Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 2

Johann-Nikolaus Andreae

HPC für embedded Systeme am Beispiel der Antriebsschlupfregelung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Kurzzusammenfassung

In dieser Ausarbeitung werden aktuelle Forschungsergebnisse zu der modellbasierten Entwicklung einer hardwarebeschleunigten Antriebsschlupfregelung untersucht. Hierbei wird auf die folgenden Themengebiete eingegangen: Modellierung von Fahrzeug und Streugeräten, Übertragung des Regelsystems in eine Hardware-Softwarepartitionierung, der Einsatz von FPGAs im Fahrzeug, Atmel CAP als SoC Platfform sowie Java Realtime als Alternative zum RTOS.

1 Einführung

Steuergeräte spielen in der Automobilindustrie eine immer größere Rolle. Die Entwicklung begann mit Tempomaten und Antiblockiersystemen (ABS) (ab 1979 in Serie) [Isermann (2006)]. Im Zuge der Weiterentwicklung sind weitere Systeme, wie Antriebsschlupfregelung (ASR) und elektronisches Stabilitätsprogramm (ESP) hinzugekommen. Heute wird an der Kollisionsverhütung und Fahrspurerkennung geforscht. Die im Serienfahrzeug integrierte Elektronik macht 20 – 25% des Kaufwertes aus, Tendenz steigend [Isermann (2006)].

Die Entwicklung eines Fahrzeuges beginnt mit kinematischen und dynamischen Modellen. Diese Modelle werden mit Simulationen auf ein optimales Fahrverhalten getrimmt. Parameter, die sich nicht mechanisch optimieren lassen, müssen später von den Streugeräten ausgeglichen werden. Zusätzlich haben die Streugeräte auch die Aufgabe, die Sicherheit des Fahrzeuges weiter zu erhöhen und den Fahrer in schwierigen Situationen zu unterstützen.

Im Rahmen der Fahrzeugsimulation werden auch die Assistenzsysteme mit Modellen getestet. Hierbei bildet das Fahrzeugmodell die Regelstrecke für die Regelkreissimulation. Das Simulationsmodell wird zur Zeit in einer zweiten Entwurfsphase in eine Softwareimplementierung übertragen. Wünschenswert wäre es, aus diesen Modellen direkt eine Software-Hardwareimplementation zu generieren.

Ein weiteres Ziel ist es, in mobilen Geräten, so auch in den Steuergeräten im Fahrzeug, den Stromverbrauch zu senken. Eine Möglichkeit dies zu tun ist, die Zahl der Mikrocontroller zu senken und ihre Effizienz zu steigern. Hierfür sollen Softwarefunktionen in die Hardware verlagert werden.

Softwarefunktionen lassen sich am einfachsten auf FPGAs in die Hardware auslagern und Testen. FPGAs haben konstruktionsbedingt einen höheren Stromverbrauch als ASICs, sind aber flexibler. Hier sind die Vor- und Nachteile beider abzuwägen.

Neben der Hardwareoptimierungen sind die Softwareoptimierungen nicht zu vernachlässigen. In Steuergeräte ist es wichtig das bestimmte Aufgaben zu einer bestimmten Zeit erledigt wer-

den. Hierbei ist meist die genaue Einhaltung der Zeitvorgaben erforderlich. In dieser Ausarbeitung wird auf Realtime Java als Alternative zum RTOS eingegangen.

Als Rahmen für diese Ausarbeitung diente die Antriebsschlupfregelung als Beispiel für ein Steuersystem im Fahrzeug.

2 Vorstellung vergleichbarer Arbeiten

Das Themengebiet des Masterprojektes setzt sich aus verschieden Teilgebieten zusammen. Diese Zusammenhänge der Themengebiete sind im Abb. 1 dargestellt.

