder der aktuelle Zustand eines chemischen Vorgangs sind.

Diese beiden Umstände haben eine Reihe von Konsequenzen für das BS: Das unterliegende physische System ist in hohem Maße heterogen und zudem je nach Zielstellung des Systems starken Änderungen unterworfen, da die Entscheidung, welche Teile der physischen Umgebung als Ausführungseinheiten bzw. Speicher gesehen werden, vom Zweck des Systems abhängt.

Das BS muss folglich ein hohes Maß an Konfigurierbarkeit unterstützen, die sich nicht nur auf die statische Definition verschiedenartiger Ausführungseinheiten und Speicherelemente beschränkt, sondern darüber hinaus auch dynamische Veränderungen erlaubt. Unabhängig von CPS stehen jedoch bereits BS, wie sie heute im Einsatz sind, vor ähnlichen (ebenfalls architekturbedingten) Herausforderungen, wie das Beispiel von GPGPUs (*General Purpose Graphic Processing Unit*) zeigt. Auch hier handelt es sich aus klassischer Sicht um Ein-/Ausgabegeräte, die aber zur Durchführung von Berechnungen herangezogen werden.

III. Betriebssystemkonzepte

Nachdem diskutiert wurde, wie sich ein CPS-Betriebssystem in ein Schichtenmodell einordnet, sollen jetzt grundlegende Konzepte bzw. Abstraktionen besprochen werden, die sich im Betriebssystembereich etabliert haben.

A. Prozess

In einem klassischen BS ist ein Prozess eine Virtualisierung des Prozessors. Die Einführung dieses Konzepts hat den Zweck, mehrere quasi-nebenläufige Ausführungsströme zuzulassen. Ein- und Ausgabegeräte werden dabei als Co-Prozessoren angesehen, bei denen der Ausführungsstrom häufig mehr Zeit verbringt als auf dem Hauptprozessor. Dies führte in klassischen Rechnern zum Leerlauf auf dem Hauptprozessor und damit zu einer mangelhaften Auslastung, die durch die Existenz von mehreren, (quasi-)nebenläufigen Prozessen kompensiert wird. Später wurden andere Konzepte mit Prozessen verknüpft, wie z.B. Speicher- und Ressourcenschutz. Aber im wesentlichen ist ein Prozess ein *Ausführungsstrom*, der nebenläufig zu anderen Ausführungsströmen abläuft.

In einem verteilten System wird dieses Prinzip der nebenläufigen Ausführungsströme fortgeführt. Der Ausführungsstrom kann zu einem entfernten Gerät wechseln, beispielsweise durch RPC oder durch Prozessmigration. Bei einem CPS sind physische Elemente Teil des Ausführungssystems, haben also die Funktion, die in einem klassischen Rechner (Co-)Prozessor oder I/O-Geräte einnehmen. Damit können in CPS Ausführungsströme – ähnlich wie in verteilten Systemen – auf andere Komponenten übergehen, ohne dass sie (unbedingt) auf die ursprüngliche Komponente zurückkehren. In verteilten Systemen nennt man diese Eigenschaft *Migrationsfähigkeit*. Migrationsfähigkeit ist nicht unbedingt trivial erreichbar, da damit eine Übertragung des lokalen Zustandes verbunden ist. Typischerweise handelt es sich dabei um einen Teil des Speicherabbildes, aber auch Ressourcen oder wenigstens die Möglichkeiten und Rechte des Zugriffes auf diese.

Im CPS vergrößert sich diese Schwierigkeit noch einmal, da auf den Zustand eines Prozessen, der ja auch Zustände der realen Welt beinhalten kann, nicht immer zugegriffen werden kann und er (oder ein *Abbild*) daher auch nicht ohne weiteres migrierbar ist.

