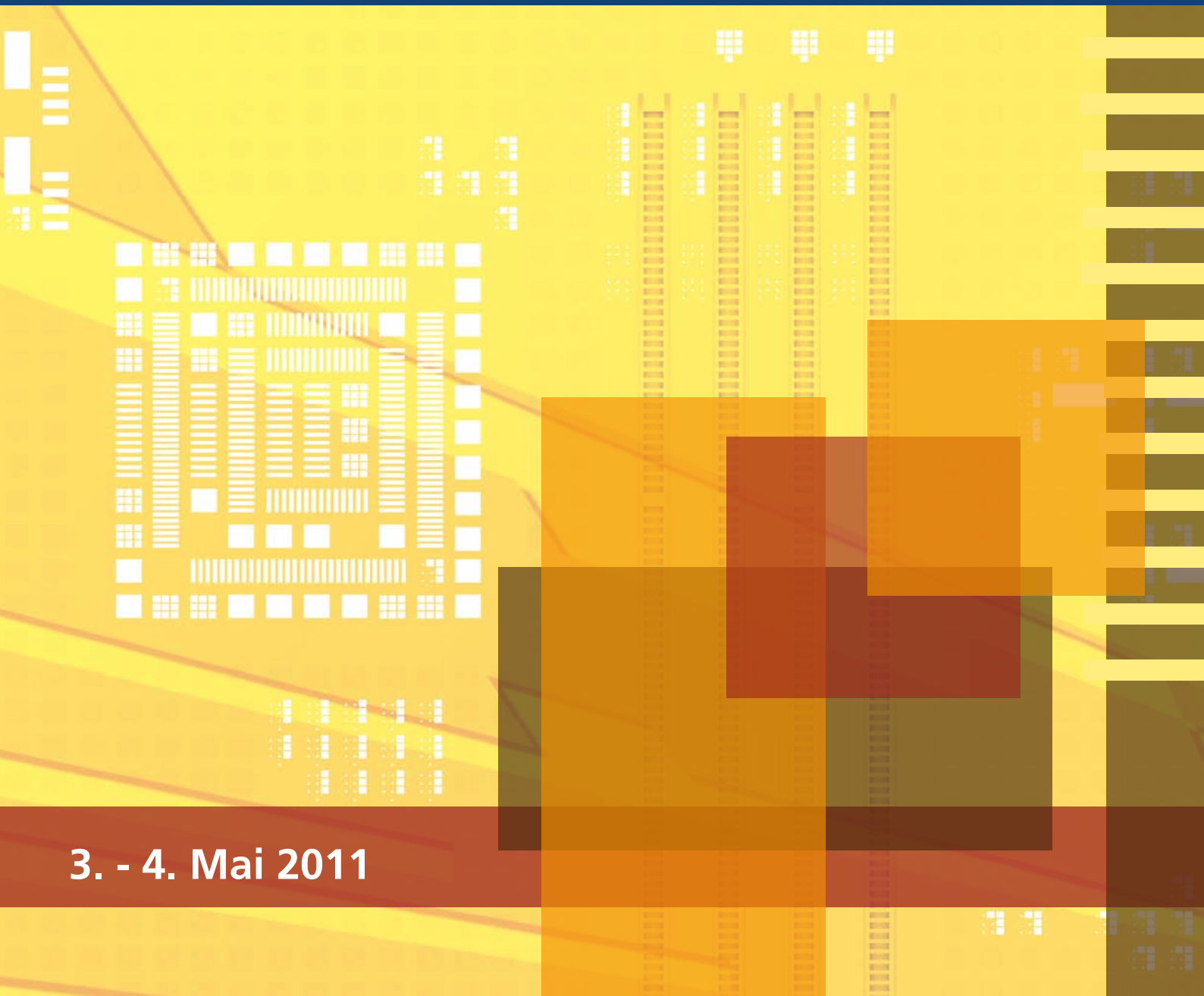
Fraunhofer
IIS

FRAUNHOFER-INSTITUT FÜR INTEGRIERTE SCHALTUNGEN ISS
INSTITUTSTEIL ENTWURFSAUTOMATISIERUNG EAS

DASS 2011
DRESDNER ARBEITSTAGUNG SCHALTUNGS- UND SYSTEMENTWURF



3. - 4. Mai 2011

DASS 2011 - Tagungsband

Dresdner Arbeitstagung Schaltungs- und Systementwurf

03. – 04. Mai 2011

Günter Elst (Hrsg.)
Thomas Klotz (Hrsg.)

TECHNISCHE UNIVERSITÄT DRESDEN

Technische Universität Dresden
Fakultät Elektrotechnik und Informationstechnik
Fakultät Informatik

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS
Institutsteil Entwurfsautomatisierung EAS

Kontaktadresse:

Fraunhofer-Institut für Integrierte Schaltungen IIS
Institutsteil Entwurfsautomatisierung EAS
Zeunerstr. 38, 01069 Dresden (Tagungsort)
Tel: +49 351 4640-700
Fax: +49 351 4640-703
E-Mail: dass@eas.iis.fraunhofer.de
www.eas.iis.fraunhofer.de

© Fraunhofer IIS, Institutsteil EAS 2011
Die Autoren sind für den Inhalt der Beiträge dieses Tagungsbandes verantwortlich.

Layout: Fraunhofer IIS, Institutsteil EAS
Hintergrund Titelgrafik: © MEV

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: 978-3-8396-0259-1

© by FRAUNHOFER VERLAG, 2011
Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 800469, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Tel: +49 711 970-2500
Fax: +49 711 970-2508
E-Mail: verlag@fraunhofer.de
http://verlag.fraunhofer.de

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 Verlages 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 der Verlag keine Gewähr für Richtigkeit, Vollständigkeit oder Aktualität übernehmen.

Vorwort

Sehr geehrte Damen und Herren,

die Dresdner Arbeitstagung Schaltungs- und Systementwurf hat sich als Forum für die Präsentation aktueller Arbeiten und Ergebnisse zum Entwurf sowie zur Entwurfsautomatisierung etabliert.
Wissenschaftler und Ingenieure aus der Industrie tauschen sich hier über aktuelle Entwicklungen aus. Dabei stehen zukünftige elektronische und heterogene Systeme und die damit verbundenen Anforderungen an die Entwurfsunterstützung im Mittelpunkt.

Die Arbeitstagung hat in diesem Jahr

Methoden, Werkzeuge und Verfahren für den anwendungsspezifischen Systementwurf

als Schwerpunktthema. Beiträge hierzu betrachten insbesondere neuartige Entwurfsmethoden. Dazu gehören beispielsweise Ansätze zum modellbasierten Design von heterogenen Systemen, die das Ziel haben, die zunehmende Komplexität im Entwurfsprozess zu beherrschen.

Das Tagungsprogramm bietet den Teilnehmern in gewohnter Qualität interessante und aktuelle Beiträge über neue Lösungen zum Entwurf komplexer Schaltungen und Systeme. Neben konstruktiven Verfahren werden Methoden des technologiegerechten Entwurfs unter den Randbedingungen Robustheit, funktionale Zuverlässigkeit, Wiederverwendbarkeit sowie Rekonfigurierbarkeit vorgestellt.

Als Veranstalter laden wir Sie herzlich zur dieser Arbeitstagung nach Dresden ein. Wir bedanken uns bei den Autoren für die Bereitstellung der Beiträge und bei den Teilnehmern für ihr Interesse.

René Schüffny
Rainer Spallek
Ronald Tetzlaff
Günter Elst

Dresden, Mai 2011

Inhaltsverzeichnis

Keynote Gunter Strube

Entwurfsautomatisierung: Von der Forschung bis zum Standard

This keynote is meant to be inspirational for all those, who are developing and presenting technical approaches to IC Design issues. While some presented solutions are likely to forgotten soon, some of the solutions presented at this conference could become tools that will be applied in IC design in the future.

Firstly we will look at the main stages that represent critical milestones in the growth of the product life cycle, as the title of this presentation says "from research to standard". As an example we will look at the evolution of a solution that can truly be considered a standard today SPICE circuit simulation. The 40 year history shows the stages of adoption and evolution.
Then we will take a look at the development of a solution which has come out of European Research Projects – probably just like many of the presentations at this conference are derived from the same kind of projects today.

The history of this solution – the WiCkeD tool suite for analog sizing - is also quite long too. The adoption curve is slower than SPICE, but it has steadily evolved through the stages to reach commercial success.

The somewhat visionary question is asked if this technology has the potential to ever become standard. By taking a look the fundamental concept behind the solution and how it addresses key design requirements of today, we conclude that this potential exists especially in the context of supportive industry trends.

This shall be seen as food for thought and no one knows the further evolution, but hopefully it's motivation not only for further WiCkeD adoption, but for anyone working on current solutions to see that success in EDA is possible in Europe.

Entwurfsautomatisierung: Von der Forschung bis zum Standard

DASS 2011
Dresden, 3.5.2011

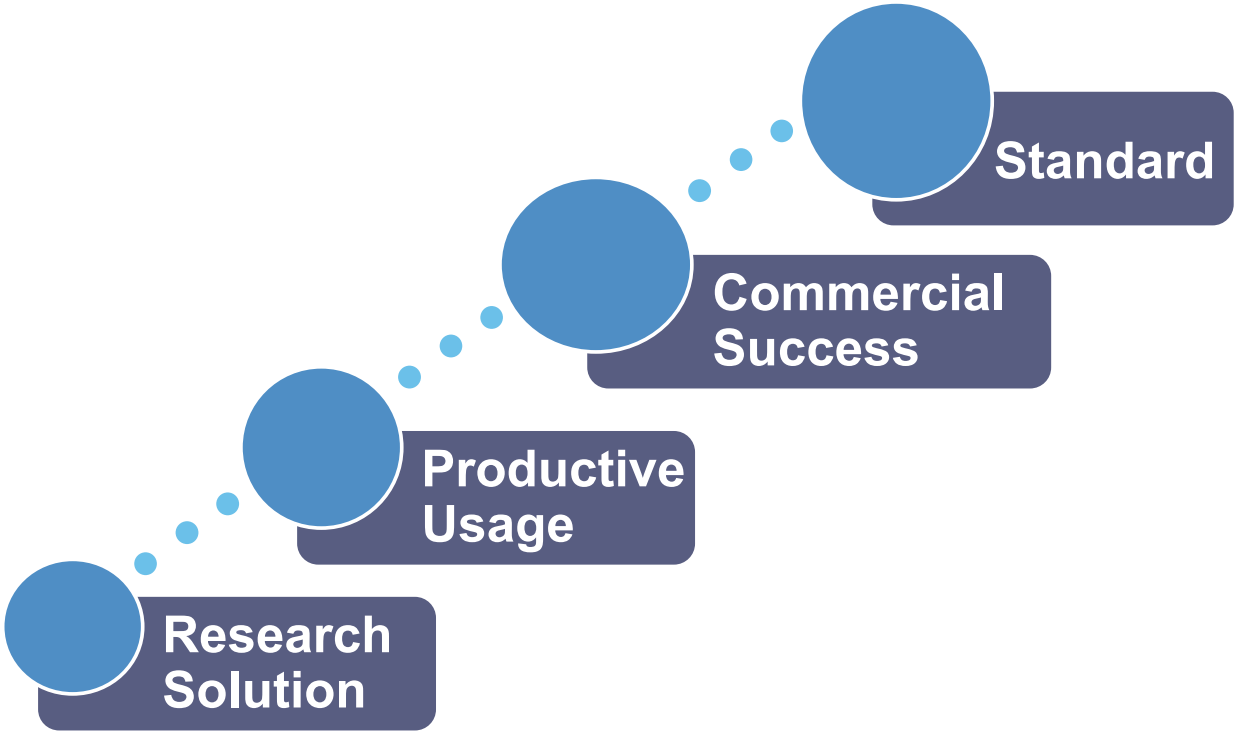
© Copyright by Dr. Gunter Strube - All rights reserved

Motivation

- Vision of Semiconductor Future ?

- Inspiration
  - ***Vision of a researcher***
  - ***Experience / Hindsight***
  - ***Visionary food for thought***

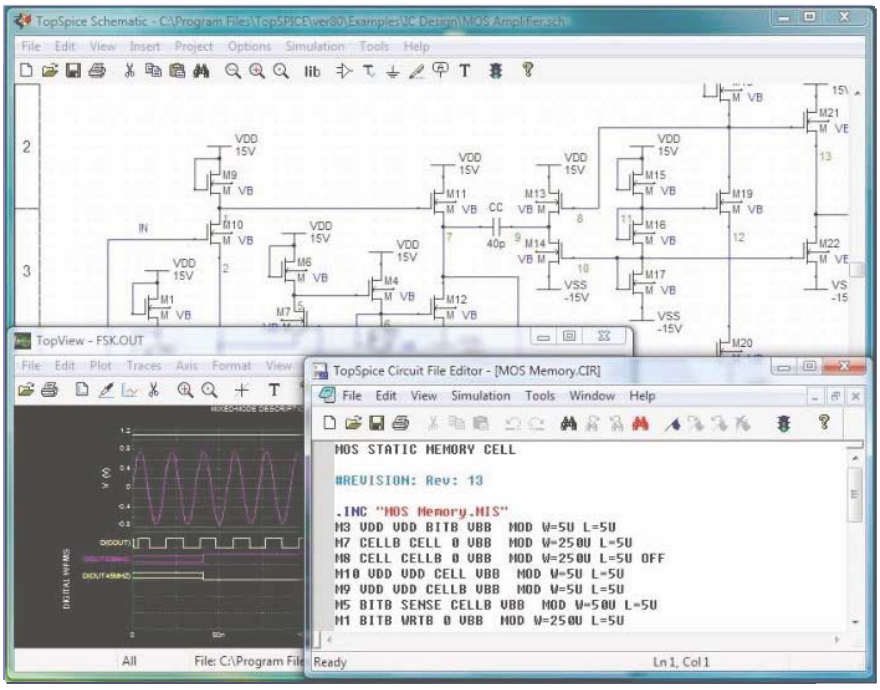
Main Stages



MunEDA Improve Design Performance & Yield

Example of a true standard

- Circuit Simulation … also known as SPICE



SPICE Development

Research Solution

Ron Rohrer

- Returned to U.C. Berkeley in 1968 to teach circuit synthesis
- No time to prepare / little faith in circuit synthesis
- Instead of teaching made students write a circuit simulator

Larry Nagel

- Took lead in his student group and developed CANCER (computer analysis of nonlinear circuits, excluding radiation)
- Improved CANCER in his master's and doctoral theses

- The result was SPICE,
  - first simulator with sparse matrix techniques
  - the first program to combine DC-, AC- , and transient analysis
  - one of the first public-domain tools (general-purpose circuit simulator)
  - first released by the University of California in 1971

MunEDA Improve Design Performance & Yield

SPICE Usage

Productive Usage

- Early stages of SPICE,
  - 1973: SPICE1 was first presented at a conference
  - 1975: The real popularity of SPICE started with SPICE2
  - Widely distributed and used as early open source program (SPICE was considered as first "valuable free software")

Commercial Success

- Commercialization of SPICE,
  - first commercial version of SPICE was ISPICE (National CSS, Inc.)
  - Kim Hailey, developer of H-SPICE co-founder of Meta-Software, (first version of SPICE considered commercial successful, 1978)

Standard

Almost Endless versions SPICE … free & commercial
i.e. Ken Kundert (Designer's Guide), developed Cadence Spectre simulator and derivations SpectreHDL and SpectreRF

Example of a true Standard

"If you ever designed a circuit you used SPICE … or you never really designed a circuit."

Michael Anderson

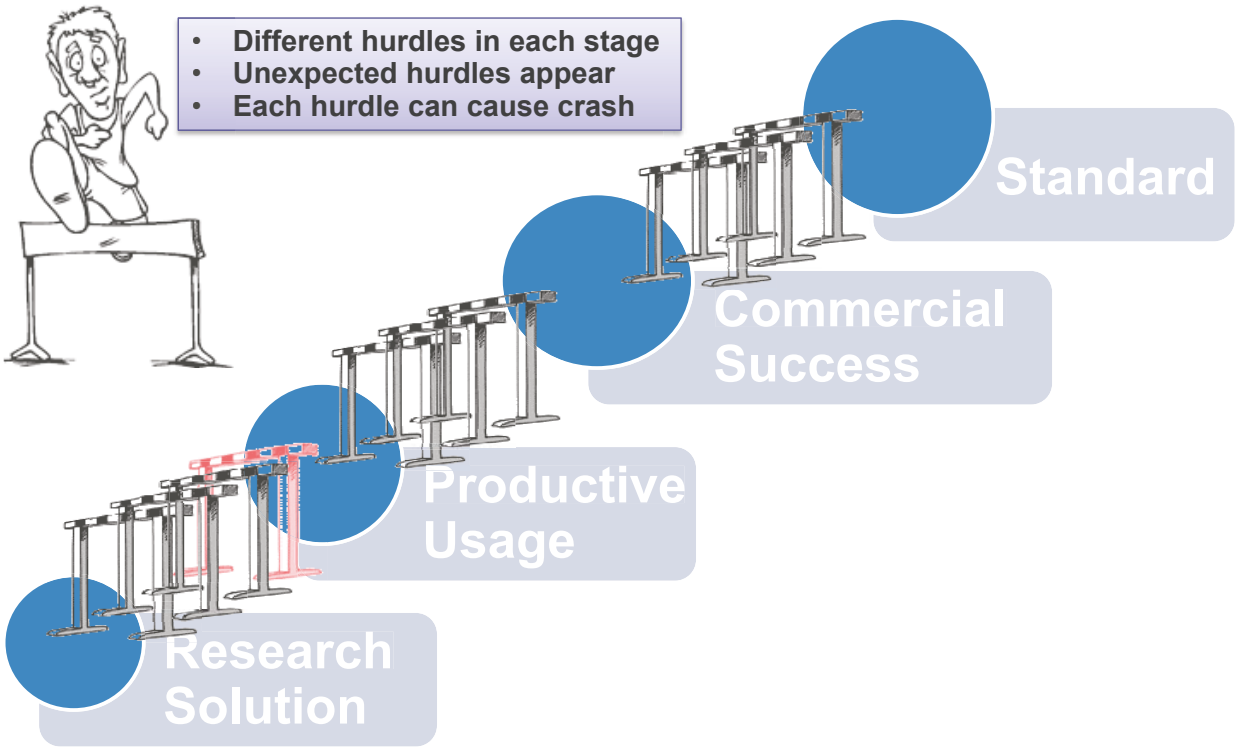
Sources:

- Michael Anderson http://www.geardiary.com/
- Richard Goering http://www.cadence.com/Community/
- Michael Demmler http://www.edn.com
- http://en.wikipedia.org/wiki/SPICE

MunEDA Improve Design Performance & Yield

Various Hurdles

- Different hurdles in each stage
- Unexpected hurdles appear
- Each hurdle can cause crash