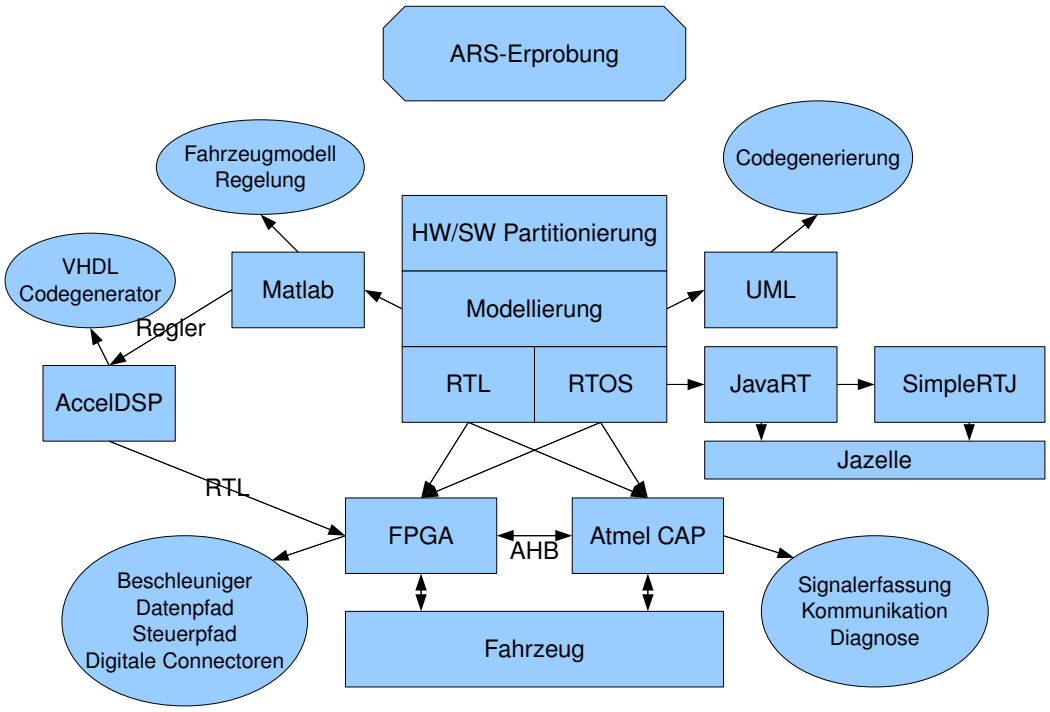

Abbildung 1: Arbeitsfelder der Vorbereitung des Masterprojektes

Im mittleren Block der Abbildung sind die Schritte für das Hardware-Software-Codedesign dargestellt. Der erste Schritt ist die Entscheidung über die Aufteilung in Hardware oder Softwareimplementierung. Hieraus werden in der Modellierungsphase die entsprechenden Modelle mit

MATLAB und UML erarbeitet (Abschnitt 2.2). Die letztendlich entstehenden Implementierungen in Hard- und Software werden in den FPGA bzw. den Mikrocontroller übertragen. Unter diesem Block befindet sich die Hardware. Sie besteht zum einen aus einem FPGA für eigene Hardwareimplementierungen (Abschnitt 2.4) und dem Atmel CAP als Mikrocontroller (Abschnitt 2.5). Der Erprobungsort des Systems ist das Kraftfahrzeug.

Links und rechts vom Mittelblock befinden sich die für das Teilgebiet relevanten Themengebiete, die hier behandelt werden. Dies ist auf der linken Seite die Übertragung des MATLAB-Modells mit AccelDSP in ein RTL-Schema (Abschnitt 2.3), auf der rechten Seite die Betrachtung von JavaRT als Alternative zum RTOS (Abschnitt 2.1). Auf das Thema der Softwaremodellierung mit der UML wird hier nicht weiter eingegangen.

2.1 Java Real Time

In einem Realtime-System hat das Betriebssystem nur wenige Aufgaben. Die Anzahl und die Funktion der auf dem System laufenden Tasks ist im Vorfeld bekannt. Das Scheduling lässt sich also auf ein Minimum beschränken. Die Hardware, auf der die Software läuft, ist auch vorher definiert. Eine große Unterstützung und Erkennung von Hardwarekomponenten ist auch nicht nötig. Was bleibt, ist eine Routine zur Initialisierung der Hardware, ein einfacher Scheduler und Speichermanagement und eine Bibliothek für den Zugriff auf die Hardwarefunktionen.

Die Java-VM kapselt für das Java-Programm alle Zugriffe auf das Betriebssystem. Sie stellt für das Programm das Betriebssystem dar. Die Idee hinter dem Einsatz eines realtime fähigen Java-VM auf einem embedded System ist, das RTOS durch die VM zu ersetzen. Für das Schedulling und die Hardwarezugriffe werden die Java Methoden genutzt. Die Hardwareinitialisierung wird in die VM implementiert.