B. Kontext

Der Zustand eines Prozesses wird im BS als *Kontext* bezeichnet. Wie bereits im Abschnitt III-A erwähnt, verlässt der Ausführungsstrom einer CPS-Anwendung mitunter den gut kontrollierbaren Rechner und tritt in den (aus Sicht des Rechners) schlechter kontrollierbaren physischen Bereich über. Daher gehören zum Zustand

einer CPS-Anwendung auch physische Zustandsgrößen (insbesondere der Teil des vom Prozess benutzten Speichers, der kein klassischer Speicher ist, sondern in Form physischer Eigenschaften außerhalb des Rechners vorliegt).

In klassischen eingebetteten Systemen werden solche Zustandsgrößen über Sensoren erfasst und auf Repräsentationen innerhalb des Rechners abgebildet (sogenannte *real-time images*, vgl. [18, Chap. 5]). Ein wesentlicher Unterschied zwischen „klassischen“ eingebetteten Systemen und CPS ist aber, dass physische Prozesse *Teil* des Systems und folglich physische Zustandsgrößen Teil des Kontextes eines Prozesses sind. Sie werden nicht (unbedingt) ständig beobachtbar sein oder (ständig) durch Abbildungen innerhalb der Rechner-Komponenten repräsentiert werden. Vielmehr sind sie als unorthodoxer Teil des Zustandes aufzufassen: Beispielsweise trägt der Impuls, den ein physischer Körper erhalten hat, eine Information, die nicht innerhalb von Rechnern gespeichert werden muss. Vielmehr kann sie über Sensormessungen *wiedergewonnen* werden, wenn eine weitere Verarbeitung innerhalb eines Rechners (also beispielsweise im Fall einer Migration des Prozesses von den physischen Ausführungselementen auf einen klassischen Prozessor) stattfinden soll.

Herausforderung bei diesem Ansatz ist, dass der physische Prozess zwar genauen „Zugriff“ auf den entsprechenden Teil hat und auch korrekt funktioniert (der rollende Ball nutzt genau seinen Impuls, um ihn an ein Hindernis weiterzugeben, so dass dieses in der beabsichtigten Weise umkippt), doch die Wiedergewinnung dieser Daten im Rechner ist entweder sehr aufwändig (ständige Beobachtung durch Sensoren) oder ungenau (da Ungenauigkeiten der Sensordaten die Extrapolation künftiger Zustände ohne ständige Beobachtung erschweren). Dementsprechend ist es sinnvoll, auf Seiten des Rechners eine Art Prognosedienst anzubieten, der die Wiedergewinnung dieser Art von Informationen zumindest erleichtert (indem beispielsweise extrapoliert wird, wo sich der Ball befinden könnte und damit die Suche beschleunigt wird).

C. Scheduling

Eine Anwendung in einem CPS hat reale physische Prozesse zum Gegenstand. Diese sind in der Regel über die Zeit veränderlich. Dies macht das CPS zu einem *Echtzeitsystem*.

Echtzeitsysteme wurden in den vergangenen Jahrzehnten ausgiebig erforscht und eine Reihe von Verfahren für das Scheduling von Prozessen unter Echtzeitanforderungen entwickelt. Die meisten der real verwendeten Verfahren gehen vom periodischen Taskmodell aus, das annimmt, dass alle Aktivitäten periodisch geschehen, jeweils eine bekannte maximale Ausführungszeit haben und bis zu einer gegebenen Frist (relativ zum Beginn der entsprechenden Periode) abgeschlossen sein müssen, damit keine zeitlichen Garantien verletzt werden. Damit ist das periodische Taskmodell eine stark vereinfachte Abbildung des Reglermodells. Die starken Vereinfachungen führen auf der einen Seite zu sehr einfachen Schedulingverfahren wie EDF oder RMS [19], haben auf der anderen Seite aber auch die Konsequenz, dass bei realen Anwendungen Kompromisse zwischen der Einfachheit des Modells und den Anforderungen geschlossen werden müssen. So werden beispielsweise sporadische Aktivitäten in die Regeln des Modells eingefügt, indem sie als periodisch mit ihrem Minimalabstand als Periode angenommen werden. Ähnliche Erweiterungen existieren für weitere Einschränkungen (insbesondere die Annahme, dass die Prozesse voneinander unabhängig sind oder die Einschränkung, dass es nur einen Prozessor gibt). Gemeinsam ist diesen Erweiterungen, dass sie die Komplexität der Lösungen, also der Schedulingverfahren, deutlich nach oben treiben. Für ein CPS, das schon durch seine Natur (Vielzahl von Ausführungseinheiten) verteilt ist und bei dem die einzelnen Prozesse auf vielfältige Weise voneinander abhängen, treffen diese Probleme in hohem Maße zu.