European Example: WiCkeD

- … → 1992
  Early research at TU Munich leads to Worst-Case Analysis
- 1992 – 1995
  First Implementation of WiCkeD as tool inkl. GUI
- 1996/1997
  Research collaboration with designers (usage in R&D)

- Productive usage denied due to "lack of commercial support"

- Funded by BMBF as part of joint research project JESSI AC12

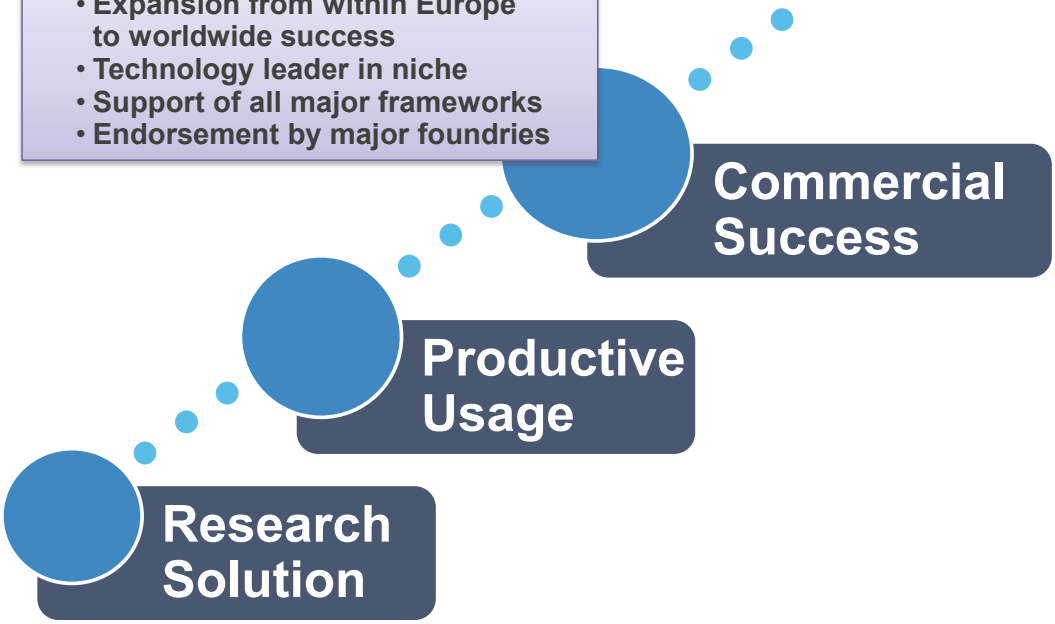
MunEDA Improve Design Performance & Yield

European Example: WiCkeD (2)

- 1996/1997
  initial discussions to provide "commercial support"
- 1997
  First business plan created as part of "Münchner Businessplan Wettbewerb" (MBPW)
- 1997-2001
  Expanded development efforts and usage in R&D
- 2001
  MunEDA started / first productive usage

European Example: WiCkeD (3)

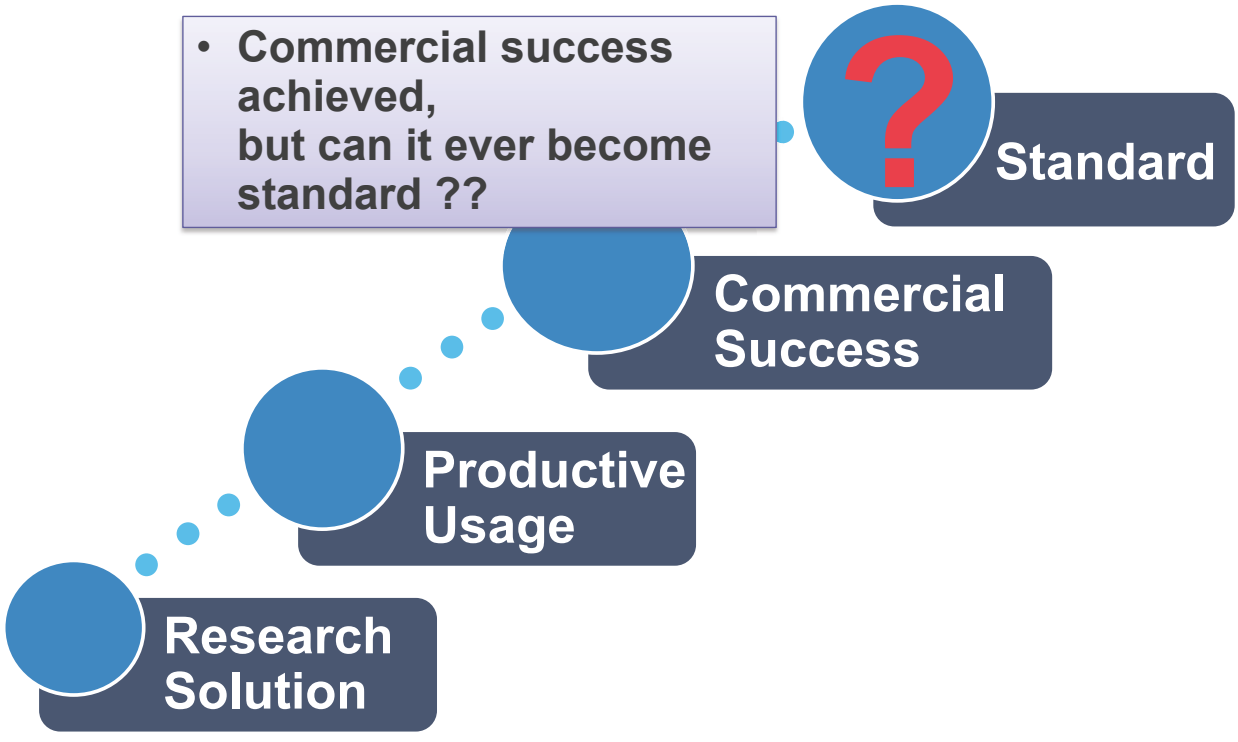
- 2001-2011
  - 10 years of commercial usage (survival of several industry crisis)
  - Expansion from within Europe to worldwide success
  - Technology leader in niche
  - Support of all major frameworks
  - Endorsement by major foundries



MunEDA Improve Design Performance & Yield

Vision of this Keynote

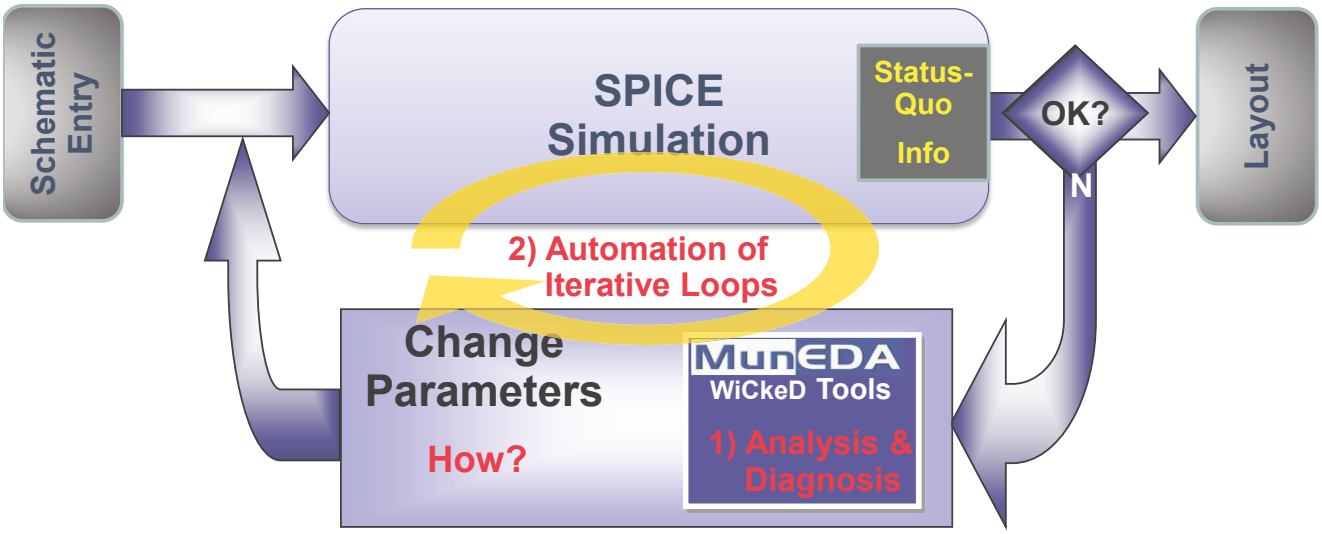
- Commercial success achieved, but can it ever become standard ??



MunEDA

Improve Design Performance & Yield

Quick Excursion: Fundamentals



1. Behavioural Analysis and Diagnosis calculate change (vector of parameters)
2. This enables automation of iterative loop
3. Same principle is applied including consideration of operating conditions and process statistics (significant increase in complexity)

WiCkeD™ Circuit Design

MunEDA

Improve Design Performance & Yield

Quick Excursion: Fundamentals (2)

- Behavioural Analysis
- automated iterative loop
- incl. operating ranges and process variances

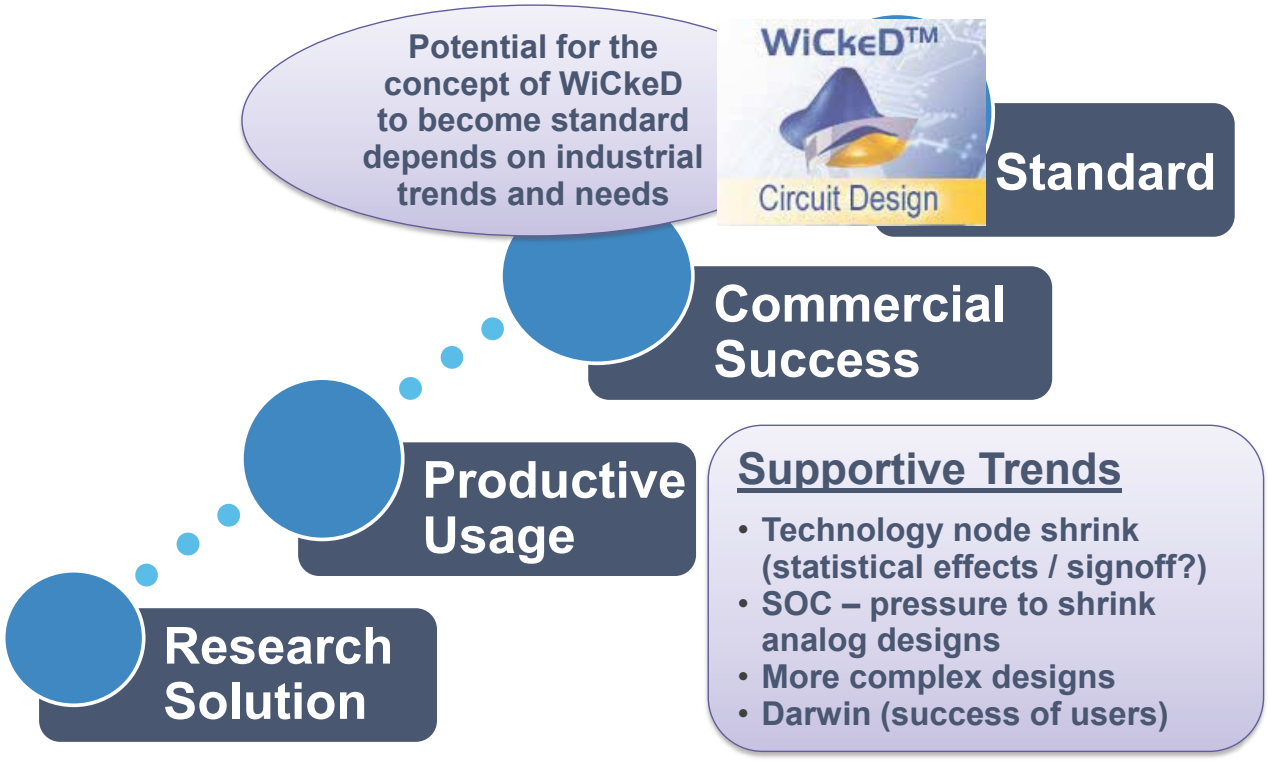
The concept of WiCkeD

The concept enables

- **Analysis** of circuit behaviour including 2nd order effects
- Exhaustive **design space exploration**
- High **design confidence**
- **Productivity** increase for fine tuning over operating ranges
- **Yield and circuit robustness** within feasible effort/time

MunEDA Improve Design Performance & Yield

Vision of this Keynote



MunEDA Improve Design Performance & Yield

Thanks !

- Food for thought, inspiration for
  - ***further WiCkeD adoption***
  - ***new solutions from Europe to evolve to commercial success***

- Talking about anniversaries …
  … MunEDA joins Silicon Saxony
  … and is looking forward to next 10 Years of

MunEDA and NEXT 10 YEARS OF SILICON SAXONY

Integration of Design Optimization into Automatic IP Generation with 1Stone® and WiCkeD™

Diese Arbeit ist Bestandteil des Forschungsprojektes SyEnA und wird vom Deutschen BMBF unter dem Zeichen 01 M 30 86 gefördert.

Harald Bothe[3], Uwe Eichler[1], Roland Jancke[1], Karl-Heinz Rooch[1], Matthias Sylvester[2], Reimund Wittmann[3]

Kurzfassung

Am Beispiel eines OTA-Entwurfs wird ein effizienter Entwurfsportierungsflow vorgestellt, der auf den Entwurfswerkzeugen 1Stone und WiCkeD basiert. Die besondere Kombination beider Werkzeuge ermöglicht eine symbolische Schaltungsabstraktion aus einer vorhandenen Prozessumgebung heraus und eine nachfolgende syntheseähnliche Abbildung incl. Dimensionierung in einen Zielprozess hinein. Die Portierung des Entwurfs umfasst Symbol, Schematic, Prüfschaltung und Layout einschließlich Spezifikationsanpassung und Dimensionierung. Damit gestattet dieser neue Design Flow schnellere Entwurfsschleifen, in denen Entwurfsoptimierung und Layoutgenerierung zusammen genutzt werden.

1 Einleitung

Die im digitalen Entwurf erreichten Fortschritte führen inzwischen dazu, dass der Analog/mixed-signal und RF-Entwurf nun auf dem kritischen Pfad beim Chip-Design-Flow liegt [1]. Insbesondere der Entwurf analoger Schaltungen schränkt hier die Entwurfsproduktivität ein.

Der Institutsteil Entwurfsautomatisierung (EAS in Dresden) des Fraunhofer-Instituts für Integrierte Schaltungen (IIS) sowie die IPGEN Microelektronics GmbH (Bochum) und MunEDA (München) konzentrieren ihre analogen Entwurfsaktivitäten im Förderprojekt SyEnA (Syntheseunterstützer Entwurf analoger Schaltungen) mit besonderer Ausrichtung auf Analog-IP-Nachnutzung.

Der Begriff „Nachnutzung von Analog-IP“ wird viel strapaziert; die Praxis in der Welt der Analogdesigner sieht aber vielfach noch anders aus. Die Schere zwischen Wunsch und Praxis geht bei der Nachnutzung analoger IP aus einer Quelltechnologie in einer Zieltechnologie noch weit auseinander.

In diesem Beitrag wird eine Entwurfsmethode vorgestellt, die bei einer Nachnutzung von analogen Entwürfen in einer anderen Technologie zu höherer Entwurfsproduktivität bei höherer oder zumindest gleichbleibender Entwurfsqualität führen kann.

Ein zwingendes Entwurfsziel sollte die Ausbeute sein [2]; der neue Entwurfsansatz berücksichtigt deshalb, dass bei einer Entwurfsportierung zwischen zwei Technologien ein zuvor nachgewiesenes hohes Ausbeuteniveau nur durch erneute Optimierung wieder erreicht werden kann.

Zur Validierung der neuen Entwurfsmethode wurde die Schaltung eines in einer 150-nm-CMOS-Technologie entworfenen OTA (Fraunhofer IIS Erlangen) in eine 350-nm-CMOS-Technologie portiert, entsprechend angepasst und optimiert.

Dieser Test hatte zum Ziel, Mängel und Schwachstellen im gesamten Portierungsprocedere aufzufinden – von der Generierung der Entwurfssichten mit 1Stone, über Machbarkeitsanalyse im ADE (Cadence) und

[1] Fraunhofer IIS EAS; vorname.name@eas.iis.fraunhofer.de

[2] MunEDA; matthias.sylvester@muneda.com

[3] IPGEN Microlectronics GmbH; vorname.name@ipgenme.de

Entwurfsoptimierung mit WiCkeD bis zur erneuten Generierung des Entwurfs einschließlich des neuen Layouts mit den nun optimierten Schaltungsparametern.

In der finalen Version wird sich die neue Entwurfsmethode darüber hinaus durch signifikante Alleinstellungsmerkmale wie die Kontrolle auf Einhaltung sogenannter Entwurfsrichtlinien und die Berücksichtigung zuverlässigkeitsrelevanter Betriebsbedingungen der einzelnen Bauelemente auszeichnen.

Bei der Konzeption dieser Entwurfsmethode wurde besonders die für den praktischen Einsatz analoger IP nötige Akzeptanz durch Analogdesigner beachtet.

2 Entwurfsabstraktion und Entwurfsverbesserung

Zunächst wird eine technologieunabhängige Beschreibung der Schaltung, ein sogenanntes Generic Engineering Model (GEM) definiert. Unter Zuhilfenahme eines formalen PDK werden aus diesem GEM die erforderlichen Entwurfssichten, z. B. Schematic und Layout für die jeweilige Zieltechnologie generiert. Dieser neugenerierte Entwurf kann dann mit Hilfe von WiCkeD optimiert werden. Dabei werden die geometrischen Parameter so eingestellt, dass für den Entwurf in der neuen Technologie eine entsprechend hohe Ausbeute gewährleistet ist.