Die Java-VM wird direkt beim Booten des Systems statt des RTOS gestartet und lädt ein einziges Java-Programm. In der Main-Methode des Programms werden die einzelnen Prozesse als Threads gestartet. Die VM führt das Schedulling mit den standard Java-Threads durch.

Die Realtime-VM von Sun setzt ein Betriebssystem wie Solaris oder Linux mit Realtime-Erweiterung als Unterbau voraus. Eine VM die ohne OS auskommt, ist simpleRTJ[1] von RTJ Computing. Die Implementierung dieser VM ist für eine Vielzahl von Mikrocontrollern erhältlich und kann für Forschungszwecke kostenlos im Sourcecode von der Internetseite des Herstellers heruntergeladen werden. Das aktuelle Release[2] ist vom 2003-09-22, was die Vermutung nahe legt, dass die VM nicht mehr aktiv weiterentwickelt wird.

[1] Technical Brief http://www.rtjcom.com/downfile.php?f=techpdf Stand 2009-06-20
[2] Stand: 2009-04-28

Die Firma ARM hat Jazelle entwickelt, um Java auf dem ARM-Prozessor effizient auszuführen. Jazelle besteht aus zwei Teilen: Einer hardwareseitigen Byte-Code-Ausführung und einem „Just in Time“-Compiler. Die genauen Spezifikationen für eigene VM Implementierungen stellt ARM im Internet nicht zur Verfügung, diese sind Laut Wikipedia[3] nur sehr unvollständig. Für eine genaue Einschätzung des Arbeitsaufwandes sind weitere umfangreiche Recherchen erforderlich.

Neben Jazelle gibt es weitere Projekte, die sich mit der Implementierung von Java-Prozessoren auseinander setzen. In der Dissertation [Pfeffer (2004)] geht es um die Entwicklung eines echtzeitfähigen multithredded Mikrocontrollers. Schöberl (2005) beschreibt in seiner Dissertation die Entwicklung eines embedded echzeitfähigen Java Prozessors. Dieser Prozessor ist als OpenSource (GPLv3) verfügbar. Eine Liste von Forschungsprojekten an anderen Universitäten, wie z.B. picoJava (Sun), Komodo, FemtoJava befinden sich bei Wikipedia[4].

2.2 Modellierung des Fahrzeuges

Das Fahrzeug wird mit Modellen simuliert, um das Verhalten des Fahrzeuges schon vor der Fertigung bestimmen zu können. Auch die Regelsysteme des Fahrzeuges lassen sich in einer solchen Simulation schon vor der Entwicklung mit Modellen testen. Die Simulation von Fahrzeug und Regelsystemen wird mit dem „Hardware in the Loop“-Verfahren (HiL) durchgeführt. Grundlagen zu HiL und der modellbasierten Optimierung werden in [Gühmann und Wolter (2005)] und [Lorenz (2008)] beschrieben.

Der Tagungsband „Fahrdynamik-Regelung“ enthält mehrere Artikel zur Entwicklung von Fahrzugmodellen. Im Teil A des Tagungsbandes wird auf die Grundlagen der Modellbildung und Simulation eingegangen. Der Teil B geht es um die Längs- und Querdynamikregelung. In den weiteren Teilen wird noch auf die aktive Fahrwerksregelung und die Spurhalte- und Einparkassistenen eingegangen.

Für den hier bearbeiteten Einsatzzweck ist vor allem der Artikel (Drogies, 2006, Objektorientierte Modellbildung des fahrdynamischen Verhaltens mit MODELICA) interessant. Er beschreibt den Aufbau des Fahrzeugmodells mit Hilfe von MODELICA, welches ähnlichen Funktionen wie MATLAB hat. Dieser Artikel wird durch die Grundlagen aus den Artikeln (Börner, 2006, Modellierung, Analyse und Simulation der Fahrzeugquerdynamik) und (Schorn, 2006, Modelle zur Beschreibung des Fahrzeugverhaltens) unterstützt.

In der Designspezifikation [Chen u. a. (2007)] wird eine Antriebsschlupfregelung für einen Formula Student Rennwagen entwickelt. Am Anfang der Ausarbeitung werden verschiedene im

[3] http://en.wikipedia.org/wiki/Jazelle Zugriff: 2009-04-28
[4] http://en.wikipedia.org/wiki/Java_processor Zugriff: 2009-06-11

Handel erhältliche Antriebsschlupfregelungen aufgelistet. Anschließend wird auf die hardwareseitige Umsetzung eingegangen. In Blockdiagrammen wird Dargestellt welche Sensorwerte in welchem Modul ausgewertet werden. Die Funktionalität der Module wird in Software umgesetzt, da für eine Implementierung im FPGA die Zeit fehlt. Wie die Implementierung der Software aus sieht und wie auf die Sensorwerte in den Modulen reagiert wird ist nicht beschrieben.