Zusätzlich zu dieser bereits hohen Komplexität sind CPS oder Teile von ihnen in vielen Fällen mobil. Mobilität bringt neben der Anforderung der Planung in der Zeit mit dem Raum noch eine weitere Dimension mit sich: Scheduling ist nun im Raum und der Zeit erforderlich, um zu entscheiden, welche Aktivität wann an welchem Ort durchgeführt werden muss, um die ebenfalls raumzeitlichen Anforderungen der Anwendung zu erfüllen.

D. Prozessinteraktion

IPC-Mechanismen klassischer BS gliedern sich in die Teilaspekte Koordination, Kommunikation und Kooperation. Dabei ist Koordination die gegenseitige zeitliche Abstimmung aller beteiligter Aktivitäten. So kann zum Beispiel ein exklusiver Zugriff auf einzelne Resourcen mittels Semaphoren realisiert werden. In CPS ist mitunter jedoch nicht nur der zeitlichen Ablauf zu koordinieren, sondern auch die Umwelt mit in die Koordination einzubeziehen. Dabei ist eine räumliche und zeitliche Koordination denkbar, bei der auch ein *Trade-Off* zwischen Raum und Zeit möglich ist. Die Grundkonstrukte können weiterhin genutzt werden, haben jedoch durch die höhere Dimension der Kontrolle eine andere Semantik. Ein kritischer Abschnitt könnte so auch als ein Raum betrachtet werden, der für eine gewisse Zeit in Anspruch genommen wird. Ähnliche Interpretationen sind auch für andere Koordinationskonstrukte möglich.

Kommunikation und Kooperation basieren auf dem Austausch bzw. der Verteilung von Information, die klassischer Weise durch byteorientierte Datenströme bzw. -blöcke kodiert wird. Bei genauerer Betrachtung stellt sich die Übermittlung von Information als Nutzung von physikalischen Wechselwirkungen heraus. Übertragen auf CPS-Betriebssysteme bedeutet dies, dass mit dem Konzept von Kommunikation und Kooperation weitere physikalische Vorgänge modelliert werden können. Das Passen eines Balls zwischen zwei Fußballrobotern stellt im Sinne der IPC eine Impulsübertragung zwischen den Robotern dar. Der Ball entspricht in diesem Fall dem Kanal, da er während des Übertragungsvorgangs den Impuls speichert.

E. Ein- und Ausgabe

Die klassische Abstraktion der Ein- und Ausgabe ist der Datenstrom. Dies kann – als Signal interpretiert – auch in einem CPS erhalten bleiben. Jedoch ändert sich der Ort, *wo* die Daten generiert/konsumiert werden: Während im klassischen System das elektronische Interface (ggf. durch A/D- oder A/D-Wandlung unterstützt) als Datenquelle bzw. Datensenke angesehen wird, kann das Interface eines CPS weit in den physischen Raum hineinreichen. Das klassische Interface am Rechner ist dann mitunter *transparent*. Genau betrachtet, ist dies aber nichts Neues: Auch bei klassischen Systemen schreibt/liest man von einer Festplatte (bzw. in der Regel vom logischen Betriebsmittel „Datei“), während die dazwischen liegenden Einheiten und Interfaces (Busse, Controller, etc.) transparent bleiben.