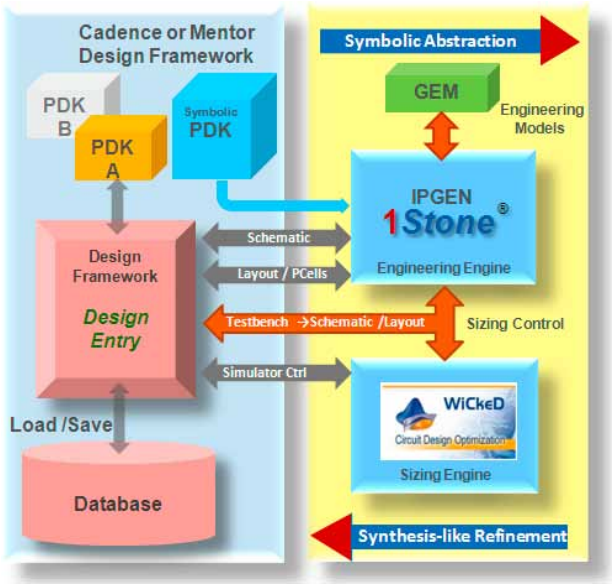


Bild 1 Vorgeschlagene Plattform für einen sicheren und portierbaren analogen Schaltungsentwurf

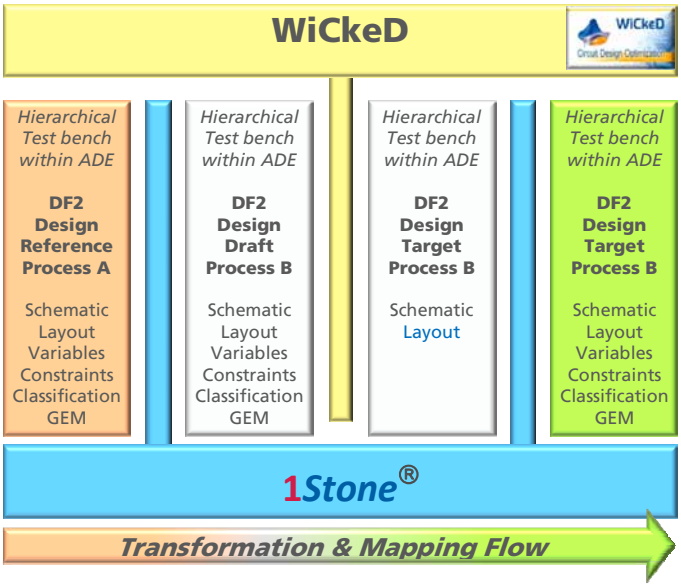


Bild 2 Vorgeschlagener Entwurfsportierungsflow

Auf der linken Seite zeigt Bild 1 eine Standard-Entwurfsumgebung, wie sie für einen auf Mentor oder Cadence basierenden analogen Schaltungsentwurf üblich ist. IPGEN Microelectronics GmbH schlägt eine Erweiterung (hier auf der rechten Seite in Bild 1) vor, um die Designer in komplexen Entwurfsaufgaben, wie Entwurfsportierung oder Spezifikationsänderung zu unterstützen. Hier besteht die Grundidee darin, die Aufmerksamkeit weniger auf das Ergebnis der IP-Entwicklung, als vielmehr auf den IP-Entwicklungsprozess selbst zu richten [3][4][5]. So können Entwurfsparameter, das PDK und selbst die Entwurfsumgebung bei der IP-Nachnutzung auf einfache Weise geändert werden.

Die professionelle IP-Entwicklungsumgebung 1Stone®-Developer (IPGEN Microelectronics GmbH) steht hier als zusätzliche Erweiterung zu den vorhandenen und auf Mentor oder Cadence aufsetzenden Entwurfsumgebungen zur Verfügung. Durch die Einbindung einer formalen Entwurfsabstraktion und der synthesenahen Entwurfsverbesserung zeichnet sich der Entwurf durch eine höhere Flexibilität aus, die auch das Schaltungslayout erfasst.

Die so erweiterte Entwurfsumgebung besitzt Merkmale, wie sie in zukünftig zu entwickelnden Entwurfsabläufen für die Automatisierung analoger Entwürfe unabdingbar sind. Eine ausführbare Prüfschaltung sichert die notwendige Konsistenz zwischen Verhaltensmodell, Schematic und Layout, welche durchaus mit Entwurfswerkzeugen unterschiedlicher Anbieter erzeugt werden können.

3 Entwurfsportierungsflow

Der hier vorgestellte Portierungsablauf beschreibt die für die Übertragung einer verifizierten Referenzschaltung in ein neues PDK und für die dann notwendige Anpassung der Schaltung erforderlichen Arbeitsschritte. Auf der Basis des GEM-Ansatzes werden alle Entwurfssichten einschließlich Schematic, Symbol, Layout und Prüfschaltung, generiert und verwaltet. Dieses Procedere stellt sicher, dass der Entwurf in allen Portierungsschritten konform ist.

Verfügt der Designer über die GEM-Beschreibung des Entwurfs, kann er individuelle Entwurfsänderungen in einer transparenten und überprüfbaren Weise umsetzen. Geänderte Spezifikation, aber auch aus der Layoutvisualisierung sich ergebende Schlussfolgerungen können umgehend berücksichtigt werden. Das trifft selbstverständlich auch auf die für die Maskenfertigung relevante Layoutfassung zu. Bei der Verwendung von Algorithmen zur Gestaltung von Layouttopologien und Aspektverhältnissen ist diese Funktionalität von besonderem Vorteil.

Die Übertragung eines Entwurfs in einen neuen Fertigungsprozess kann über folgende drei Portierungsschritte erfolgen (Bild 2):

Schritt 1: Die ursprünglich für Prozess A definierte GEM-Beschreibung wird mittels 1Stone in der PDK-Umgebung für Prozess B ausgeführt. Dabei hat der Designer alle Freiheiten in der Skalierung oder im Einfrieren einer Referenzgeometrie. Es werden alle Entwurfssichten für die neue PDK-Umgebung generiert.

Schritt 2: Bei der Entwurfsoptimierung mittels WiCkeD erfolgt nun die endgültige Definition der Geometrieparameter. Im Gegensatz zu separaten Lösungen kann WiCkeD hier bekannte Layoutauflagen (z. B., Symmetrie, Matching, Plausibilitätsbereich, Flächenverbrauch) berücksichtigen. Entsprechende Layoutversionen und die dazugehörigen Layoutauflagen stehen in allen Iterationsschritten zur Verfügung. Die Zahl der überwachten Entwurfsvariablen wird hierdurch reduziert, womit auch die Optimierung beschleunigt wird.

Schritt 3: Nachdem die geometrischen Parameter in der Schaltung aktualisiert sind, übergibt 1Stone die aktualisierten Bauelementeparameter an das GEM (Master-Design mit der gesamten Information zum Entwurfsprozess). Mit diesem optimierten GEM-Parametersatz für die Zieltechnologie B ist das GEM in der neuen Technologieumgebung flexibel einsetzbar.

Mit diesem 3. Schritt ist die Entwurfsportierung abgeschlossen.

Für den Fall, dass für den zu portierenden Entwurf keine GEM-Beschreibung vorhanden ist, Schaltung und Layout aber existieren, bietet 1Stone eine Reihe von Werkzeugen, mit deren Hilfe die vorhandenen Entwurfsinformationen zur Generierung eines vereinfachten GEM genutzt werden können.

4 Portierungsbeispiel

Um Mängel und Schwachstellen im gesamten Portierungsprocedere besser zu erkennen, musste ein Fremdentwurf aus einer fremden CMOS-Technologie in einer aktuell verwendeten Technologie zur Nachnutzung aufbereitet werden. Der zeitliche Rahmen für die Aufbereitung der Nachnutzung entsprechend dem oben vorgestellten Portierungsflows war auf etwa eine Woche begrenzt.

4.1 OTA als Beispiel-Schaltung

Sowohl 1Stone, als auch WiCkeD sind in die analoge Entwurfsumgebung (ADE) von Cadence integriert. Für die Portierung der Beispielschaltung aus der 150-nm-CMOS-Technologie in die 350-nm-CMOS-Technologie wurden die drei im Abschnitt 3 beschriebenen Schritte durchlaufen.

Schritt 1: Zunächst wurden mit dem GEM des OTA und der Entwurfsumgebung von 1Stone Schematic und Prüfschaltung des OTA in der 350-nm-Technologie generiert. Nach leichten Anpassungen – hier noch für die skalierten Geometrieparameter und die Prüfschaltung – wurde der OTA unter verschiedenen PVT-Werten simuliert.

Bild 3 zeigt die im ADE aufgerufene Entwurfsumgebung von 1Stone (Mitte und links) sowie Prüfschaltung und Schematic des generierten OTA (rechts).

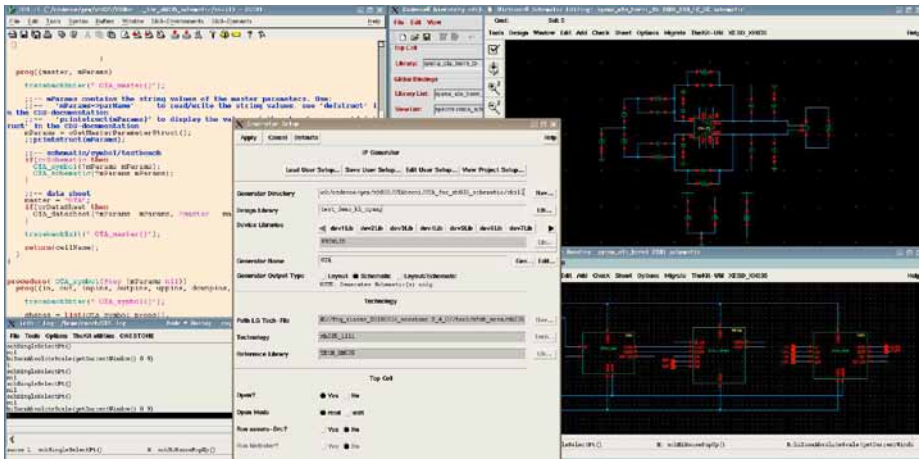


Bild 3 Generierung der Entwurfssichten in der 350-nm-CMOS-Technologie

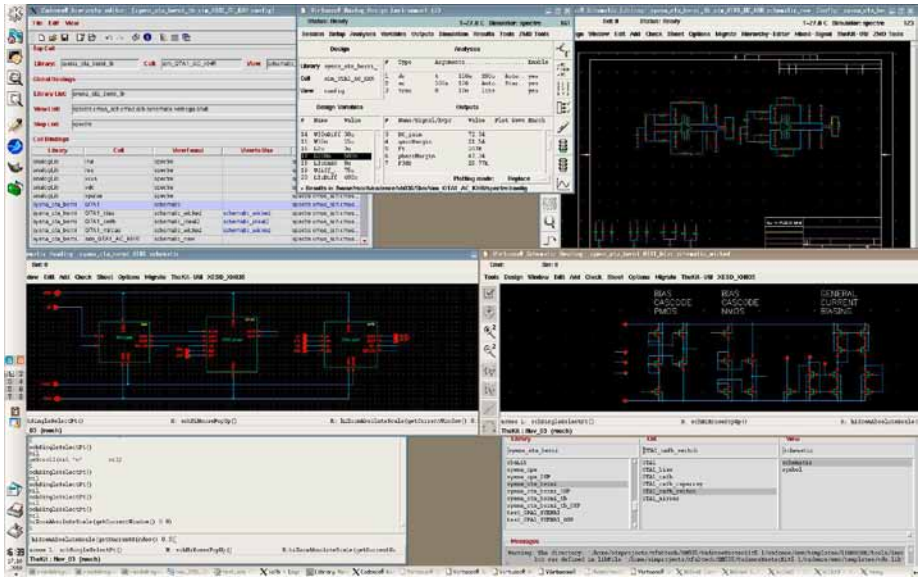


Bild 4 Erste Simulationen (PVT-Variationen) in der 350-nm-CMOS-Technologie

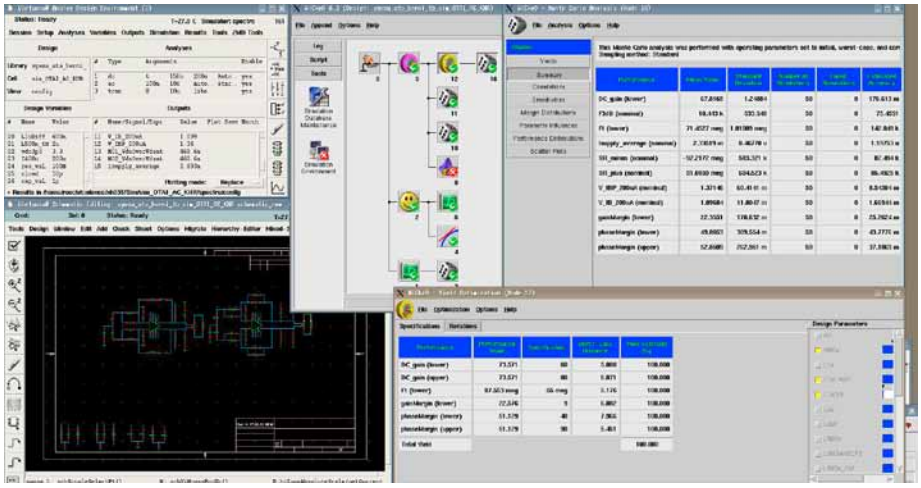


Bild 5 Ergebnis der Entwurfsoptimierung in der 350-nm-CMOS-Technologie

Die Machbarkeitsanalyse, also die ersten Simulationen (Bild 4) nach dem Setup der OTA-Analyse, sind noch weit entfernt von den angestrebten Zielparametern. Tabelle 1 (links) zeigt diese Differenz für zwei kritische Parameter.

Gerade für die Portierung analoger Schaltungen ist das auch nicht anders zu erwarten. Aus diesem Grunde sieht der vorgestellte Entwurfsablauf immer eine Entwurfsoptimierung vor.

Schritt 2: Der Verlauf der Optimierung ist aus Bild 5 ersichtlich. Im Detail wurden folgende Optimierungsschritte durchlaufen:

- Feasibility Analysis
- Monte Carlo Analysis
- Sensitivity Analysis
- Nominal Optimization
- Worst Case Analysis
- Monte Carlo Analysis
- Design Centering

Wie in Bild 5 zu sehen ist, verlief die Worst-Case-Analyse nicht fehlerfrei, d. h., die Schaltung funktioniert nicht im gesamten PVT-Bereich ohne Einschränkung; dieser Fakt zeigt sich auch in der erzielten Ausbeute von nur 92 % (siehe Tabelle 1). Sowohl das Ausbeuteergebnis (analoge Blöcke sollten eine Ausbeute nahe 100 % haben), als auch das suboptimale Frequenzverhalten sind der begrenzten Zeit und dem Charakter des Tests zur Feststellung noch offener Fragen und Probleme bei der Portierung analoger Schaltungen aus einer Quelltechnologie in eine Zieltechnologie geschuldet.

Schritt 3: Die im zweiten Schritt ermittelte Dimensionierung wurde nun an die aus dem ursprünglichen Prozessvorhandene GEM_Beschreibung übergeben und damit das Schaltbild und Layout im Zielprozess neu generiert.

Da WiCkeD in dieser Testanordnung noch nicht alle Layout-Auflagen bekannt gemacht wurden, waren noch kleine manuelle Optimierungsschritte durchzuführen. Diese betrafen aber lediglich die Einstellung der Faltungsparameter der Transistoren, die in der GEM-Beschreibung variabel beschrieben sind. Innerhalb weniger Minuten waren diese so eingestellt, dass das Layoutergebnis der bekannten Referenzschaltung entsprach.

	Initial-Simulation		**nach Optimierung**	
Parameter	**Ist**	**Ziel**	**Ist**	**Ziel**
DC-gain	>61dB	>65dB	>65dB	>65dB
Ft	>22MHz	>80MHz	>70MHz	>80MHz

Tabelle 1: Ergebnisse nach der ersten Anpassung / nach Optimierung mit 92 % Ausbeute

Der Einfluss der vom Designer vorgenommenen Parameteränderungen kann hierbei direkt im nachfolgend generierten Gesamtlayout abgelesen werden, so kann z. B. leicht die Flächenausnutzung optimiert werden.

Zukünftig werden auch die hier noch „manuell“ durchgeführten Optimierungsschritte mit in den Optimierungsprozess von WiCkeD integriert.

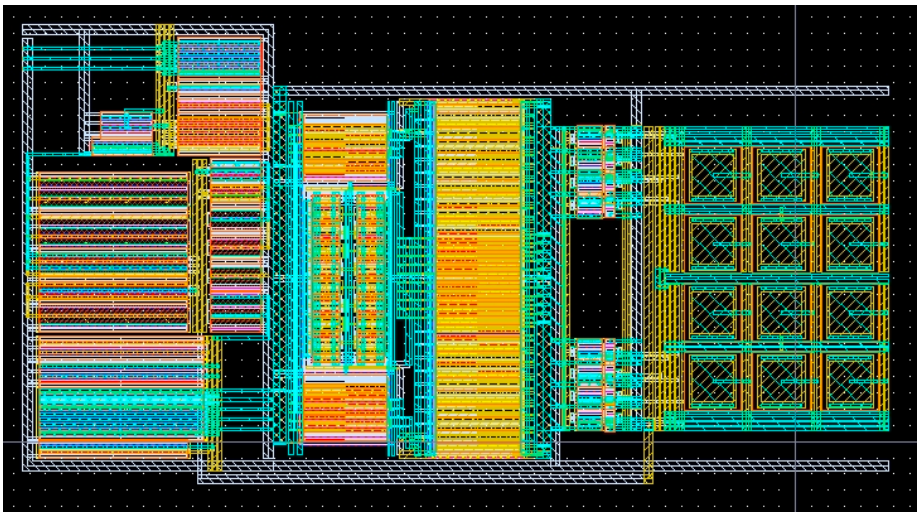


Bild 6 Neues OTA-Layout in der 350-nm-CMOS-Technologie

4.2 Abschätzung des Zeitaufwands

Ausgangspunkt war das GEM-Modell eines in einer 150-nm-CMOS-Technologie entworfenen OTA. Berücksichtigt man nur die tatsächliche Bearbeiterzeit (Rechenzeit und rechentechnischer Aufwand bleiben unberücksichtigt), so ergeben sich für die durchgeführten fünf Arbeitsschritte folgende Zeiten:

1. **1.5 h** → Den GEM-Schaltungsmaster in der 350-nm-PDK-Umgebung ausführen, um Schematic und Prüfschaltung zu generieren und Geometriedaten der Einzelelemente durch die zur Optimierung erforderliche Parameterliste zu ersetzen.
2. **2 h** → Simulationsstart im ADE (Cadence) zum Testen der Spezifikation und der Prüfschaltung.
3. **2 h** → Optimierung mit WiCkeD vorbereiten und ausführen sowie optimierte Geometrieparameter an 1Stone übergeben.
4. **2 h** → GEM-Master mit den optimierten Geometrieparametern ausführen und die geforderten Entwurfssichten einschließlich Prüfschaltung generieren.
5. **1.5 h** → Entwurf in der Zieltechnologie im ADE verifizieren.

Das Ergebnis dieser fünf Arbeitsschritte ist die geprüfte Entwurfsdatenbasis im ADE (Cadence), wobei die reine Bearbeitungszeit 9 h betrug. In der finalen Version mit existierendem Interface zwischen 1Stone und WiCkeD entfallen die Arbeitsschritte 1 und 4, wodurch sich die Bearbeitungszeit noch auf ca. 5.5 h reduziert. Diese 5.5 h stehen geschätzten 80 h Handentwurf gegenüber, wobei auch für den Handentwurf hier die Rechenzeit nicht berücksichtigt ist.