2.3 Übertragung des Regelsystems in die HW/SW-Partitionierung

In der Hardware-Software-Partitionierung wird festgelegt, welche Teile der Funktionalität in Hardware und welche in Software realisiert werden. Die Entscheidung hängt von der verwendeten Hardware und den Anforderungen der Funktionalität ab. Kriterien für die Hardware-Software-Partitionierung werden in [Vahid und Stitt (2007)] und [Kolbe (2008)] erläutert.

Bei der Übertragung eines MATLAB-Modells eines Steuergerätes in die Hardwareimplementierung ist zu beachten, dass MATLAB mit Fließkommazahlen rechnet. In der Hardwareimplementierung werden Festkommazahlen verwendet. Dies hat zur Folge, dass die Simulationsergebnisse von denen der Implementierung abweichen [Hill (2006)].

Für die Konvertierung des MATLAB-Modells in ein RTL-Schema ist ein Compiler von Vorteil Auf diesem Gebiet wurde an der Northwestern University im Rahmen das MATCH (MATlab Compiler for Heterogeneous computing systems) Projektes[5] geforscht. Die Veröffentlichungen [Haldar u. a. (2001b)], [Banerjee u. a. (2000)], [Harriss u. a. (2002)], [Haldar u. a. (2001a)] und weitere beschreiben die verschiedenen Aspekte des Compilers. Aktuell existiert die Projektseite noch, wurde aber laut Datumsangabe auf der Seite zuletzt am 2002-02-25 geändert. Mindestens einer der betreuenden Professoren ist auch an der Gründung der Firma AccelChip beteiligt. In Banerjee u. a. (2004) wird das VHDL/Verilog Ergebnis, welches das Programm AccelFPGA der Firma AccelChip erzeugt genauer untersucht. Hierbei zeigte sich, dass durch den Einsatz von AccelFPGA eine Menge Entwicklungszeit sparen lässt.

Das automatische Konvertieren ist auch für die FPGA-Hersteller von großem Interesse. Es macht den Einsatz von FPGAs attraktiver, da man kein speziell geschultes Personal für die Konvertierung benötigt. Xilinx hat hierfür die Firma AccelChip übernommen, welche sich auf dieses Gebiet spezialisiert hat. In dem Artikel [Cigan und Lall (2005)] aus dem „DSP magazin“ wird anhand des Kalman-Filters gezeigt, wie sich MATLAB-Modell in die Hardwareimplementierung übertragen lässt. In Abb. 2 ist zu sehen, wie die Tools von Xilinx in einander greifen um die Entwicklung zu unterstützen.

[5] http://www.ece.northwestern.edu/cpdc/Match/Match.html

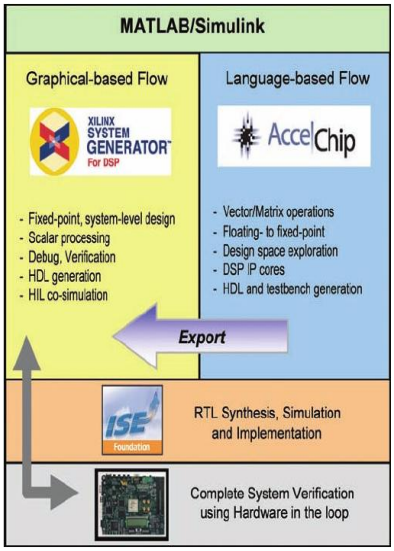

Abbildung 2: AccelChip, SystemGenerator, ISE Workflow. Quelle: Cigan und Lall (2005)

Der Workflow ist wie folgt: Mit MATLAB wird der Algorithmus mit mathematischen Gleichungen definiert. Anschließend wird mit AccelChip der Algorithmus in ein Register-Transfer-Level-Schema transformiert. Hierbei findet auch die float to fixpoint Konvertierung statt. Mit dem SystemGenerator wird im Anschluss der VHDL-Code generiert, welcher mit dem ISE an den Prozessor angebunden wird.