Es ist also die Aufgabe eines BS für CPS, entsprechende logische Interfaces zur Verfügung zu stellen. Aufgrund der Vielfältigkeit der physischen Welt stellt dies besondere Herausforderungen an die Konfigurierbarkeit des Systems (vgl. Abschnitt II-C).

Außerdem müssen I/O-Interfaces im stärkeren Maße virtualisiert werden. Dies wird besonders in CPS mit mobilen Knoten deutlich: Für die Anwendung ist es egal, durch welchen physischen Sensor ein Signal aus der Umwelt aufgenommen wird, solange dieses Signal zu einer bestimmten Zeit *an einem bestimmten Ort* gelesen werden kann. Es ist also möglich, das verschiedene physische Sensoren zu unterschiedlichen Zeiten das Signal aufnehmen, ohne dass die Anwendung den Wechsel der Sensoren veranlasst: Dieser Wechsel ist für die Anwendung transparent. Es ist die Aufgabe des BS, diese Transparenz herzustellen.

F. Namensdienst

Verteilte Systeme müssen auf eine Vielzahl von Ressourcen zugreifen, die über die Knoten des Systems verteilt sind. Um einen einfachen und einheitlichen Zugriff zu ermöglichen, existieren *Namens- oder Verzeichnisdienste*, die Informationen liefern, auf welche Weise auf bestimmte Objekte zugegriffen werden kann. In einem CPS mit seiner dynamischen Natur von Anwendungen, Ausführungseinheiten und Speicher ist eine Erweiterung dieses Konzepts erforderlich, um der unterschiedlichen Natur der Ressourcen Rechnung zu tragen. Diese Erweiterung betrifft dabei weniger das Konzept des Namensdiensts selbst, als viel mehr die Art der Informationen, die darin gespeichert werden, sowie ihre Gewinnung.

Komponenten des Systems wie die Ausführungseinheiten, der vorhandenen Speicher und die aktuellen Schnittstellen zur Umgebung – die ja nicht unbedingt statisch sind, sondern ständigen Änderungen unterworfen sein können, müssen vom Dienst erfasst werden. Entsprechend müssen solche Änderungen, die entweder durch Aktionen des Systems selbst (hinzukommende oder abgegebene Ressourcen) oder durch externe Einflüsse (ausfallende Ressourcen) ausgelöst werden, im Namensdienst widergespiegelt werden. Nicht nur die Ressourcen selbst und ihre Eigenschaften, sondern auch die Art des Zugriffs auf sie muss gespeichert werden. Zu einem physischen Ausführungselement kann so beispielsweise gespeichert werden, über welche Sensoren und Aktuatoren Beobachtungen oder Manipulationen möglich sind. Aus Sicht des BS kann dieser Teil des Namensdienstes dann verwendet werden, um z. B. für eine Prozessmigration ein mögliches Ziel zu suchen oder um Ressourcenanforderungen von Prozessen zu erfüllen.

Zum anderen werden Referenzen auf Objekte des BS in diesem Namensdienst gespeichert. Das betrifft insbesondere Virtualisierungen echter Ressourcen, also beispielsweise Prozesse oder Virtualisierungen von Ein-/Ausgabeschnittstellen. Unter Benutzung des Namensdienstes kann zu diesen Objekten der Ort und das aktuelle Zugriffsprotokoll (das sich ändern kann, wenn ein Prozess beispielsweise gerade auf ein physisches Objekt migriert worden ist) ermittelt werden.

IV. Diskussion

Die hier angestellten Betrachtungen zeigen, dass die „üblichen“ Abstraktionen und Konzepte, die für das BS in den 60er Jahren des letzten Jahrhunderts geprägt wurden, bis zu einem gewissen Grad auch für CPS eine Anwendung finden können. Jedoch müssen diese Abstraktionen und Konzepte z. T. von einigem „Ballast“ befreit werden, der sich im Laufe der Zeit angesammelt hat.