5 Zusammenfassung und Ausblick

Der Beitrag behandelt einen auf 1Stone (symbolische Abstraktion) und WiCkeD (syntheseähnliche Entwurfsverbesserung) basierenden Entwurfsportierungsflow. Die neue Methodik wurde am Beispiel eines zuvor in einer 150-nm-Technologie realisierten, nicht als Analog IP aufbereiteten und nun zur Nachnutzung in einer 350-nm-Technologie vorgesehenen OTA-Entwurfs demonstriert.

Durch den Einsatz von 1Stone®-Developer für die Generierung von Schematic, Layout und Prüfschaltung sowie von WiCkeD für die Entwurfsoptimierung sollte die neue Entwurfsmethodik für eine effiziente Analog-IP-Nachnutzung geeignet sein.

In naher Zukunft streben wir für Cadence, 1Stone und WiCkeD eine gemeinsame Datei für Entwurfsauflagen an, die für Schematic, Layout, Verhaltensmodell und Prüfschaltung frei von Redundanzen ist. Beide Funktionen sollten der Fehlerfreiheit dienen und den Entwurf beschleunigen.

Um die Akzeptanz bei den Designern zu erhöhen, werden wir zusätzlich Prüfroutinen für sogenannte Entwurfsrichtlinien in die IP-Entwicklungsplatt-form integrieren.

Darüber hinaus arbeiten wir an einer neuen Topologielösung, welche zur Vereinfachung der Entwurfsoptimierung und damit zu geringerem personellen Aufwand und höherer Akzeptanz führen sollte.

6 Literatur

[1] ITRS, International Technology Roadmap for Semiconductors,: ITRS 2010 update, http://www.itrs.net/Links/2010ITRS/Home2010.htm

[2] Israel Koren: Should Yield be a Design Objective? Proceedings of the 1st International Symposium on Quality of Electronic Design (ISQED '00, ISBN:0-7695-0525-2

[3] R. Wittmann, D. Rosendahl: Ausführbare Entwurfsablaufbeschreibungen für einen sicheren und effizienten Entwurfsablauf. Silicon Saxony Workshop „Entwurf von integrierten Analog- / Mixed-Signal- / HF-Schaltungen", 10. Mai 2007, Dresden

[4] R. Wittmann, R. Kakerow, H. Bothe, W. Schardein: A multi-purpose digital controlled potentiometer IP-core for nano-scale integration. IP 08 - IP Based Electronic System Conference, Proceedings, pp. 189-192, Grenoble, Dez. 2008

[5] R. Wittmann, R. Jancke, H. Bothe and B. Oelkrug: Rechnergestützter Entwurf wiederverwendbarer Analogschaltungen. GMM/ITG-Fachtagung Analog 2010 in Erfurt/Germany

Modellierung CNT-basierter thermischer Vias für den effektiven Wärmetransport

Jörg Hertwig, Matthias Thiele, Holger Neubert, Jens Lienig
{hertwig, thiele, neubert, lienig}@ifte.de
Technische Universität Dresden, Institut für Feinwerktechnik und Elektronik-Design

Kurzfassung

Thermische Vias sind wichtige Elemente zur Verlustleistungsabführung. Sie sind mit fortschreitender Miniaturisierung elektronischer Bauelemente und Strukturen ständig zu verbessern. Großes Potenzial, diese Herausforderung zu erfüllen, liegt in der Anwendung von hochwärmeleitfähigen Werkstoffen, besonders von Kohlenstoffnanoröhren (CNTs) und deren Kompositen. Die zielgerichtete Technologieentwicklung verlangt den Einsatz von mathematischen Modellen. Der Beitrag stellt ein thermisches Modell für CNT-basierte Komposite sowie mit ihm erzielte Simulationsergebnisse vor. Der Einfluss von Füllgrad, Ausrichtung und Struktureigenschaften der eingebetteten CNTs auf das thermische Verhalten des Komposits wird untersucht. Aus dem CNT-Komposit-Modell wird ein Mehrskalenmodell für thermische Vias und Viafelder entwickelt. Damit sind funktionell lohnende Ansatzpunkte für technologisch orientierte Forschungsarbeiten bestimmbar.

1 Einleitung

Als unvermeidliches Nebenprodukt fällt in elektronischen Bauelementen und Leitungen Wärme an, mit negativen Auswirkungen auf die Funktion und Zuverlässigkeit. Die freigesetzte Wärme muss von den Quellen weg über geeignete Wärmepfade in höhere Systemebenen und letztlich in die Umgebung transportiert werden. Ein besonderer Engpass der Wärmeabführung ist der senkrechte Durchgang durch Leitungsebenen in Schaltkreisen und Leiterplatten, insbesondere wegen der geringen Wärmeleitfähigkeit der verwendeten Substrate und der hohen Wärmestromdichten in unmittelbarer Nähe der Quellen. Um diesen kritischen Wärmedurchgang zu verbessern, verwendet man *thermische Vias*. Konventionelle thermische Vias auf Kupferbasis stoßen derzeit an die Grenzen ihrer Leistungsfähigkeit, besonders in Schaltkreisen mit großer Verlustleistung oder dreidimensionaler Bauelementeanordnung („3D-Integration“). Künftig werden deutlich kleinere Wärmewiderstände bei ungefähr gleichen oder kleineren Via-Abmessungen verlangt. Erfolg verspricht vor allem die Anwendung von Werkstoffen hoher Wärmeleitfähigkeit als Viamaterial.

Besonders Kompositwerkstoffe mit *Kohlenstoffnanoröhren* (engl. *Carbon Nano Tubes, CNTs*) sind aufgrund ihrer hohen Wärmeleitfähigkeit hierfür geeignet. Bisher wurden hauptsächlich CNT-Kunststoff-Komposite untersucht, die jedoch nur thermische Leitfähigkeiten bis zum 3,6-fachen des ungefüllten Matrixmaterials erreichen. Um die Leitfähigkeit weiter zu steigern, sind die CNTs zum einen an angrenzenden Festkörperoberflächen zu kontaktieren. Zum anderen sind sie im Komposit auszurichten, ohne dass es dabei zu gegenseitigen Berührungen und Verklumpungen kommt. Dies ist ein technologisch ungelöstes Problem.

Um der technologischen Forschung Ziele und Richtungen zu geben, sind Modelle zur Berechnung des thermischen Verhaltens von Kompositen wünschenswert. Dieser Beitrag stellt solche Modelle und mit ihnen erzielte Simulationsergebnisse vor.

2 Thermisches Modell von CNT-Kompositen

Das entwickelte thermische Modell basiert auf der Theorie des effektiven Mediums. Es berücksichtigt den Wärmetransport in den eingebetteten CNTs und der Matrix, den Einfluss von gegenseitigen Berührungen der CNTs, den Wärmeübergang zwischen CNTs und Matrix sowie stochastische Aspekte der Anordnung der CNTs im Komposit.

Der dazu in [1] entwickelte hierarchischer Modellierungsansatz mit den Modellebenen CNT, Komposit-Basiszelle und thermisches Via bzw. Viafeld mit jeweils angepassten Detaillierungsgrad ist in Bild 1 veranschaulicht. Ein solcher Ansatz minimiert die Anzahl der Freiheitsgrade im Modell und gestattet die Bewertung unterschiedlicher konstruktiv-technologischer Prinziplösungen. Ein weiterer Vorteil ist seine hohe Granularität.

Jedes einzelne CNT einer Basiszelle wird wegen der nahezu eindimensionalen Wärmeleitung als ein Linienelement mit einer spezifischen Wärmeleitfähigkeit modelliert. Dies ist zur Beschreibung stationärer Vorgänge ausreichend. Die spezifische Wärmeleitfähigkeit der CNT ist von Struktur und Abmessungen des CNT sowie der Temperatur abhängig. Die dreidimensionale Basiszelle des Komposits wird nach der Finite-Elemente-Methode (FEM) modelliert, wobei die eingebetteten CNTs als Ketten von Linienelementen implementiert sind. Position und Orientierung der Linienelemente berücksichtigen die zufällige Anordnung der CNTs mit Verteilungsfunktionen. Der Einfluss von Kontakten von CNTs untereinander wird über den Abstand benachbarter Linienelemente berücksichtigt.

3 Thermisches Modell von Vias und Viafeldern

Das Modell der Basiszelle errechnet die anisotrope Wärmeleitfähigkeit des Komposits, gemittelt über die Abmessungen der Basiszelle. Sie geht als Parameter in die Modelle von Vias und Viafeldern ein (Bild 1). Um Inhomogenitäten in den Eigenschaften, der Anordnung oder den Umgebungsbedingungen der CNTs in höheren Modellebenen zu berücksichtigen, sind entsprechend viele Basiszellen anzulegen.

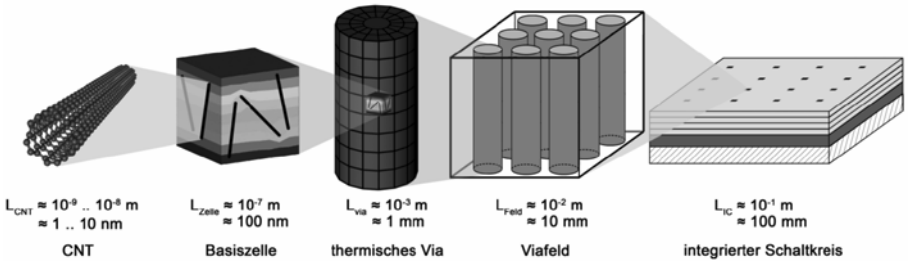


Bild 1. Mehrskalenmodell thermischer Viafelder mit CNT-Kompositen [1]

4 Simulationsergebnisse

Beispielhaft untersucht wurde die Basiszelle für ein Epoxydharz-Komposit mit (10,10)-CNTs[1] einer Länge von 80 nm. Die Ergebnisse beziehen sich auf die Leitfähigkeit der ungefüllten Matrix und geben die Werte in Richtung der Ausrichtungsachse der CNTs an. Entlang dieser Achse erreicht das Komposit die größte effektive Wärmeleitfähigkeit.

Zunächst wird die Ausrichtung von CNTs im Komposit untersucht. Die Ausrichtung jedes CNT im Kompositmodell weicht um einen Winkel ϕ von einer definierten Achse ab. Die Verteilungsdichte der Winkel aller im Komposit angeordneten CNTs quantifiziert den Grad der Ausrichtung der CNTs im Komposit.

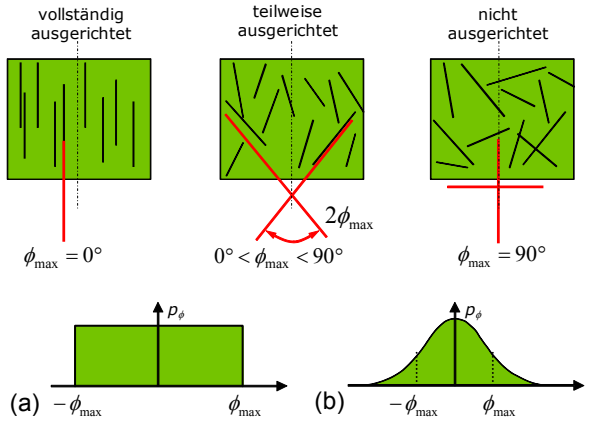


Bild 2 Ausrichtungswinkel ϕ bei (a) gleich- und (b) normalverteilter Anordnung der CNTs im Komposit

[1] Es gibt verschiedene Typen von CNTs, die abhängig von dem Winkel entstehen, unter dem eine monomolekulare Kohlenstoffschicht zu einer Nanoröhre aufgerollt ist. Diese Typen werden durch ihre Chiralität beschrieben und durch ein Indexpaar (n, m) gekennzeichnet: armchair (n, n), zig-zag (n, 0) und chiral (n, m) CNTs. [7].

Bei gleich verteilter Anordnung beschreibt ϕ_{max} die maximale Verdrehung der CNTs gegenüber der definierten Achse (Bild 2). $\phi_{max} = 0°$ bedeutet daher eine vollständige Ausrichtung und $\phi_{max} = 90°$ eine vollständige Unordnung. Bei normalverteilten CNTs gibt ϕ_{max} die Standardabweichung der Winkel ϕ aller CNTs im Komposit an.

Simulationsergebnisse für verschieden starke Ausrichtungen zeigt Bild 3 bei Annahme von gleichverteilten, Bild 4 von normalverteilten CNTs. Beide Verteilungen weisen auf ein ähnliches Verhalten hin. Deshalb werden im Weiteren nur gleichverteilte CNTs vorgestellt.

Komplett oder nahezu komplett zum Wärmestrom ausgerichtete CNTs erzielen erwartungsgemäß die größte thermische Leitfähigkeit des Komposits. Sie erreicht maximal das 11-fache der Wärmeleitung der Matrix. Je ungeordneter die CNTs, desto geringer wird sie. Bei einem Komposit mit ungeordneten CNTs erhöht sich die Wärmeleitfähigkeit nur noch maximal um den Faktor drei. Das Maximum der Leitfähigkeit wird abhängig vom Ausrichtungsgrad bei verschiedenen Füllgraden erreicht. Isotrope Komposite erreichen im Beispiel das Leitfähigkeitsmaximum bei einem Massenanteil von etwa 20 %, Komposite mit ausgerichteten CNTs bei etwa 40 %.

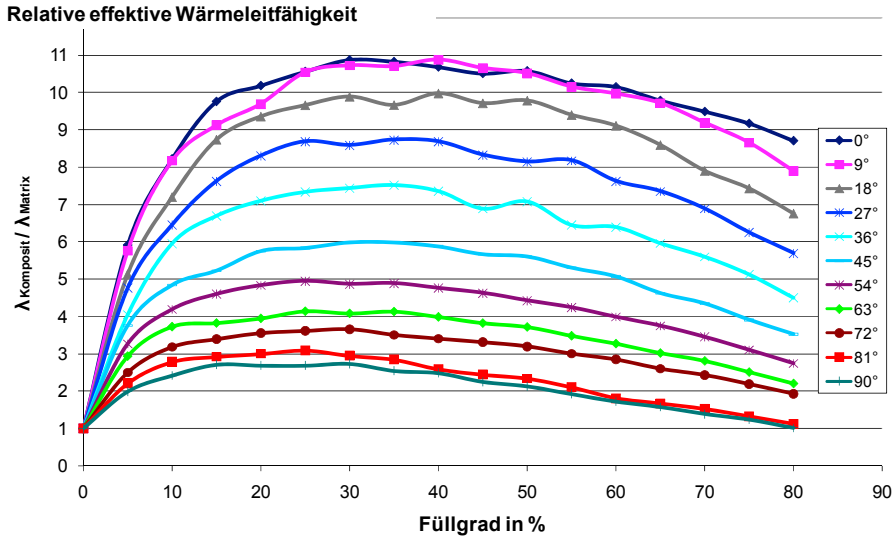


Bild 3 Einfluss des Ausrichtungswinkels ϕ auf die effektive Wärmeleitfähigkeit des Komposits

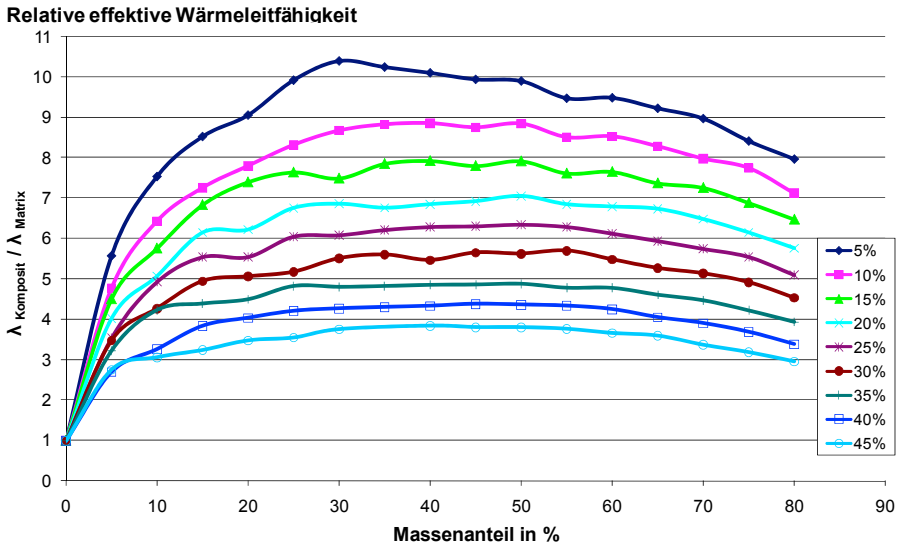


Bild 4 Einfluss der Standardabweichung auf die effektive Wärmeleitfähigkeit des Komposits bei normalverteilten CNTs

Bild 3 zeigt weiterhin, dass ab einem Maximum bei einem bestimmten Füllgrad die Leitfähigkeit des Komposits mit weiter zunehmendem Füllgrad wieder sinkt. Dies ist auf die mit dem Füllgrad zunehmende Anzahl von Kontakten zwischen CNTs in der Matrix zurückzuführen, welche die ballistische Wärmeleitung behindern (Bild 5). In Bild 6 ist zu erkennen, dass dieser Effekt bis zu einem Massenanteil von etwa 5 % kaum eine Rolle spielt. Mit weiter steigendem Füllgrad jedoch begrenzt er immer stärker den Zugewinn an Leitfähigkeit des Komposits, die sich ab etwa 15 % Füllgrad kaum noch steigern lässt.