2.4 FPGAs im Fahrzeug

FPGAs haben zur Zeit nur einen geringen Anteil an der KFZ-Elektronik. Auf Nachfrage erklärte ein Audi-Mitarbeiter bei einem Werksbesuch, dass FPGAs zur Zeit nicht im Fahrzeug eingesetzt werden. Dies wird daran liegen das in großen Stückzahlen ASICs günstiger sind.

Ungeachtet dessen sehen die FPGA Hersteller einen Markt für ihre Produkte, was ich u.a. aus den Veröffentlichungen in der Xcell Junal schließe [Bagni u. a. (2008), Parnell (2005) und weitere]. Zielgebiet für die FPGAs im Fahrzeug sind vor allem die neueren Assistenzsysteme wie die Fahrspurerkennung. Hier ist zum einen die Hürde geringer, da es sich um Neueinwicklungen handelt, aber auch der Vorteil der FPGAs durch die beschleunigte Bildverarbeitung besonders groß. In dem Artikel (Bagni u. a., 2008, Building Automotive Driver Assistance System Algorithms with Xilinx FPGA Platforms) wird der Weg der Entwicklung einer Fahrspurerkennung beschrieben. Gestartet wird mit einem Simulink-Modell, welches mit dem Xilinx System Generator in eine FPGA-Hardwareimplementierung transformiert wird.

Der Forschungsbericht [Chujo (2002)] befasst sich mit der Konfiguration, sowohl Hauptcontroller als auch den Backupcontroller durch einen einzigen FPGA zu ersetzen. Hierzu soll der FPGA im Fehlerfall mit dem Backupsystem rekonfiguriert werden. Mit diesem Ansatz könnte man nicht nur unterschiedliche Softwareimplementationen sondern auch unterschiedliche Hardwarearchitekturen ohne zusätzliche Mikrocontroller auf der Platine implementieren. Zu beachten ist bei diesem Ansatz die Zeit, die für das Rekonfigurieren des FPGAs nötigt ist. In dieser Zeit steht das System nicht zur Verfügung. Unabhängig von diesem Konzept muss wie zuvor auch ein Backupkonzept für die am Controller angeschlossenen Sensoren und Aktoren entwickelt werden.

Der einzige Artikel, der über den Einsatz von FPGAs in den Steuersystemen des Fahrzeugs berichtet, stammt aus dem Bereich der Formel 1. In [Boland (2003)] wird der Einsatz von FPGAs beim Team BMW Williams im Jahre 2003 vorgestellt. In der Formel 1 ändern sich jährlich die Regeln, so dass die Steuersysteme jedes Jahr angepasst werden müssen. Dies lässt sich mit FPGAs besonders einfach bewerkstelligen, da nicht jedes mal ein neues Hardwaremodul entwickelt werden muss, sondern nur die Konfiguration des FPGAs geändert.

2.5 Atmel CAP als SoC-Plattform

Die AT91CAP-Serie der Firma Atmel ist ein Angebot zur Herstellung von ASICs. Die Basis des SoC bildet ein ARM7/9-Prozessor, der mit einer Anzahl von Standartschnittstellen wie SPI, I2C, USB, etc. ausgestattet ist [Bischop (2008)]. Zusätzlich ist in dem Chip ein Metalblock enthalten, der nach den Kundenvorgaben gefertigt wird. Für die Fertigung des Metalblocks stehen zum einen die von Atmel angebotenen IP-Blöcke zur Verfügung, zum anderen lassen sich aber auch eigene Funktionsblöcke integrieren. Es ist auch möglich, in den Metalblock einen zweiten Prozessor zu Implementieren. Der Metallblock wird bei der Produktion fest verdrahtet und lässt sich nicht wie ein FPGA rekonfigurieren.

Für das Testen des Zusammenspiels zwischen dem Mikrocontroller und den eigenen Interfaces gibt es eine Version, die statt dem Metalblock mit einem Interface für einen FPGA ausgestattet ist. Über dieses Interface ist der FPGA direkt mit dem Advanced High-performance Bus (AHB) und dem Advanced Peripheral Bus (APB) des ARM-Prozessors verbunden.

Der Vorteil für die Entwicklung eines Mikrocontrollers für die Serienfertigung ist, dass nur der Metalblock kundenspezifisch gefertigt werden muss, was die Kosten reduziert. Aber auch für die Forschung hat dieses Konzept einen Vorteil. Es bietet die Möglichkeit eigene Logikblöcke an einen Mikrocontroller anzuschließen, statt diesen im FPGA mit zu integrieren.