Das soll nicht bedeuten, dass völlig neue Abstraktionen und Konzepte für CPS und andere moderne Architekturansätze nicht nützlich wären. Im Gegenteil – allein die hier diskutierten Probleme zeigen die derzeitigen Grenzen auf.

Einige ausgewählte Beispiele sollen demonstrieren, wie solche neuen Ansätze aussehen könnten:

Die Erweiterung von LINDAs [20] bekannten Assoziativspeichermodell durch (zeitbehaftete) asynchrone Zugriffe in [21] ermöglicht eine Auflösung eines gemeinsamen Systemzustandes bei gleichzeitiger Ermöglichung von Echtzeitanforderungen.

Das experimentelle BS *Barrelfish* [22], [23] nutzt Constraintsolving zur Lösung von Konfigurationsproblemen bereits auf unterster Software-Ebene. Außerdem verzichtet es auf Speicherkonsistenz schon im nichtverteilten Fall (auf einem Rechner). Beide Ansätze, obwohl vollständig außerhalb jeden Bezugs auf CPS entwickelt, könnten sich als nützliche Ansätze für CPS-BS erweisen.

Das in [24] vorgestellte Konzept des *Distributed Active Objects* als „Hüll-Abstraktion“ für die durch eine CPS-Anwendung zu manipulierenden physischen Objekte nutzt das bekannte Konzept der aktiven Objekte, verschiebt aber die (abstrakte) Sicht des Anwendungsdesigners und ermöglicht eine systemische Sicht auf die CPS-Anwendung.

Welche konkreten Abstraktionen sich in künftigen CPS-Betriebssystemen etablieren werden, ist noch offen. Die Diskussion hier zeigt aber, dass einige „alte Bekannte“ dabei sein werden.

Literatur

[1] C. Gill and D. Niehaus, “Towards system software platforms for cyber-physical systems,” in *NSF Workshop on Cyber-Physical Systems*, 2006.

[2] E. A. Lee, “Cyber-physical systems - are computing foundations adequate?” in *NSF Workshop on Cyber-Physical Systems*, 2006.

[3] ——, “Cyber physical systems: Design challenges,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-8, Jan 2008.

[4] Y. Zhang, C. Gill, and C. Lu, “Reconfigurable real-time middleware for distributed cyber-physical systems with aperiodic events,” in *Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems*, ser. ICDCS ’08, 2008, pp. 581–588.

[5] C. Piechnick, “Entwicklung cyber-physikalischer Systeme am Beispiel des NAO Roboters,” in *Chemnitzer Linuxtage*, 2012.

[6] W. Stallings, *Operating Systems: Internals and Design Principles*. Prentice Hall, 2004.

[7] A. Tanenbaum, *Modern Operating Systems*. Prentice Hall, 2008.

[8] B. W. Lampson, “Hints for computer system design,” *IEEE Softw.*, vol. 1, pp. 11–28, January 1984.

[9] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to-end arguments in system design,” *ACM Trans. Comput. Syst.*, vol. 2, pp. 277–288, November 1984.

[10] O. Beucher, *MATLAB und Simulink – Scientific Computing*. Pearson Studium, 08 2006.

[11] C. Brooks, E. A. Lee, X. Liu, Y. Zhao, H. Zheng, S. S. Bhattacharyya, C. Brooks, E. Cheong, M. Goel, B. Kienhuis, E. A. Lee, J. Liu, X. Liu, L. Muliadi, S. Neuendorffer, J. Reekie, N. Smyth, J. Tsay, B. Vogel, W. Williams, Y. Xiong, Y. Zhao, and H. Zheng, “Ptolemy II - heterogeneous concurrent modeling and design in java,” EECS Department, University of California, Berkeley, Tech. Rep., 2005.

[12] E. A. Ashcroft, “LUCID Programming,” in *International Symposium on Methodologies for the Design and Construction of Software and Hardware Systems*, Rio de Janeiro, 1976.