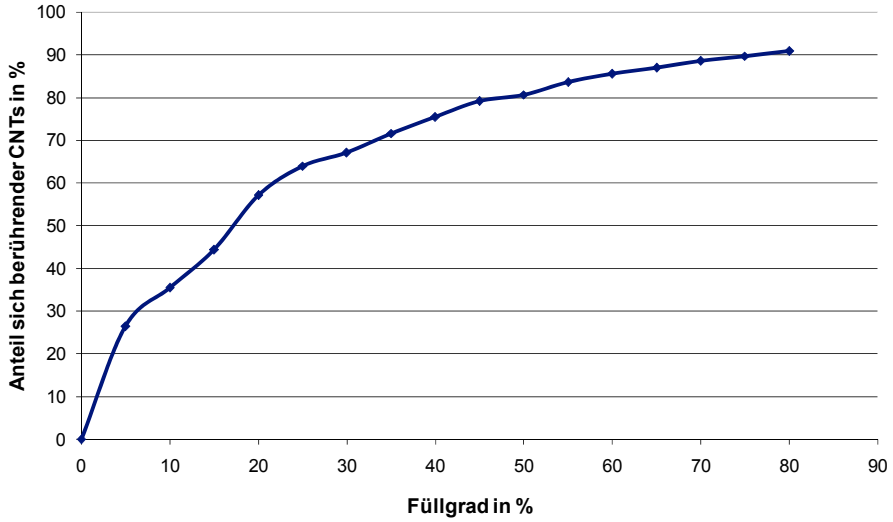


Bild 5 Anteil sich berührender CNTs

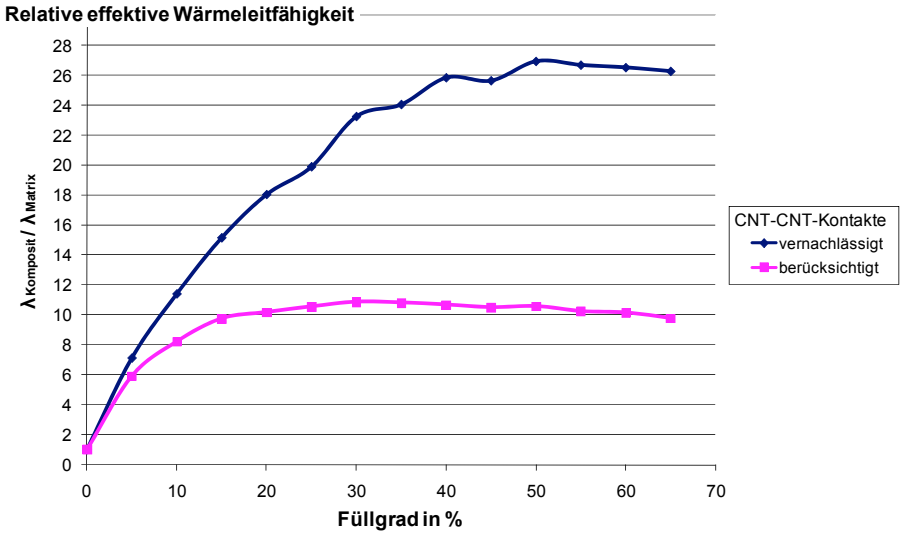


Bild 6 Einfluss der CNT-CNT-Kontakte, abhängig vom Füllgrad bei einem Ausrichtungswinkel $\phi = 0$

Weiterhin wurden der Einfluss von Länge und Durchmesser der CNT auf die Leitfähigkeit von Kompositen mit ausgerichteten CNTs untersucht. Prinzipiell besser wärmeleitend sind Komposite mit langen (Bild 7) und dünnen CNTs (Bild 8).

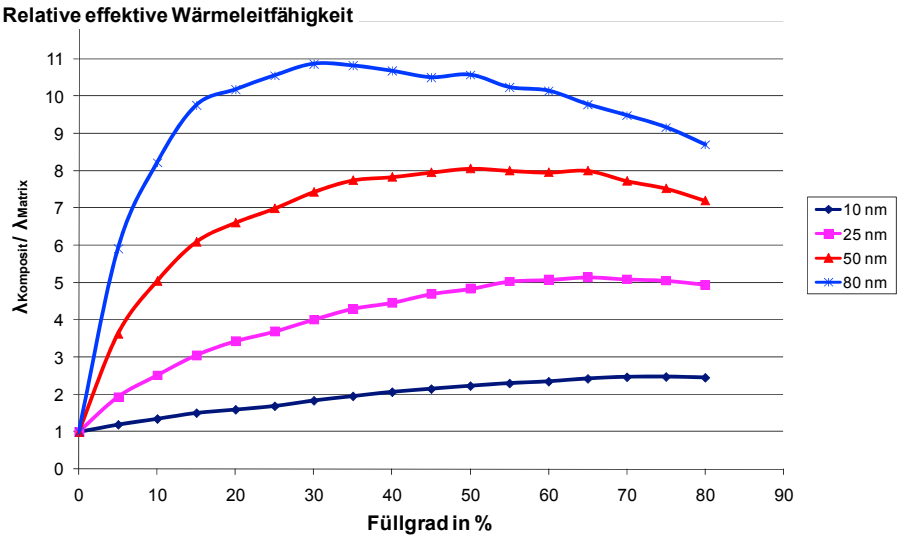


Bild 7 Abhängigkeit der thermischen Leitfähigkeit von der CNT-Länge

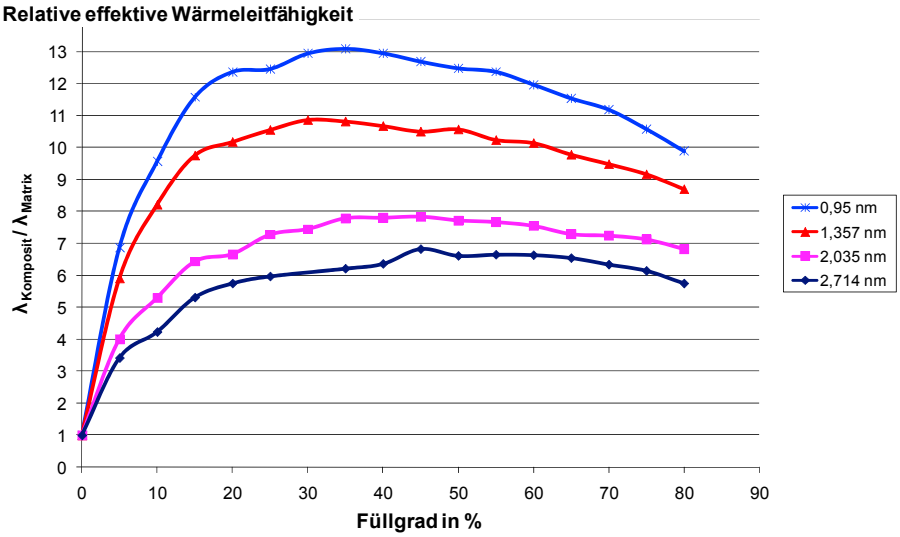


Bild 8 Abhängigkeit der thermischen Leitfähigkeit vom CNT-Durchmesser

Die zufällige Anordnung der CNTs in der Basiszelle beeinflusst die Ergebnisse. Die daraus resultierende Unsicherheit der Ergebnisse wurde mit einer Stichprobe von 36 unterschiedlichen Anordnungen von CNTs bei vollständiger Ausrichtung mit $\phi = 0°$ für verschiedene Füllgrade untersucht. Bei kleinen Füllgraden streuen die Ergebnisse maximal um 7 %. Für Füllgrade ab etwa 30 % nimmt die Streuung der Ergebnisse auf etwa 0,6 % ab. Diese Streuungen lassen sich durch größere Basiszellen, allerdings auf Kosten der Rechenzeit, verringern. Mit vom Füllgrad abhängigen Abmessungen der Basiszelle ist eine gleichmäßigere Ergebnisstreuung zu erreichen.

5 Diskussion der Ergebnisse

Die mit dem entwickelten Modell errechneten Ergebnisse sind im Hinblick auf die Mechanismen der Wärmeleitung in CNTs und Matrix plausibel. Es ist jedoch schwierig, sie mit experimentellen Befunden oder Simulationsergebnissen in der Literatur zu vergleichen. Bisher veröffentlichte Arbeiten gehen von sehr kleinen Füllgraden

von 1 bis 10 % aus ([7] Füllgrad bis 1,5 %; [2] bis 2 %). Dabei geht der Einfluss der Berührungen der CNTs in der oft erheblichen Messunsicherheit unter (23 % in [5]). Der Effekt wurde jedoch bei Anordnungen von CNTs, die nicht in eine Matrix eingebettet sind, beobachtet. Einzelne CNTs zeigen eine Wärmeleitfähigkeit von bis zu 3000 W/(m·K), Matten mit ausgerichteten CNTs von 250 W/(m·K) und Filze mit ungeordneten CNTs von 20 W/(m·K) [4].

Ebenfalls schwierig ist eine Bewertung der Ergebnisse zum Einfluss der Ausrichtung der CNTs in der Matrix. Die bisher veröffentlichten Modelle lassen keinen Vergleich zu. Experimentelle Untersuchungen zeigen jedoch, dass die Ausrichtung der CNTs einen messbaren Einfluss hat. In [3] konnte beispielsweise die Wärmeleitfähigkeit von Kompositen durch die teilweise Ausrichtung von CNTs gegenüber unausgerichteten etwa verdoppelt werden. Dieses Verhalten zeigt auch das vorgestellte Modell.

Die Leitfähigkeit des Komposits wird weiterhin maßgeblich von Länge und Durchmesser der CNTs bestimmt. Komposite mit langen CNTs leiten Wärme besser als solche mit kürzeren. Dies liegt daran, dass es bei vielen kurzen CNTs häufiger zu einem Wärmeübergang zwischen Matrixmaterial und CNT mit einem entsprechenden Übergangswiderstand kommt. Auch besitzen längere CNTs eine höhere thermische Leitfähigkeit als kurze.

Weiterhin errechnet das Modell für Komposite mit CNTs kleinerer Durchmesser eine höhere thermische Leitfähigkeit als für solche mit größeren (Bild 8). Die Ursache hierfür ist, dass bei gleichem Füllgrad in der Basiszelle eine größere Anzahl von CNTs angeordnet wird, wenn ihr Durchmesser kleiner ist.

Ein Vergleich mit anderen Modellen aus der Literatur ist insgesamt schwierig, da in ihnen verwendete Leitfähigkeiten der CNTs oft unrealistisch hoch (z. B. 3000 W/(m·K) in [6]) oder die experimentellen Untersuchungen, aus denen sie stammen, nicht näher beschrieben sind.

6 Zusammenfassung und Ausblick

Das entwickelte Mehrskalenmodell für CNT-basierte Kompositwerkstoffe gestattet detaillierte Parameterstudien hinsichtlich Typ, Abmessungen und Anordnung der eingebetteten CNTs sowie Einflussfaktoren der Umgebung. Damit lässt sich erstmals das Potenzial technologischer Ansätze, z. B. zur Ausrichtung und Kontaktierung der CNTs, vergleichen und bewerten. Der theoretisch optimale Volumenanteil von CNTs in einem Komposit kann abhängig von verschiedenen Einflussfaktoren bestimmt werden. Einflüsse wie CNT-Durchmesser, Volumenanteil, Ausrichtung oder lokale inhomogene Anordnung lassen sich so gemeinsam in einem dreidimensionalen Modell erfassen. Die hier vorgestellten Modelle zeigen das beste Potenzial für hochgefüllte Kompositen mit ausgerichteten, relativ langen und sich nicht berührenden CNTs. Perspektivisch lassen sich mit den Modellen die Eigenschaften verschiedener Anordnungen von thermischen Vias in Viafeldern simulieren und optimieren.

7 Literatur

[1] Hertwig, J.; Neubert, H.; Lienig, J: "Ein Ansatz zur Modellierung CNT-basierter thermischer Vias für den effektiven Wärmetransport in elektronischen Schaltkreisen," *Tagungsband Dresdner Arbeitstagung Schaltungs- und Systementwurf (DASS 2010)*, Fraunhofer Verlag, ISBN 978-3-8396-0126-6, S. 43-48, 2010

[2] Biercuka, M.J.; Llaguno, M.C.; Radosavljevicb, M.; Hyunc, J.K.; Johnson, A.T.: “Carbon nano-tube composites for thermal management”, *Applied Physics Letters*, 80:2767–1 – 12, 2002.

[3] Gonnet, P.; Liang, Z.: “Thermal conductivity of magnetically aligned car-bon nanotube buckypapers and nanocomposites” *ScienceDirect* 6, S. 119–122, 2006

[4] Eletskii, A. V.: “Transport properties of carbon nanotubes” *Physics - Uspekhi*, 52, S. 209 – 224, 2009.

[5] Haggenmueller, R.; Guthy, C.; Lukes, J.; Fischer, J.; Winey, K.: “Single wall carbon nanotube/polyethylene nanocomposites: Thermal and electrical conductivity” *Macromolecules*, 40, S. 2417–2421, 2007

[6] Singh, I. V.; Tanaka, M.: “Effect of interface on the thermal conductivity of carbon nanotube composites”, *ScienceDirect*, 46, S. 842–847, 2007

[7] Bara, A.; Bondar, A. M.; Svasta, P. M.: “Polymer/cnts composites for electronic packaging”, *Proc. 1st Electronics Systemintegration Technology Conference*, volume 1, S. 334–336, 2006.

Elektromigrationserscheinungen in zukünftigen digitalen Schaltungen

Matthias Thiele, Jens Lienig
{thiele, lienig}@ifte.de
Technische Universität Dresden, Institut für Feinwerktechnik und Elektronik-Design

Kurzfassung

Elektromigration stellt ein in zunehmendem Maße zu berücksichtigendes Problem in der Mikroelektronik dar. Durch die fortschreitende Strukturverkleinerung mit immer geringeren Leiterquerschnitten nimmt die Stromdichte stetig zu, was zu verstärkten Elektromigrationserscheinungen führen wird. Verschiedene Technologien müssen in zukünftigen digitalen Schaltungen eingesetzt werden, um trotz hoher Stromdichten die gewünschte Lebensdauer zu erreichen.

1 Einleitung

Die Zuverlässigkeit von integrierten Schaltungen ist eine zentrale Anforderung, die jeder Entwickler vor Augen hat. Diesem Zuverlässigkeitsanspruch wird durch vielfältige konstruktive Maßnahmen Rechnung getragen, z. B. durch die Wahl von Materialien, die den zu erwartenden Benutzungsansprüchen gerecht werden. Im Zuge der immer kleiner werdenden Strukturabmessungen kommen jedoch neue, die Zuverlässigkeit negativ beeinflussende Faktoren zum Tragen, die bisher vernachlässigbar waren. Insbesondere sind hier Elektromigrationsprozesse in elektrischen Leitungselementen zu nennen, die sich auch beim Schaltungs- und Layoutentwurf nicht mehr ignorieren lassen.

Bisher konnte die Elektromigration durch relativ einfache Maßnahmen, wie z. B. Leiterzugaufweitungen oder Oberflächenbehandlungen, unterdrückt werden. Auch waren derartige Maßnahmen meist auf Analogschaltungen mit einem hohen Gleichstromanteil begrenzt. Aufgrund eines gewissen Kompensationsprozesses in digitalen Schaltungen mit ihren alternierenden kapazitiven Auf- und Entladungen von Leiterbahnen konnte man hier die Elektromigration bisher weitestgehend ignorieren.

Jedoch ist abzusehen, dass die durch die ITRS-Roadmap prognostizierte weitere Strukturverkleinerung ein deutliches Ansteigen der zu erwartenden Stromdichte zur Folge hat [1]. Damit sind die bisher praktizierten Gegenmaßnahmen nicht mehr ausreichend. Auch ist mit Elektromigrationserscheinungen in digitalen Schaltungen zu rechnen.

In diesem Beitrag erfolgt zu Beginn eine Einführung in den Elektromigrationsprozess. Danach werden die Schaltungs- und Layoutparameter untersucht, die für das Auftreten bzw. die Vermeidung von Elektromigration von Bedeutung sind. Dabei steht insbesondere deren zukünftige Entwicklung im Vordergrund, erlaubt diese doch eine detaillierte Analyse der zu erwartenden Elektromigrationserscheinungen und ihrer Konsequenzen für den Schaltungs- und Layoutentwurf. Abschließend werden Maßnahmen angesprochen, mit deren Hilfe man nach heutigem Wissensstand dem Auftreten von Elektromigration auch in zukünftigen digitalen Schaltungen entgegentreten kann.

2 Was ist Elektromigration?

Elektromigration (EM) ist ein Materialmigrationsprozess in den metallischen Leiterbahnen integrierter Schaltkreise.

Die Kupfer- oder Aluminium-Leiterbahnen einer elektronischen Baugruppe sind polykristallin, d. h. sie bestehen aus Körnern, die Kristallgitter identischer Bauart, aber unterschiedlicher Orientierung, enthalten. Während des Stromflusses durch eine derartige Leiterbahn kommt es zu einer Wechselwirkung zwischen den bewegten Elektronen, die man sich als eine Art „Elektronenwind“ vorstellen kann, und den Metallionen in diesen Gitterstrukturen. Insbesondere die an den Korngrenzen anliegenden Atome werden verstärkt „Opfer“ des Elektronenwinds, d. h. es kommt zum Herauslösen und einer erzwungenen Wanderung dieser Atome in Richtung des Elektronenflusses. Damit werden im Laufe der Zeit Kupfer- bzw. Aluminium-Atome an einzelnen Korngrenzen in Stromflussrichtung angehäuft („Hillocks“). Gleichzeitig können in der entgegengesetzten Richtung Hohlräume an den Korngrenzen entstehen („Voids“). Während die Materialanhäufungen Kurzschlüsse zu den benachbarten Leiterzügen hervorrufen können, reduzieren die gleichzeitig entstehenden Hohlräume den Stromfluss stellenweise bis zur völligen Auftrennung des Leiterzuges (Bild 1).

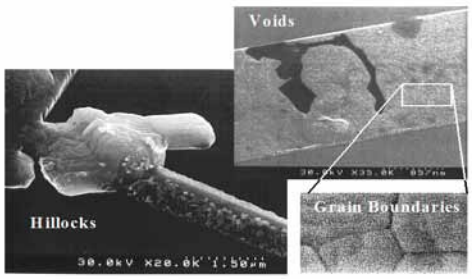


Bild 1: Hillock- und Void-Bildungen in Leiterbahnen aufgrund von Elektromigration (Foto: G. H. Bernstein und R. Frankovic, University of Notre Dame, USA) [2]

Zur Veranschaulichung der physikalischen Ursachen der Elektromigration sind die Kräfte zu betrachten, die auf Metallionen im Kristallgitter einer Leiterbahn wirken [2]. Der Stromfluss durch eine Leiterbahn erzeugt zwei Kräfte, denen die einzelnen Metallionen ausgesetzt sind. Zum einen wirkt eine elektrostatische Kraft auf die Metallionen ein, welche durch die elektrische Feldstärke im Leiterzug hervorgerufen wird. Aufgrund einer gewissen Abschirmung der positiven Metallionen durch die negativen Leitungselektronen lässt sich diese Kraft in den meisten Fällen vernachlässigen. Eine zweite Kraft entsteht nach heutigem Wissensstand durch Impulsübertragung von bewegten Leitungselektronen auf die Metallionen im Kristallgitter. Diese Kraft wirkt in Richtung des Stromflusses und ist die wesentliche Ursache des Elektromigrationsprozesses (Bild 2).