3 Abgrenzung zu eigenen Arbeitszielen und Methoden

Ziel der Arbeit ist es, den Weg der modellbasierten Entwicklung von embedded high performance Systemen zu untersuchen. Die Implementierung und die Erprobung wird anhand der Antriebsschlupfregelung für ein Kraftfahrzeug durchgeführt. Die vorgestellten Arbeiten bilden die Grundlage für die einzelnen Teilaufgaben des Projektes. Die Verknüpfung der Teilaufgaben muss selber entwickelt werden. Eine übergreifende Arbeit ist nicht vorhanden.

Java Realtime stellt ein interessantes Anwendungsfeld für den embedded Bereich dar. Die Forschung und Entwicklung in diesem Bereich scheint aber nicht mehr aktiv verfolgt zu werden. Aufgrund des Forschungsstandes ist noch eine menge Arbeit erforderlich, um Realtime Java im Steuergerät einsetzen zu können. Die Untersuchung in wie weit sich z.B. simpleRTJ mit Jazelle beschleunigen läßt und welche Auswirkungen es auf die Realtimeimplementierung hat, lässt sich aufgrund der Unzugänglichkeit der Jazelle-Spezifikationen nur schwer einschätzen. Aufgrund des zu erwartenden Aufwandes wird dieses Anwendungsszenario nicht weiter verfolgt.

Für die Modellierung des Fahrzeuges und der Antriebsschlupfregelung liefern die in Abschnitt 2.2 vorgestellten Artikel die Grundlagen. Es werden aber keine tiefgehenden Details oder fertige Modelle geliefert. Das Modell der Antriebsschlupfregelung muss durch eigene Überlegungen und Erprobungen entwickelt und verfeinert werden.

Die Übertragung von MATLAB-Modellen in ein RTL macht mit dem Xilinx Tool AccellChip einen benutzbaren Eindruck und muss im praktischen Test erprobt werden. Eine genaue Untersuchung zu der internen Funktionsweise des Compilers ist nicht geplant. Die Informationen zu der Fließkomma-/Festkommazahlenumwandlung sind aber für die Verifikation des Compilerergebnisses wichtig.

Ein Entwicklungsboard mit einem AT91CAP und einem Xilinx FPGA bietet die Grundlage der Implementations- und Testumgebung. Auf diesem Board wird die Antriebsschlupfregelung getestet. Nach erfolgreichem Abschluss der Tests besteht die Möglichkeit die Antriebsschlupfregelung in einem HAWKS Formula Student Rennwagen zu erproben.

4 Zusammenfassung

In den vorangegangenen Abschnitten wurden die einzelnen Teilgebiete der modellbasierten Entwicklung einer Antriebsschlupfregelung dargestellt. Realtime Java stellt eine Alternative zum RTOS dar, ist aber zur Zeit nicht in einem Zustand, um es einfach einzusetzen. Es muss mindestens noch auf die eigene Architektur portiert werden. Bei der Modellierung des Fahrzeuges ist nicht viel Literatur vorhanden, vor allem was die genaue Funktion von Steuergeräten

wie die Antriebsschlupfregelung betrifft. Bei der Automobielindustrie fällt eine Menge unter das Geschäftsgeheimnis. Das Modell für die Antriebsschlupfregelung muss selber entwickelt werden. Die Forschungsergebnisse für die Übertragung der MATLAB-Modelle in ein RTL-Schema wurde das Produkt Accell DSP entwickelt, welches von Xilinx weiterentwickelt wird. Wie gut das Programm im praktischen Einsatz funktioniert muss noch erprobt werden. FPGAs spielen im Serienfahrzeug zur Zeit nur eine untergeordnete Rolle, was wohl hauptsächlich an dem hohen Aufwand der Verifikation der korrekten Funktion des Steuergerätes liegt. Aber auch der etwas höhere Stromverbrauch der FPGAs könnte ein Grund für die Zurückhaltung sein. In der Formel 1 haben sie aber schon vor Jahren ihren Platz gefunden, da sie hier ihren Flexibilitätsvorteil voll ausspielen können. Die Atmel CAP-Mikrocontroller stellen eine Möglichkeit dar, einen ARM-Prozessor um eigene Schnittstellen zu erweitern ohne die hohen Fertigungskosten eines ASICs zu haben.