[13] G. Wang, “The ChucK Audio Programming Language: An Strongly-timed and On-the-fly Environ/mentality,” Ph.D. dissertation, Princeton University, 2008.

[14] J. Brandes, S. Eichentopf, P. Elzer, L. Frevert, V. Haase, H. Mittendorf, G. Müller, and P. Riede, “PEARL, the concept of a process and experiment oriented programming language,” *elektronische datenverarbeitung*, vol. 10, pp. 429–442, 1970.

[15] J. von Neumann, “First draft of a report on the EDVAC,” University of Pennsylvania, Moore School of Electrical Engineering, Tech. Rep., 1945, http://qss.stanford.edu/~godfrey/vonNeumann/vnedvac.pdf.

[16] *Do 960 – Systembeschreibung*, Dornier System GmbH, 1975(?).

[17] A. I. Rubin, “Analog/hybrid–what it was, what it is, what it may be,” in *Fall Joint Computer Conference*, 1970, pp. 641–652.

[18] H. Kopetz, *Real-Time Systems*. Kluwer Academics, 1997.

[19] C. L. Liu and J. Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment,” *Journal of ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[20] D. Gelernter, “Generative communication in linda,” *ACM Transactions on Programming Languages and Systems*, vol. 7, pp. 80–112, 1985.

[21] G. Hackmann, C. Gill, and G.-C. Roman, “Towards a real-time coordination model for mobile computing,” in *Proceedings of the 12th Monterey conference on Reliable systems on unreliable networked platforms*, 2007, pp. 184–202.

[22] A. Schüpbach, S. Peter, A. Baumann, T. Roscoe, P. Barham, T. Harris, and R. Isaacs, “Embracing diversity in the barrelfish manycore operating system,” in *Workshop on Managed Many-Core Systems*, 2008.

[23] A. Baumann, S. Peter, A. Schüpbach, A. Singhania, T. Roscoe, P. Barham, and R. Isaacs, “Your computer is already a distributed system. why isn’t your OS?” in *12th Workshop on Hot Topics in Operating Systems*, 2009.

[24] D. Graff, J. Richling, T. M. Stupp, and M. Werner, “Distributed active objects - a systemic approach to distributed mobile applications,” in *8th IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems*, 2011.

Programmkomitee

Vorsitz
Oliver Bringmann
FZI Research Center for Information Technologies
Haid-und-Neu-Str. 10-14
76131 Karlsruhe

Mitglieder
D. Boers, *QNE*
M. Brandstetter, *Bosch GmbH*
M. Broy, *TU München*
S. Chakraborty, *TU München*
R. Drechsler, *DFKI Bremen*
K. Einwich, *FhG IIS/EAS Dresden*
M. Glaß, *U Erlangen-Nürnberg*
A. Graupner, *ZMDI*
E. Griese, *U Siegen*
C. Grimm, *TU Wien*
D. Große, *U Bremen*
M. Hahn, *IMMS Erfurt*
C. Haubelt, *U Rostock*
E. Kallenbach, *TU Ilmenau*
J. Kampe, *FH Jena*
M. Kasper, *TU Hamburg-Harburg*
J. Kelber, *FH Schmalkalden*
R. Laur, *U Bremen*
Y. Manoli, *U Freiburg*
J. Mehner, *TU Chemnitz*
K. Müller-Glaser, *KIT/FZI Karlsruhe*
R. Münzenberger, *inchron GmbH*
M. Porrmann, *HNI Paderborn*
P. Schneider, *FhG IIS/EAS Dresden*
F. Slomka, *U Ulm*
R. Sommer, *IMMS gGmbH*
G. Wachutka, *TU München*
Z. Wang, *KIT Karlsruhe*
G. Wirrer, *Continental AG*

Organisation des Workshops:
Karsten Einwich
Fraunhofer-Institut für Integrierte Schaltungen IIS
Institutsteil Entwurfsautomatisierung EAS
Zeunerstraße 38
01069 Dresden