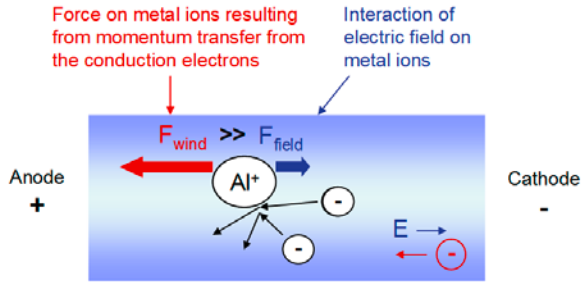


Bild 2: Metallionen, welche die Gitterstruktur einer Leiterbahn bilden, sind beim Stromfluss zwei Kräften ausgesetzt. Elektromigration resultiert aus der dominanten Kraft F_{wind}, die durch die Impulsübertragung von bewegten Leitungselektronen auf die Metallionen hervorgerufen wird. [2]

In einer homogenen kristallinen Struktur treten aufgrund der gleichmäßigen Gitteranordnungen der Metallionen kaum Impulsübertragungen zwischen den Leitungselektronen und den Metallionen auf. Diese Symmetrieverhältnisse sind jedoch an den Korngrenzen nicht mehr gegeben, womit verstärkt Bewegungsimpulse von den Leitungselektronen auf die Metallionen übertragen werden. Da die Metallionen an den Korngrenzen deutlich schwächer eingebunden sind als in einem regulären Kristallgitter, werden ab einer gewissen Stärke des Elektronenwinds Atome von den Korngrenzen abgetrennt und in Richtung des Stromflusses bewegt. Die Bewegungsrichtung wird außerdem noch von der Korngrenze selbst beeinflusst, da Atome bevorzugt entlang der Korngrenzen wandern. Wird die Stromrichtung über einen längeren Zeitraum konstant gehalten, entstehen die bereits erwähnten Hohlräume (Voids) und Materialanhäufungen (Hillocks).

Der Physiker J. R. Black entwickelte Ende der 60er Jahre ein empirisches Modell zur Abschätzung der mittleren Lebensdauer ($MTTF$, mean time to failure) einer Leiterbahn unter Berücksichtigung der Elektromigration [3]:

$$MTTF = \frac{A}{J^2} \cdot \exp\left(\frac{E_a}{k \cdot T}\right) \tag{1}$$

Die Konstante A ist dabei eine Materialkonstante, J die Stromdichte, E_a die Aktivierungsenergie für den Schädigungseffekt der Elektromigration, k die Boltzmannkonstante, und T die Temperatur im Leiterzug. Dabei wird offensichtlich, dass die im Rahmen des Schaltungs- und Layoutentwurfs beeinflussbaren Faktoren im Wesentlichen die Temperatur T und die Stromdichte J sind [2].

3 Elektromigrationsabhängige Schaltungs- und Layoutparameter

3.1 Strukturgröße

Wie aus Gleichung (1) ersichtlich, ist die auftretende Stromdichte J ein wesentlicher Parameter, mit dem sich die mittlere Lebensdauer einer Leiterbahn beeinflussen lässt (s. Abschnitt 3.2). Da sich die Stromdichte

aus dem Betrag des Quotienten von Strom I zu Querschnittsfläche A ermittelt, und bei den meisten Prozesstechnologien von einer festen Leiterbahnhöhe ausgegangen wird, ergibt sich eine direkte Beeinflussbarkeit der Stromdichte aus der Breite der Leiterbahn: Je breiter diese ist, um so geringer ist die auftretende Stromdichte, und um so größer ist die Beständigkeit gegen Elektromigration.

Die sogenannte Bambusstruktur der Korngrenzen erhöht die Elektromigrationsbeständigkeit einer Leiterbahn. Neben der bewussten Wahl einer Leiterbahnbreite, bei der eine derartige Bambusstruktur existiert, werden bei integrierten Schaltungen oft durch einen abgestimmten Abkühlungsprozess große Korndurchmesser und damit die Unterstützung einer Bambusstruktur angestrebt. Ebenfalls genutzt werden „geschlitzte“ Leiterbahnstrukturen, bei denen die geringen Breiten der einzelnen Metallstrukturen zwischen den Schlitzen die Ausbildung einer Bambusstruktur ermöglichen [2].

3.2 Stromdichte

Der Strom eines Segmentes einer Leiterbahn wird von der elektrischen Schaltung vorgegeben. Durch die Geometrie der Leiterbahn ergibt sich daraus die Stromdichte, wobei $J = I/A$ (mit Stromdichte J, Strom I und Querschnittsfläche A). Dabei ist nicht allein die Querschnittsfläche der Leiterbahn ausschlaggebend, sondern auch deren Verlauf mit Knickwinkeln und Richtungsänderungen sowie Vias. An Knicken erhöht sich die lokale Stromdichte, da sich die Stromlinien an der Innenseite sammeln (*current crowding*). An Orten mit erhöhter Stromdichte entstehen bevorzugt Elektromigrationsschäden.

Die ITRS-Roadmap [1] gibt als maximale Stromdichte bei Cu $2 \cdot 10^6$ A/cm² bei 105 °C an. Industrienahe Simulationen nutzen oftmals geringere Werte (z. B. $1 \cdot 10^4$ A/cm² bei Kupfer), um nicht erfasste Randbedingungen (z. B. Risse oder Inhomogenitäten im Material) in Betracht zu ziehen. Unabhängig davon gilt, dass der maximale Stromdichtewert von Kupfer etwa fünf Mal höher ist als der von Aluminium.

Auch ist auf die starke Temperaturabhängigkeit von maximalen Stromdichtewerten zu achten. In der Black'schen Gleichung (1) erscheint die Temperatur der Leiterbahn im Exponenten, d. h. sie hat einen sehr starken Einfluss auf deren mittlere Lebensdauer. Konkret verringert eine Temperaturerhöhung von 100 K in einer Al-Metallisierung die zulässige Stromdichte um ca. 90 % [4].

3.3 Frequenzabhängige Strombetrachtung

Stromart und Elektromigrationsbeständigkeit sind stark voneinander abhängig. Untersuchungen [5] zeigen eine erhöhte Elektromigrationsbeständigkeit von Leitungen mit bidirektionalen und gepulsten Strömen, verglichen mit Gleichströmen und konstanten Strömen. Ein Grund für diese Abhängigkeit ist der sogenannte „Selbstheilungs“-Effekt (durch Material-Rücktransport aufgrund wechselnder Stromrichtungen), der die effektive Materialmigration reduziert.

Bei der Berechnung von EM-relevanten Strömen nutzt man demzufolge verschiedene Modelle abhängig von der Frequenz des Stromflusses: (1) das Effektivwert-Modell, basierend auf dem RMS (*root-mean-square*)-Wert des Stroms für Frequenzen kleiner 1 Hz, (2) das Mittelwert-Modell für Frequenzen größer 1 Hz und (3) das Spitzenwert-Modell zur Berücksichtigung elektrostatischer Entladungen (*ESD – electrostatic discharge*).

Das Effektivwert-Modell berücksichtigt keinen Selbstheilungseffekt. Dieses Modell stellt einen konservativen Ansatz dar und ist geeignet für alle analogen Gleichstromnetze und zuverlässigkeitskritische Anwendungen im Allgemeinen.

Das Mittelwert-Modell berücksichtigt den Selbstheilungseffekt wechselnder Stromrichtungen. Es wird gewöhnlich in digitalen Netzen angewendet.

Der Spitzenwert – zur Berücksichtigung kurzzeitiger Ströme wie bei einem ESD-Ereignis – muss unabhängig von den anderen Modellen untersucht werden, da die vorausschauende Berücksichtigung verschiedener Schadensmechanismen unterschiedliche Auswirkungen auf den Layoutentwurf hat [4].

4 Wie entwickelt sich die Mikroelektronik?

Die Technologie-Roadmap *ITRS* [1] beschreibt die Trends in der zukünftigen Halbleiterfertigung. Darin wird in Fortschreibung des Moore'schen Gesetzes die weitere Verkleinerung der Strukturabmessungen prognostiziert. Die wichtigsten verdrahtungsrelevanten Parameter für die Technologien der nächsten Jahre sind in Tabelle 1 zusammengefasst.

Jahr		2010	2013	2016	2019	2022
Technologie		45 nm	32 nm	22 nm	16 nm	11 nm
Zahl der Metallebenen		12	13	13	14	15
Gesamte Verdrahtungslänge[1]	m/cm^2	2222	3737	5285	7475	10571
Leiterbahnbreite	nm	45	32	22	16	11
Leiterbahn-Aspektverhältnis		1,8	1,9	2,0	2,0	2,1
Via-Aspektverhältnis		1,6	1,7	1,8	1,8	1,9
Leiterbahnhöhe	nm	81	61	45	32	23
Leiterbahnquerschnitt	nm^2	3600	2000	990	510	250
Maximale Frequenz	MHz	5875	7344	9180	11475	14343
Strom	µA	52	48	31	23	15
Maximale Stromdichte	MA/cm^2	1,44	2,40	3,13	4,53	5,99
Leiterbahnwiderstand[2]	$\mu\Omega \cdot cm$	4,08	5,63	7,00	8,88	11,71
Betriebsspannung	V	0,97	0,87	0,78	0,71	0,64
Leistungsindex[3]	$W/GHz \cdot cm^2$	1,3–1,4	1,6–1,8	1,6–1,9	1,8–2,0	1,9–2,1

Tabelle 1: Technologieparameter der aktuellen und zukünftigen Halbleitertechnologien mit maximalen Frequenzen, Strömen und Stromdichten. Alle leiterbezogenen Angaben für Kupfer bei 105 °C, nach ITRS [1].

4.1 Strukturgröße

Zur Erzielung einer zunehmenden Integrationsdichte, welche mit erhöhter Zuverlässigkeit der einzelnen Elemente einhergeht, ist die Größe der Halbleiterstrukturen stetig zu reduzieren (Tabelle 1). In der Halbleiterindustrie strebt man eine Halbierung der Strukturgröße alle sechs Jahre an. Gleichzeitig wird ein geringerer Flächenbedarf der integrierten Schaltkreise erreicht, was höhere Stückzahlen pro Wafer ermöglicht. Darüber hinaus sind mit kleineren Strukturgrößen energieeffiziente oder schnelle Schaltungen realisierbar. Als Nebeneffekt werden aber trotz wachsenden Aspektverhältnisses die Querschnittsflächen der metallischen Leitungen reduziert, von 3600 nm^2 im Jahr 2010 auf prognostizierte 990 nm^2 im Jahr 2016.

4.2 Strom, Stromdichte und Temperatur

Durch geringere Strukturgrößen und Betriebsspannungen lassen sich die benötigten Ströme reduzieren, da kleinere Transistoren auch kleinere Gate-Kapazitäten besitzen. Aufgrund der steigenden Frequenzen werden die Ströme jedoch nicht im gleichen Maße wie die Leiterbahnquerschnitte reduziert. Aus der Beziehung $J = I/A$ folgt, dass dadurch die auftretende Stromdichte steigt.

Der spezifische Widerstand des Leitermaterials steigt aufgrund von Streueffekten bei geringen Leiterbahnquerschnitten. In Kombination mit der steigenden Stromdichte ist mit höheren lokalen Temperaturen aufgrund der Eigenerwärmung (*joule heating*) zu rechnen. Die erlaubten Umgebungstemperaturen für integrierte Schaltkreise bleiben aufgrund der limitierten Sperrschichttemperatur von Silizium annähernd konstant im Bereich zwischen Raumtemperatur und 125 °C. Durch hohe Verlustleistungen und die Eigenerwärmung kann die Temperatur im integrierten Schaltkreis stark inhomogen verteilt sein.

4.3 Frequenz

Aus den kleineren Strukturabmessungen folgen geringere Leitungskapazitäten und -induktivitäten. Deshalb können höhere Signalfrequenzen erzielt werden, welche für höhere Verarbeitungsgeschwindigkeiten und dadurch für eine höhere Leistungsfähigkeit der Schaltung sorgen. Als Nebeneffekt erzeugen hohe Signalfrequenzen allerdings, durch häufiges Umladen von Gate-Kapazitäten, hohe Ströme in den Versorgungsnetzen. Die zu erwartenden maximalen Frequenzen sind in Tabelle 1 aufgeführt. Durch neue Transistor-Technologien (Tabelle 2) wird der Trend zu höheren Frequenzen fortgesetzt. Allerdings reduziert sich bei einigen dieser Technologien die benötigte Schaltenergie, was eine Stromsenkung in den Versorgungsnetzen ermöglicht.

[1] Verdrahtungslänge pro Chipfläche
[2] Spezifischer Widerstand des Leitungsmaterials
[3] Leistung pro GHz und cm^2 Chipfläche

Devices		Si CMOS	CNT FET	Graphene Nanoribbon FET	Nanowire	III-V FETs	Ge FETs	FinFET, Tri-gate FET, GAA FET, Local SOI
Cell Size (spatial pitch)	P	100 nm	100 nm	100 nm	30 nm	15 nm	15 nm	100 nm
	D	590 nm	≈ 1,5 µm	1,5 µm	≈ 1 µm	40 nm	26 nm	300 nm
Density, devices/cm^2	P	1,0E+10	4,5E+9	4,5E+9	1,0E+11	1,0E+11	1,0E+11	1,0E+10
	D	2,8E+8	4,0E+7	4,0E+7	1,0E+8	1,5E+10	3,0E+10	4,7E+9
Switch Speed	P	12 THz	6,3 THz	n. k.	6,5 THz	n. k.	n. k.	12 THz
	D	1,5 THz	4 GHz	26 GHz	250 GHz	2 THz	290 GHz	> 200 GHz
Circuit Speed	P	61 GHz	61 GHz	61 GHz	100 GHz	n. k.	n. k.	61 GHz
	D	5,6 GHz	220 Hz	22 kHz	11,7 MHz	n. k.	n. k.	8 GHz
Switching Energy, J	P	3,0E-18	3,0E-18	3,0E-18	4,0E-20	n. k.	n. k.	3,0E-18
	D	1,0E-16	1,0E-11	n. k.	6,0E-16	n. k.	4,0E-15	n. k.

Tabelle 2: Parameter von Transistoren zukünftiger Technologien, D = nachgewiesen (*demonstrated*), P = hochgerechnet (*projected*), n. k. = nicht bekannt (*not known*), aus ITRS-Roadmap [1]

5 Künftige Probleme durch Elektromigration

Wie in den vorangegangenen Abschnitten erläutert, nimmt die Stromdichte mit fortschreitender Strukturverkleinerung in den metallischen Leiterbahnen zu. Ursache hierfür ist die stärkere Abnahme der Querschnittsfläche im Vergleich zu den Stromwerten. Parallel dazu sinken die effektiven Stromdichtegrenzwerte für die Elektromigration mit abnehmenden Strukturgrößen (Bild 4), da bei kleinen Strukturgrößen bereits Schäden kleiner räumlicher Ausdehnung zum Ausfall führen können. Diese Entwicklungen sorgen dafür, dass die für die Integration unabdingbare hohe Zuverlässigkeit der einzelnen Elemente durch Elektromigration gefährdet wird.

Bild 3 veranschaulicht die nach der ITRS-Roadmap prognostizierten sinkenden Ströme in digitalen Netzen. Gleichzeitig nehmen die durch Elektromigration bestimmten Grenzwerte für minimal dimensionierte Leiterbahnen jedoch so stark ab, dass bereits im Jahr 2011 die untere Grenze (gelb) überschritten wird. Es sind daher Maßnahmen zum Schutz vor Elektromigration zu treffen. Ab dem Jahr 2015 reichen nach dieser Prognose die aktuell bekannten Maßnahmen nicht mehr aus, um bei den prognostizierten Strömen Leiterbahnen mit minimaler Strukturabmessung zu verwenden.

Die steigenden Stromdichten nach der ITRS-Prognose sind in Bild 4 dargestellt. In der logarithmischen Darstellung ist besonders deutlich die Diskrepanz zwischen steigenden Stromdichten (durch die Schaltungsanforderungen) und den sinkenden Grenzwerten (durch Elektromigration) zu erkennen.

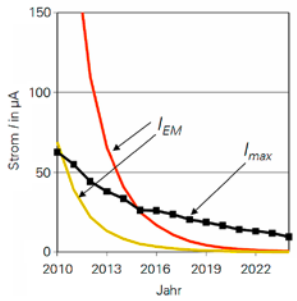


Bild 3: Maximal auftretende Ströme in digitalen Schaltungen (schwarz) und zulässige Ströme für minimal dimensionierte Leiterzüge (gelb und rot) in der jeweils aktuellen Technologie bei Kupfer, Metallisierungsebene 1 bei 105 °C, nach [1].
gelb: Elektromigration ist zu berücksichtigen, rot: schnelle Zerstörung durch EM

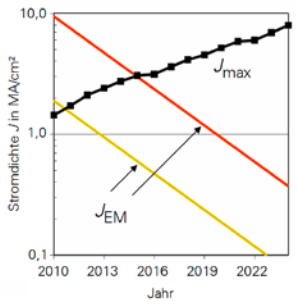


Bild 4: Maximal auftretende Stromdichten in digitalen Schaltungen (schwarz) und zulässige Stromdichten für minimal dimensionierte Leiterzüge (gelb und rot, siehe Bild 3), nach [1]

Deshalb kommt es ab einer Strukturgröße von unter 40 nm zu einer Überschreitung der Grenzwerte, wenn bei minimalen Abmessungen die nach ITRS prognostizierten Ströme auftreten. Damit sind Ströme in minimal dimensionierten Leitungen erstmals einer Elektromigrationsbegrenzung zu unterziehen. Diese Grenze wird bereits mit den aktuellen Technologien überschritten, weshalb in Fällen, in denen hohe Ströme und minimale Abmessungen zusammentreffen, bereits heute mit Elektromigration zu rechnen ist.

Die Entwicklung hin zu hohen Frequenzen wirkt sich positiv auf die Beständigkeit gegen Elektromigration aus. Bei hochfrequenten Strömen ohne Gleichanteil lässt sich aufgrund der Selbstheilung ein höherer Stromdichtegrenzwert ansetzen. Die meisten digitalen Netze führen solche Ströme. In diesem Fall ist erst bei weiterer Strukturverkleinerung mit Elektromigrationserscheinungen zu rechnen.

Die geringen Leiterbahnabmessungen können gegebenenfalls einen positiven Einfluss auf die Lebensdauer haben. Durch Erwärmen und langsames Abkühlen (*annealing*) während des Herstellungsprozesses lassen sich Strukturen mit großen Korngrößen erzeugen, die einer Bambusstruktur nahe kommen. Aufgrund des bereits erwähnten Bambus-Effekts kann eine solche Bambusstruktur höhere Stromdichten ohne Elektromigrationserscheinungen vertragen.