Die einzelnen Teilgebiete für sich machen scheinen sich für das Anwendungszenario einer Antriebsschlupfregelung einsetzen zu lassen. Das Zusammenspiel der einzelnen Komponenten muss durch Simulation und anschließendem Test in Fahrzeug noch erprobt werden.

Literatur

[Bagni u. a. 2008] BAGNI, Daniele ; MARZOTTO, Roberto ; ZORATTI, Paul: Building Automotive Driver Assistance System Algorithms with Xilinx FPGA Platforms. In: *Xcell Jurnal* (2008), Nr. 66. – URL http://www.xilinx.com/publications/xcellonline/xcell_66/xc_pdf/p20-26_66_F_XiASIM1.pdf

[Banerjee u. a. 2004] BANERJEE, Prithviraj ; HALDAR, Malay ; NAYAK, Anshuman ; KIM, Victor ; SAXENA, Vikram ; PARKES, Steven ; BAGCHI, Debabrata ; PAL, Satrajit ; TRIPATHI, Nikhil ; ZARETSKY, David ; ANDERSON, Robert ; URIBE, Juan R.: Overview of a compiler for synthesizing MATLAB programs onto FPGAs. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 12 (2004), März, Nr. 3, S. 312–324. – ISSN 1063-8210

[Banerjee u. a. 2000] BANERJEE, Prithviraj ; SHENOY, N. ; CHOUDHARY, Alok ; HAUCK, S. ; BACHMANN, C. ; HALDAR, Malay ; JOISHA, P. ; JONES, A. ; KANHARE, A. ; NAYAK, Anshuman ; PERIYACHERI, S. ; WALKDEN, M. ; ZARETSKY, David: A MATLAB compiler for distributed, heterogeneous, reconfigurablecomputing systems. In: *2000 IEEE Symposium on Field-Programmable Custom Computing Machines*, August 2000, S. 39–48. – ISBN 0-7695-0871-5

[Bischop 2008] BISCHOP, Peter: *Efficient System-onChip Development using Atmel's CAP Customizable Microcontroller*. April 2008. – White Paper

[Boland 2003] BOLAND, Liza: Formula 1 Racing: The Xilinx Advantage. In: *Xcell Journal* (2003). – URL http://www.xilinx.com/publications/xcellonline/xcell_47/xc_pdf/xc_formula47.pdf

[Börner 2006] BÖRNER, Marcus: Modellierung, Analyse und Simulation der Fahrzeugquerdynamik. In: ISERMANN, Rolf (Hrsg.): *Fahrdynamik-Regelung*. Vieweg, 2006, S. 47–70. – ISBN 978-3-8348-9049-8

[Chen u. a. 2007] CHEN, Paul ; GANSEN, Candice ; HERSHBERGER, Mike ; KNAPP, Anthony ; OLDEN, Jeff: *Electronic Traction Control System*. Dezember 2007

[Chujo 2002] CHUJO, Naoya: Fail-safe ECU System Using Dynamic Reconfiguration of FPGA. In: *R&D Review of Toyota CRDL* (2002), April, Nr. 37

[Cigan und Lall 2005] CIGAN, Eric ; LALL, Narinder: Integrating MATLAB Algorithms into FPGA Designs. In: *DSP magazine* (2005), Oktober, Nr. 1, S. 37–39. – URL http://www.xilinx.com/publications/magazines/dsp_01/xc_pdf/p37-39_dsp-accelchip.pdf

[Drogies 2006] DROGIES, Stefan: Objektorientierte Modellbildung des fahrdynamischen Verhaltens mit MODELICA. In: ISERMANN, Rolf (Hrsg.): *Fahrdynamik-Regelung*. Vieweg, 2006, S. 71–91. – ISBN 978-3-8348-9049-8

[Gipper und Dingee 2007] GIPPER, Jerry ; DINGEE, Don: Know your OS options. In: *Embedded Computing Design* (2007). – URL http://www.embedded-computing.com/pdfs/OSP3.Mar07.pdf

[Gühmann und Wolter 2005] GÜHMANN, Clemens (Hrsg.) ; WOLTER, Thieß-Magnus (Hrsg.): *Simulation und Test in der Funktions- und Softwareentwicklung für die Automobilelektronik*. 1. Expert-Verlag, März 2005. – ISBN 978-3-8169-2491-3