6 Ausblick

In diesem Beitrag wurden Trends in der Halbleiterindustrie hinsichtlich der Gefährdung durch Elektromigration betrachtet. Dabei lässt sich feststellen, dass das Risiko für Leiterbahnen, durch EM zerstört zu werden, stark steigt. Die hauptsächlichen Einflussgrößen auf Elektromigrationserscheinungen beim Schaltungs- und Layoutentwurf sind Stromdichte und Temperatur. Die Stromdichte in metallischen Leiterbahnen integrierter Schaltkreise wird in den kommenden Jahren zunehmen. Die Temperaturverteilung über den Chip muss betrachtet werden, ebenso wie die Selbsterwärmung der Leiterbahnen, um die tatsächliche Temperatur der Leiterbahn abschätzen zu können. Beide Faktoren tragen zum steigenden Elektromigrationsrisiko bei.

Bereits bei aktuellen Halbleitertechnologien ist Elektromigration eine kritische Randbedingung. Nachdem Aluminium mit seiner höheren Anfälligkeit für EM vielfach durch Kupfer ersetzt wurde, ist nun auch dieses Material stark beansprucht. Zur Erhöhung der Elektromigrationsbeständigkeit der Kupferleiterbahnen sind diese zu modifizieren. Einerseits sind CuSiN- oder CoWP-Beschichtungen zur Veränderung der Oberfläche möglich (da bei Cu ein großer Teil der Migrationsprozesse an der Oberfläche stattfindet), andererseits lässt sich die Leiterbahn durch den Einsatz von Cu-Al-Legierungen stabilisieren [1].

In zukünftigen Schaltkreisen kann bei entsprechender Berücksichtigung im Layoutentwurf eventuell der Blech-Effekt [6] ausgenutzt werden, indem man die Leitungslängen gering hält und so elektromigrationsrobuste Leitungen erzeugt.

Darüber hinaus ist der Einsatz neuer Materialien und Technologien zu untersuchen, beispielsweise die Verwendung von Carbon Nanotubes (CNT) in Vias [7] oder von Graphen-Werkstoffen und Nanowires für die elektrischen Verbindungen, welche geringe oder keine Elektromigrationsanfälligkeit besitzen. Für CNTs werden beispielsweise erreichbare Stromdichten in Höhe von 10^9 A/cm^2 angegeben [8]. Diese positive Eigenschaft lässt sich bisher jedoch nur bei sehr kleinen Strukturabmessungen ausnutzen [9].

Literatur

[1] *ITRS Roadmap*, 2009, URL `http://www.itrs.net/`

[2] Lienig, J.: *Elektromigration und deren Berücksichtigung beim Layoutentwurf*, in: Tagungsband Dresdener Arbeitstagung Schaltkreis- und Systementwurf (DASS), 13–14, 2008

[3] Black, J. R.: *Electromigration—A Brief Survey and Some Recent Results*, IEEE Transactions on Electron Devices, vol. 16, no. 4, 338–347, 1969

[4] Lienig, J.: *Introduction to Electromigration-Aware Physical Design*, in: Scheffer, L. (Ed.), ISPD, 39–46, ACM, San Jose, California, USA, 2006

[5] Tao, J.; Cheung, N.; Hu, C.: *Metal Electromigration Damage Healing under Bidirectional Current Stress*, Electron Device Letters, IEEE, vol. 14, no. 12, 554–556, 1993

[6] Blech, I. A.: *Electromigration in Thin Aluminum Films on Titanium Nitride*, Journal of Applied Physics, vol. 47, no. 4, 1203–1208, 1976

[7] Li, J.; Ye, Q.; Cassell, A.; Ng, H. T.; Stevens, R.; Han, J.; Meyyappan, M.: *Bottom-Up Approach for Carbon Nanotube Interconnects*, Applied Physics Letters, vol. 82, no. 15, 2491–2493, 2003

[8] Tan, C. M.: Electromigration in ULSI Interconnections, World Scientific, 2010

[9] Alam, N.; Kureshi, A. K.; Hasan, M.; Arslan, T.: *Analysis of Carbon Nanotube Interconnects and Their Comparison with Cu Interconnects*, in: IMPACT '09, 124 – 127, 2009

Upgrading a Standard Process Design Kit by Safe Operating Area Checker Feature

Udo Sobe and Dietmar Mörtl
{Udo.Sobe, Dietmar.Moertl}@zmdi.com
ZMD AG, Grenzstraße 28, 01109 Dresden

Abstract

The safe operating area (SOA) data of devices are used to develop reliable integrated circuits. To realize this the SOA data must be available for tools, such as the design environment, the verification tool and optimization tool. There are process design kits (PDKs) where the SOA data are only available inside the documentation. The preparation of the SOA data are necessary to enable the tool support for the complete development process. A common method is presented to extend standard PDKs by SOA checker capability. To support the main tool chain the SOA checker is based on model encapsulation. So it is possible to report SOA violations to a log file during circuit simulation. The model encapsulation was realized by Verilog-A language.

1 Introduction

Design for reliability (DFR) has to be part of circuit development. Especially if the circuit is used for automotive or industry applications DFR is a "must-have". EDA (electronic design automation) companies had implemented techniques for reliability simulation to consider aging [1]. Reliability models are the basis to use the techniques. But today these models are not standard of the PDKs.

PDKs provide information about maximum voltage differences, branch current, power consumption or SOA diagrams. The information is part of the documentation (HTML/PDF files). In some PDKs (e.g. austriamicrosystems: 0.35 µm HV-CMOS [2], XFAB: xh035 [3]) the information is used for so called SOA checker. A SOA checker is applied to recognize and report violation of the maximum ratings inside a circuit.

The technique is used for circuits with different voltage domains, e.g. low and high voltage domains. Known SOA checker implementations use conservative settings to check the voltage. The check of branch current or power consumptions would give additional information.

This paper describes the implementation of SOA checker into an existing automotive certified PDK which does not contain a SOA checker. The first main step is the SOA data collection to one data sheet (e.g. OpenOffice calc). Because of data can be distributed on multiple sources with different formats (e.g. PDF documents, diagrams, email communications, etc.) it is a manual step. The second step integrates the SOA checker into the device models which is supported by scripts. Finally it is necessary to verify the encapsulated models. This verification step is separated into model behavior and SOA checker.

The first section of the paper gives an introduction of SOA methodology. The methods to check SOA limits and the implementation of the method into standard PDK is described in the next two sections. Finally the verification strategy and application is discussed.

2 Method to check SOA limits

The method was chosen by using the overview in [4]. The model encapsulation method is suitable for circuit verification [5, 6] with different analyses (e.g. corner, statistical) and supports analog and mixed-signal simulators (spectre, ultrasim, aps in combination with ams). Furthermore the application is simple to use in the design environment (it is Cadence in our case) and other tools, e.g. zmdAnalyser [7], for manual and automated verification respectively.

Every primitive device of the circuit is encapsulated by a SOA checker to measure electrical conditions at the ports (cf. example in Fig. 1). The reporting of violations is realized to one separate log file.

The SOA checker can be described inside the device models by using hardware description language (HDL) [8] for both parts. We chose another implementation where the basic model description code remains unchanged and the SOA checker code is written in Verilog-A language [9]. This means for the implementation: any modifications are focused on one part of the PDK, which reduces the risk of mistakes, because this part can be verified separately.

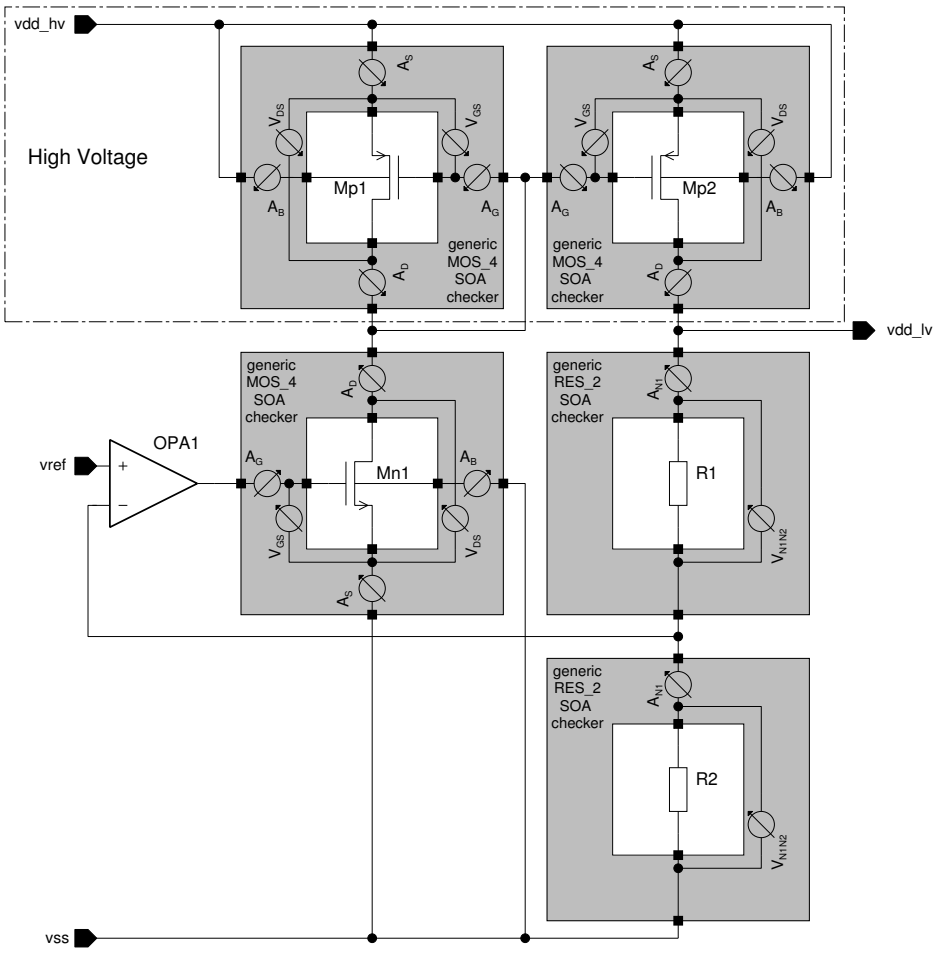


Figure 1: Example of a SOA checker inside a voltage regulator circuit.

3 Implementation of the Method

Figure 2 shows the flow to implement the SOA checker into the standard PDK's device model library. Scripting of a procedure is necessary to modify the model files because today's device model library are complex, e.g. one PDK contains about 20k lines code, 93 sections for 148 primitive devices.

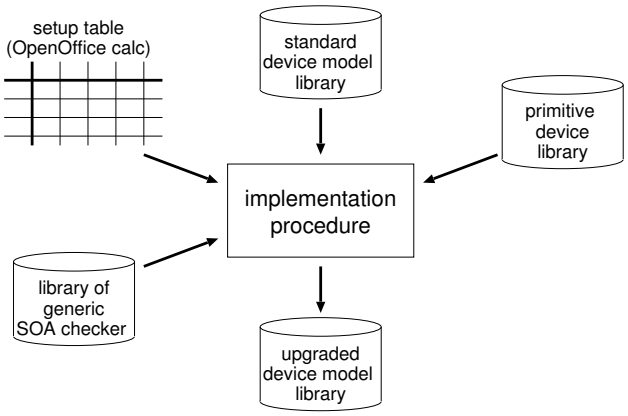


Figure 2: Data flow to implement SOA checker into the PDK.

A setup table is used to control the implementation process. The setup table maps the device model name to a generic SOA checker and related SOA limits (cf. Fig. 3). It is a manual task to collect and fill out the setup table, because the data of SOA limits are shared in the PDK documentation (HTML/PDF/email) or must be requested from FAB.

We use OpenOffice calc format to collect the data of SOA limits into a table. Furthermore the format is suitable for additional documentation e.g. nomenclature, parameter definitions, references to PDK documentation, generic SOA checker description and release status.

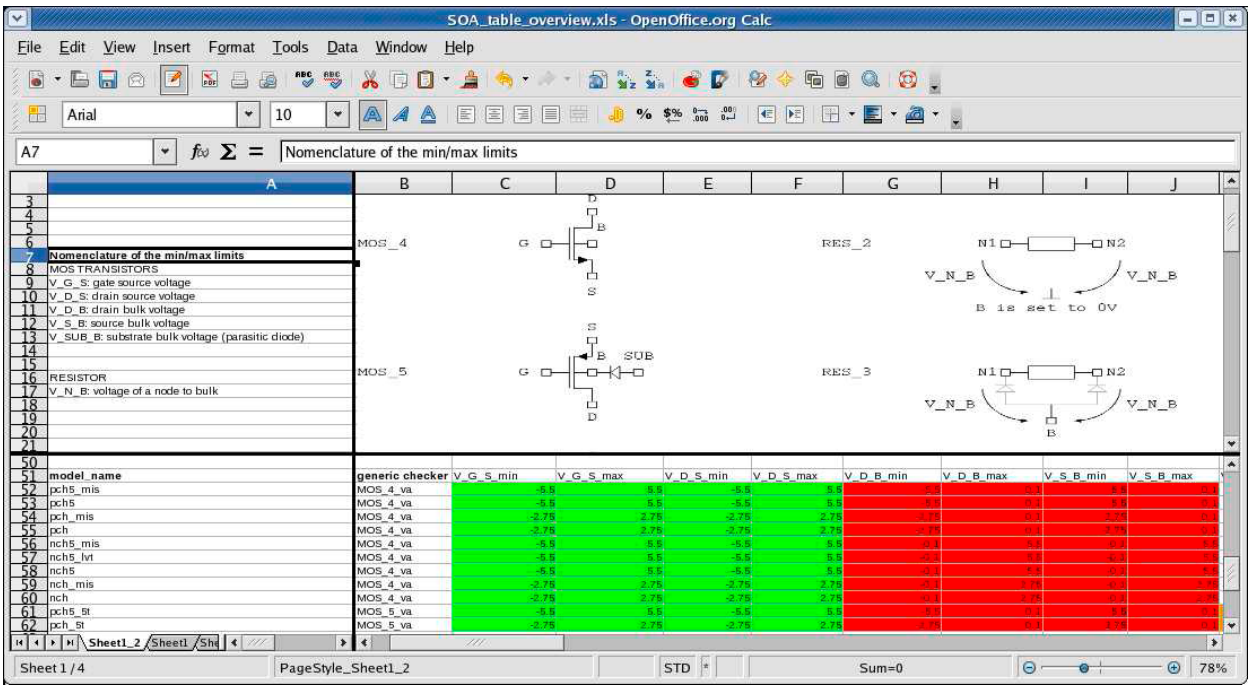


Figure 3: Setup example with documentation and values.

The SOA checkers are grouped into generic types of devices (e.g. MOS, BIP, RES, CAP, DIO) with different number of ports (e.g. 4 – 7 port mos devices). That allows to reuse one generic SOA checker for numerous devices of the same type, e.g. generic SOA checker MOS_4 is used for 4 port n- and p-channel mos devices. The code of 10 SOA checkers was realized to support about 150 devices of the PDK. The generic SOA checker library contains the following items:

- MOS_4, MOS_5, MOS_6, MOS_7
- BIP_3
- RES_2, RES_3
- CAP_2, CAP_3
- DIO_2

Beside the Verilog-A code for checking and reporting the code contains additional information, e.g. port order and SOA parameters, for the implementation scripts.

The library of primitive devices is part of the PDK. Properties and simulation informations are stored for every device by using component description format (CDF).

First the implementation procedure extracts a list of model name and the port order for every device from primitive device library. Now some checks are carried out to ensure compatibility of data between the setup table, the generic SOA checker and model name-port order list:

1. check: missing SOA parameters between setup table and generic SOA checker code
2. check: matching of port order between model name-port order list and generic SOA checker code

Next the device model files are parsed by using information from the setup table. Each model of the model files is processed by using two steps. In the "format" step the model is converted to sub-circuit model format if the current format of the model was not a sub-circuit model. The SOA checker is added to the sub-circuit model in "instantiate checker" step. The script 1 shows the generalized format example of the procedure.

```
subckt subckt_model_name (subckt_port_1 subckt_port_2 subckt_port_3 subckt_port_4)
parameters subckt_model_parameter_name_1=subckt_model_parameter_value_1
+          subckt_model_parameter_name_2=subckt_model_parameter_value_2
...
Q1   (subckt_port_1 subckt_port_2 subckt_port_3 subckt_port_4) model_name
+                                  model_parameter_name_1=subckt_model_parameter_name_1
+                                  model_parameter_name_2=subckt_model_parameter_name_2
...
XSOA (subckt_port_1 subckt_port_2 subckt_port_3 subckt_port_4) SOA_checker_name
+                                  SOA_parameter_name_1=SOA_parameter_value_1
+                                  SOA_parameter_name_2=SOA_parameter_value_2
...
model model_name
...
ends subckt_model_name
```

Script 1: Generalized example of the sub-circuit model with real device Q1 and parallel connected checker XSOA.

4 Verification

The verification of the extended models must be carried out to reduce the risk of mistakes inside the upgraded PDK. The behavior of the all devices must be validated for the upgraded PDK. The verification is divided into two main parts:

- validation of electrical parameters between standard and upgraded PDK,
- validation of SOA checks of the upgraded PDK.

We use a simulation based methodology to realize the verification of a complete primitive device library as shown in Fig. 4a. Starting point is a test bench with all provided primitive devices. Every primitive device is instantiated and connected to a respective device test bench (cf. example in Fig. 4b). The test bench is generated by script.

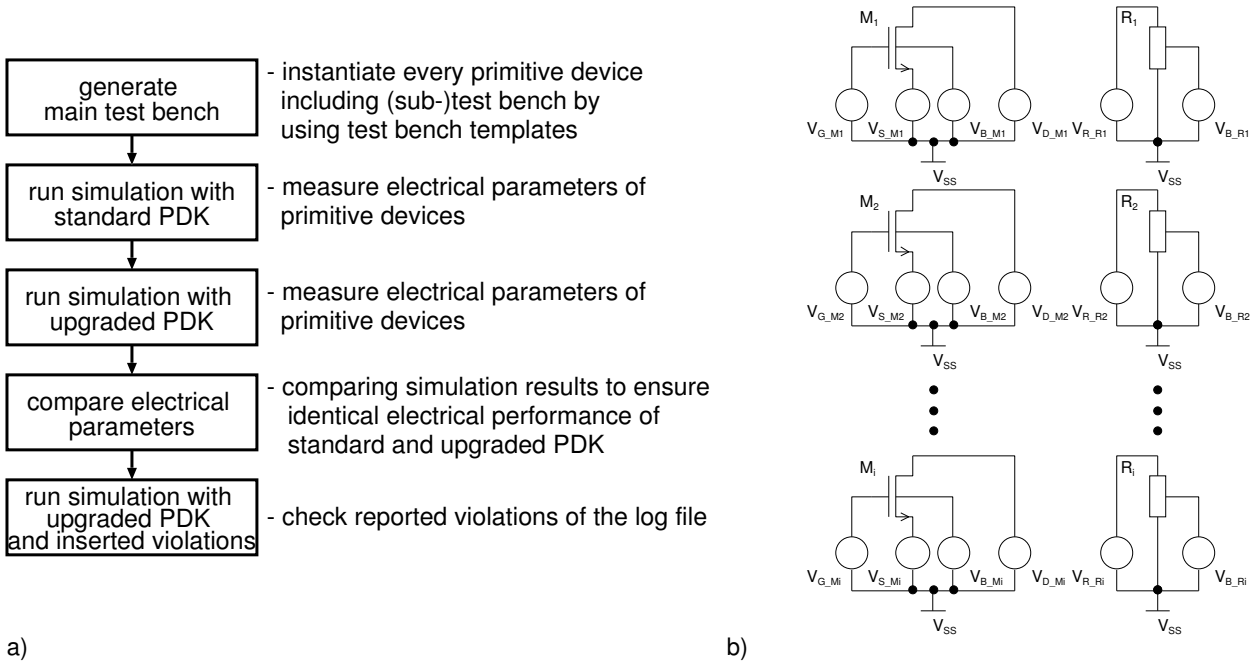


Figure 4: Verification of the upgraded PDK. a) Steps of the simulation based verification. b) Example of the test bench with matrix of device test benches.