[Haldar u. a. 2001a] HALDAR, Malay ; NAYAK, Anshuman ; CHOUDHARY, Alok ; BANERJEE, Prithviraj: A system for synthesizing optimized FPGA hardware from MATLAB. In: *ICCAD '01: Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*. Piscataway, NJ, USA : IEEE Press, 2001, S. 314–319. – ISBN 0-7803-7249-2

[Haldar u. a. 2001b] HALDAR, Malay ; NAYAK, Anshuman ; SHENOY, N. ; CHOUDHARY, Alok ; BANERJEE, Prithviraj: FPGA hardware synthesis from MATLAB. In: *2001. Fourteenth International Conference on VLSI Design*, August 2001, S. 299–304. – ISBN 0-7695-0831-6

[Halfmann und Holzmann 2003] HALFMANN, Christoph ; HOLZMANN, Henning: *Adaptive Modelle für die Kraftfahrzeugdynamik*. Springer, 2003 (VDI-Buch). – ISBN 978-3-540-44278-3

[Harriss u. a. 2002] HARRISS, Tim ; WALKE, Richard ; KIENHUIS, Bart ; DEPRETTERE, Ed: Compilation From Matlab to Process Networks Realized in FPGA. In: *Design Automation for Embedded Systems* 7 (2002), November, Nr. 4, S. 385–403. – ISSN 1572-8080

[Hill 2006] HILL, Tom: Floating- to Fixed-Point Conversion of MATLAB Algorithms Targeting FPGAs. In: *DSP magazine* (2006), Mai, Nr. 2, S. 32–35. – URL http://www.xilinx.com/publications/magazines/dsp_02/xc_pdf/p32-35_2dsp-float.pdf

[Isermann 2006] ISERMANN, Rolf: Das mechatronische Kraftfahrzeug. In: ISERMANN, Rolf (Hrsg.): *Fahrdynamik-Regelung*. Vieweg, 2006, S. 1–26. – ISBN 978-3-8348-9049-8

[Kolbe 2008] KOLBE, Felix: *System-on-a-Chip Konzepte für die Telemetrie- und Steuerungskomponenten in einem Formula Student Rennwagen*, Hochschule für Angewante Wissenschaft Hamburg, Ausarbeitung Anwendungen 1, Dezember 2008

[Kubitschek und Johnson 2007] KUBITSCHEK, Tim ; JOHNSON, Jay: Customizable ARM Processor-based MCUs Trackle DSP Algorithms. In: *Atmel Applications On-Line Journal* (2007), Nr. 9

[Lorenz 2008] LORENZ, Daniel: *HIL basierte Kalibrierung anhand des HAWKS Rennwagens*, Hochschule für Angewante Wissenschaft Hamburg, Ausarbeitung Anwendungen 1, Dezember 2008

[Parnell 2005] PARNELL, Karen: The Changing Face of Automotive ECU Design. In: *Xcell Journal* (2005)

[Pfeffer 2004] PFEFFER, Matthias: *Ein echtzeitfähiges Java-System für einen mehrfädigen Java-Mikrocontroller*, Universität Augsburg, Dissertation, März 2004

[Schlingloff u. a. 2004] SCHLINGLOFF, Holger ; CONRAD, Mirko ; DÖRR, Heiko ; SÜHL, Carsten: *Modellbasierte Steuergerätesoftwareentwicklung für den Automobilbereich*. 2004. – URL http://www2.informatik.hu-berlin.de/~hs/Publikationen/2004_GI-Automotive_Schlingloff-Conrad-Doerr-Suehl_ModellbasierteSteuergeraetesoftwareentwicklung.pdf

[Schöberl 2005] SCHÖBERL, Martin: *JOP: A Java Optimized Processor for Embedded Real-Time Systems*, Vienna University of Technology, Dissertation, Januar 2005. – URL http://www.jopdesign.com/thesis/thesis.pdf

[Schorn 2006] SCHORN, Matthias: Modelle zur Beschreibung des Fahrzeugverhaltens. In: ISERMANN, Rolf (Hrsg.): *Fahrdynamik-Regelung*. Vieweg, 2006, S. 27–46. – ISBN 978-3-8348-9049-8

[Vahid und Stitt 2007] VAHID, F. ; STITT, G.: Hardware/Software Partitioning. In: HAUCK, Scott (Hrsg.) ; DEHON, Andre (Hrsg.): *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*. Morgan Kauffman, November 2007 (Systems on Silicon). – ISBN 978-0-12-370522-8