The identical test bench is used to simulate and measure electrical behavior with and without SOA checker. The electrical parameters of both PDKs are compared and have to match. In case the model type was converted from standard device model to sub-circuit model the extraction commands for measuring differ between both PDKs, because of the additional hierarchy of the sub-circuit (s. Tab. 1).

standard PDK	**upgraded PDK**
`OP("/I\_dioden" "i")`	`pv("I\_dioden.MO" "i" ?result "dcOpInfo-info")`
`pv("I\_dioden" "i" ?result "dcOpInfo-info")`	`pv("I\_dioden.MO" "i" ?result "dcOpInfo-info")`

Table 1: Extraction commands to measure device performances differ between standard PDK and upgraded PDK for models which model type was converted from standard device model to sub-circuit model.

5 Application of the Method

The SOA checker can be activated in the Cadence's Analog Design Environment (ADE) by setting or enabling the extended models in the model setup window. During the DC or Transient simulation the SOA checker is working and reports the SOA violations into the log file. This log file can be reviewed manually by the design engineer or could be analyzed by a measurement procedure as published in [5, 6].

Furthermore the SOA checker can be combined with corner or Monte Carlo analyses. A configuration for our in-house verification environment zmdAnalyser is prepared and can be chosen for corner or Monte Carlo analyses.

6 Conclusions

The article describes the extension of an existing standard PDK by SOA checker feature. Model encapsulation methodology was chosen and was implemented into the model library. The checker is written with Verilog-A language. The implementation procedure is using one setup table, which contains the data of SOA limits. Today the SOA limits have to be collected from PDK documentations or email requests. Most available SOA limits are voltages. We plan to extend the SOA checker to support SOA limits for power or current if these are provided by FAB.

Important is the verification step of the upgraded PDK. In this step the electrical behavior of devices and the checker functionality must be validated.

References

[1] Cadence. *Reliability Simulation in Integrated Circuit Design*, 2009.

[2] Heimo Gensinger, Martin Schrems, Ehrenfried Seebacher, and Gerhard Rappitsch. *High Voltage CMOS Technologies for robust System-on-Chip Design*, 2006.

[3] XFAB. *OCC Flow for simulation*, 2010.

[4] Karl-Heinz Rooch, Udo Sobe, Dietmar Mörtl, and Michael Pronath. Safe operating areas (SOA) constraining concepts for verification and optimization of analog and mixed signal circuits. In *DASS*, Dresden, 2008.

[5] Udo Sobe, Karl-Heinz Rooch, Dietmar Mörtl, Achim Graupner, Andre Lerch, and Michael Pronath. Verification of safe operating area (SOA) constraints in analog circuits. In *GI/GMM/ITG Fachtagung Zuverlässigkeit und Entwurf*, München, Germany, 2008.

[6] Udo Sobe, Karl-Heinz Rooch, Dietmar Mörtl, Achim Graupner, Andre Lerch, and Michael Pronath. Verification method for analog and mixed circuits considering performance parameters and safe operating area (SOA) constraints. In *Semiconductor Conference Dresden (SCD) 2009*, 2009.

[7] Udo Sobe and Uwe Henniger. zmdAnalyser: Ein Design Tool von Analog/Mixed Signal Designern. *DASS*, pages 29–33, 2005.

[8] Leonid Goldgeisser, Ernst Christen, and Zhichao Deng. Verification of safe operating area (SOA) constraints in analog circuits. In *Design Automation Conference (DAC)*, 2007.

[9] Cadence. *Verilog-A Language Reference*, 2010.

Neuartige Entwurfsmethodik zur Berücksichtigung des IR-Drop bei der Power-Verdrahtung analoger Schaltungen

Andreas Krinke, Jens Lienig
{krinke, lienig}@ifte.de
Technische Universität Dresden, Institut für Feinwerktechnik und Elektronik-Design

Kurzfassung

Bei der Verdrahtung der Versorgungsnetze (Power-Verdrahtung) analoger integrierter Schaltkreise ist unter anderem der maximal zulässige Spannungsabfall (IR-Drop) bis hin zu einzelnen Modulen zu berücksichtigen. Dieser ist vom Stromverbrauch der Module und vom Widerstand des Netzes abhängig. Der Beitrag beschreibt, wie Veränderungen der Stromaufnahme aufgrund von Schwankungen im Herstellungsprozess und Veränderungen der Umgebungsbedingungen geeignet bei der Power-Verdrahtung berücksichtigt werden können.

Einleitung

Die Realisierung der Versorgungsnetze, also der Verdrahtung zur Verbindung von VDD- und GND-Pads mit den entsprechenden Anschlüssen einzelner Module, muss unter der Berücksichtigung vielfältiger Randbedingungen erfolgen. In analogen integrierten Schaltungen sind hier die Beschränkung des Spannungsabfalls (IR-Drop) sowie die Berücksichtigung von Elektromigration (EM) und elektrischer Überlastung (engl. *electrical overstress*, EOS) zur Erhöhung der Zuverlässigkeit besonders hervorzuheben.

Ein IR-Drop auf den Versorgungsnetzen führt durch Absenkung des VDD- bzw. Anhebung des GND-Potentials zur Verringerung der Spannungen an einzelnen Modulen und Bauelementen. Dies reduziert den Rauschabstand der Versorgungsspannung und sämtlicher Signale, womit sich die Anfälligkeit für Rauschen erhöht. Um diesen Effekt zu reduzieren, wird für den Entwurf der Versorgungsnetze ein maximal zulässiger IR-Drop vorgegeben.

Ein weiteres Problem stellen hohe Stromdichten dar, die aufgrund von Elektromigration zu Materialtransport und langfristig zu Leitbahnunterbrechungen oder Kurzschlüssen führen können. Bei noch größeren Stromdichten ereignen sich Unterbrechungen bereits nach kurzer Zeit, da es zu einem plötzlichen Schmelzen oder Verdampfen des Metalls kommt. Sowohl Elektromigration als auch elektrische Überlastung werden im Entwurf durch Begrenzungen der zulässigen Stromdichten berücksichtigt [9].

IR-Drop und Stromdichten sind von den Stromaufnahmen der angeschlossenen Module sowie von Topologie und Widerstand der Versorgungsnetze abhängig. Die Festlegung der Topologie und die Dimensionierung der Leitbahnbreiten müssen deshalb unter Berücksichtigung dieser Stromaufnahmen erfolgen. Gegenläufige Ziele sind dabei die Minimierung der Fläche der Versorgungsnetze und die damit verbundene Verkleinerung der Leitbahnbreiten, um wertvolle Verdrahtungsressourcen einzusparen.

Während für die Bestimmung des maximalen IR-Drop zwischen dem Versorgungspad und den angeschlossenen Modulen die Ströme durch alle Netzsegmente bekannt sein müssen, erfordert die Zuverlässigkeitsprüfung in Hinblick auf Elektromigration und Überlastung üblicherweise die Kenntnis der Durchschnitts-, Effektiv- und Spitzenwerte der zeitabhängigen Segmentströme. Diese charakteristischen Größen variieren gemeinsam mit der Stromaufnahme aller Module aufgrund von (1) Schwankungen im Herstellungsprozess, z. B. durch unterschiedliche Schwellspannungen von Transistoren infolge abweichender Oxiddicken und (2) Veränderungen der Umgebungsbedingungen, wie z. B. der Temperatur, im Betrieb. Dieser Beitrag untersucht erstmals, wie diese Variationen bei der Verdrahtung der VDD- und GND-Netze, geeignet berücksichtigt werden können.

Frühere Arbeiten

Bereits in den 1980er Jahren untersuchten Chowdhury und Breuer [4–7] die flächenoptimale Power-Verdrahtung bei bekannter Baum- oder Graph-Topologie unter Berücksichtigung von IR-Drop und EM. Sie setzten die Kenntnis der Netztopologie und der maximalen Stromaufnahmen aller Module voraus und stellten fest, dass sich der maximale Strom eines Netzsegments nicht zwangsläufig aus den Maximalströmen der Module ergibt, da diese zu unterschiedlichen Zeitpunkten auftreten können. Spätere Arbeiten [8, 10–12] nutzten ebenfalls maximale (worst-case) oder durchschnittliche Stromaufnahmen, um die Verdrahtungsfläche mit verschiedenen Verfahren zu minimieren. In [1] werden zu jedem Pin der Module die Ströme zu all den Zeitpunkten gespeichert, zu denen mindestens ein beliebiger Pin des Netzes seinen Minimal- oder Maximalstrom erreicht. Darauf aufbauend erfolgt eine stromgeführte Verdrahtung, ohne jedoch den maximal zulässigen IR-Drop zu berücksichtigen. Boyd et al. [3] modellierten die Stromaufnahme der Module als Zufallsvariable, deren Erwartungswert, Standardabweichung und Korrelation bekannt sind. Sie stellten ein heuristisches Verfahren vor, das eine robuste Verdrahtung in Graph-Topologie erzeugt. Neuere Arbeiten führen zwar eine stromgeführte Verdrahtung durch, berücksichtigen dabei jedoch nicht den maximalen IR-Drop. Im industriellen Umfeld wird der IR-Drop deswegen erst nach der Verdrahtung durch Simulation verifiziert.

Die genannten Beiträge nutzen verschiedene Modellierungen der relevanten Ströme zur Realisierung einer Verdrahtung mit geringer Fläche. Keine dieser Arbeiten betrachtet Variationen dieser Ströme durch Schwankungen im Herstellungsprozess und Veränderungen der Umgebungsbedingungen, wie z. B. der Temperatur.

Modellierung der Ströme

Die Power-Verdrahtung verbindet die VDD- und GND-Pins der Module mit den Spannungsversorgungspads des Schaltkreises. Ein Modul ist eine Teilschaltung, die sich in derselben Hierarchieebene wie das Versorgungsnetz befindet. Die Stromaufnahme einzelner Module ist zeitabhängig und wird von vielen Faktoren beeinflusst, z. B. Betriebszustand, Temperatur und Parameter der hergestellten Bauelemente. Diese Faktoren werden in Parametersätzen zusammengefasst. Je nach betrachtetem Effekt, sind unterschiedliche Kenngrößen des Stromverbrauchs interessant.

Die Einhaltung des maximal zulässigen IR-Drop $U_{\max}$ erfordert die Betrachtung des Spannungsabfalls auf allen Pfaden zwischen den Versorgungspads und den angeschlossenen Modulen (siehe Bild 1).

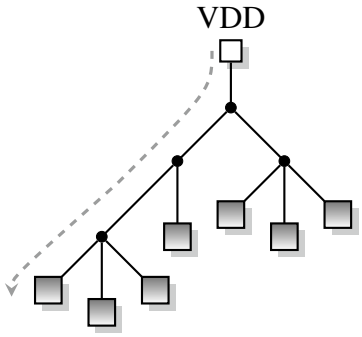


Bild 1: Beispiel für eine Netz-Topologie mit markierten Segmentgrenzen (Punkte) und einem Pfad zwischen Versorgungspad und einem Modul

Dieser Spannungsabfall ist von der Stromaufnahme aller durch einen Pfad versorgten Module abhängig. Es ist jedoch nicht sinnvoll, für diese Ströme das jeweilige Maximum der Werte aller Parametersätze zu wählen. Sie wären zu pessimistisch gewählt, da die entsprechenden Parametersätze Verhältnisse beschreiben, die in der Realität nicht gemeinsam bei einem hergestellten IC eintreten können. Beispiele sind das Auftreten der ermittelten Maximalströme zweier Module bei

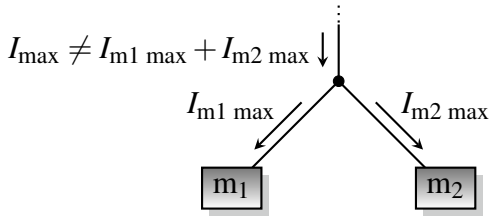


Bild 2: Der Knotensatz verliert für die Betrachtung von Maximalströmen seine Gültigkeit, wenn die maximalen Stromaufnahmen der Module bei unterschiedlichen Parametersätzen auftreten.

unterschiedlichen Betriebszuständen oder weit auseinanderliegenden Umgebungstemperaturen. Demnach ergeben sich die Maximalströme der Netzsegmente unter Umständen bei unterschiedlichen Parametersätzen (siehe Bild 2).

Die unterschiedlichen Parametersätze sind demnach unabhängig voneinander zu betrachten, um denjenigen zu finden, für den der IR-Drop sein Maximum erreicht. Für einen Pfad P aus mehreren Netzsegmenten s ergibt sich die Randbedingung zu

$$\max\left(\sum_{s\in P} \boldsymbol{I}_s R_s\right) \leq U_{\max} \tag{1}$$

mit R_s und $\boldsymbol{I}_s$ als Widerstand bzw. Stromvektor eines Segments. $\boldsymbol{I}_s$ enthält für jeden Parametersatz den Maximalstrom durch das Segment. Für die direkt in einem Modul endenden Segmente lassen sich diese Ströme z. B. durch Corner-Case- oder Monte-Carlo-Simulationen bestimmen. Dabei ist es möglich, die während der Verifikation eines Moduls erhaltenen Simulationsergebnisse erneut zu verwenden. Die Ströme der anderen Segmente ergeben sich in einer Baum-Topologie durch Anwendung des Knotensatzes und in Graph-Topologien durch Lösung des kirchhoffschen Gleichungssystems.

Um die Zuverlässigkeit der Versorgungsnetze sicherstellen zu können, ist die Betrachtung von Elektromigration (EM) und elektrischer Überlastung (engl. *electrical overstress*, EOS) notwendig. Die daraus abgeleiteten maximal zulässigen Stromdichten sind von jedem einzelnen Netzsegment bei allen möglichen Parametersätzen einzuhalten. Im Gegensatz zum IR-Drop erfordern demnach nicht Pfade, sondern einzelne Segmente genauere Betrachtung. Zur Verifikation der entsprechenden Stromdichtegrenzen, sind für jedes Netzsegment *die* Parametersätze zu finden, für die der Durchschnitts-, Effektiv- und Spitzenwert des Stromes (I_{Avg}, I_{RMS}, I_{Peak}) jeweils sein Maximum erreicht [9] (siehe Bild 3).

Genau wie beim IR-Drop werden diese Werte für die einzelnen Module und alle Parametersätze als bekannt vorausgesetzt. Sie ergeben sich aus den Zeitsignalen der Ströme, die für alle Module durch Simulationen ermittelt wurden. In einer Baum-Topologie sind die Segmentströme als Funktion der Stromaufnahme aller Module wiederum mit dem Knotensatz analytisch herleitbar, während in Graph-Topologien erneut die Lösung des Systems der kirchhoffschen Gleichungen notwendig ist.

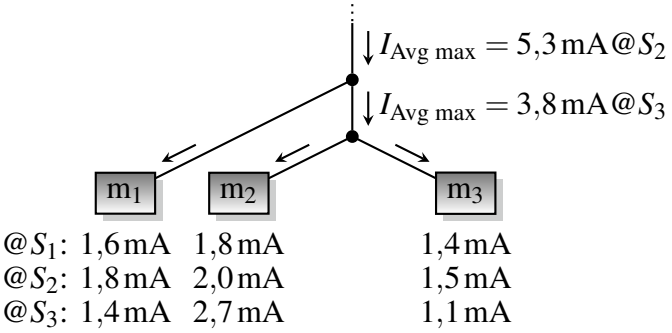


Bild 3: Ausgehend von der durchschnittlichen Stromaufnahme der drei Module bei den Parametersätzen S_1, S_2 und S_3 ergeben sich die maximalen Ströme der Segmente bei unterschiedlichen Parametersätzen.

Anhand der analytischen Zusammenhänge lassen sich $I_{s\,\{\text{Avg, RMS, Peak}\}\,\max}$ für jedes Segment s bestimmen. Sie müssen die Gleichungen

$$\frac{I_{s\,\text{Avg max}}}{w_s \cdot d_s} \leq J_{\text{Avg max}} \qquad \frac{I_{s\,\text{RMS max}}}{w_s \cdot d_s} \leq J_{\text{RMS max}} \qquad \frac{I_{s\,\text{Peak max}}}{w_s \cdot d_s} \leq J_{\text{Peak max}} \tag{2}$$

mit den gegebenen Grenzwerten für die Stromdichte $J_{\{\text{Avg, RMS, Peak}\}\,\max}$ und dem Leitbahnquerschnitt $w_s \times d_s$ erfüllen.

Methodik zur Power-Verdrahtung

Die im Folgenden beschriebene Methodik ist sowohl für VDD- als auch für GND-Netze anwendbar. Die Realisierung der Power-Verdrahtung beginnt mit der manuellen Erstellung einer ersten Version der Verdrahtung. Diese definiert die Topologie des betrachteten Netzes. Es sind dabei beliebige Leitbahnbreiten nutzbar, da diese später automatisch optimiert werden. Im Gegensatz dazu bleiben die Längen der einzelnen Segmente und die verwendeten Verdrahtungsebenen unverändert, da sie die Grundlage für die Optimierung bilden.

Nach der Fertigstellung der ersten Version schließt sich die analytische Herleitung der Ströme aller Netzsegmente als Funktionen der Versorgungsströme sämtlicher angeschlossener Module an. Wie im letzten Abschnitt beschrieben, lassen sich anschließend die maximalen Durchschnitts-, Effektiv- und Spitzenwerte der Segmentströme bestimmen.

Im nächsten Schritt erfolgt die Anpassung der Segmentbreiten mit dem Ziel, die Gesamtfläche des Versorgungsnetzes unter Einhaltung aller Randbedingungen zu minimieren. Diese Fläche ergibt sich nach Gleichung (3) aus den Längen l und den Breiten w aller Segmente des Netzes N.

$$A = \sum_{s \in N} l_s w_s \tag{3}$$

Die Segmentbreiten w_s dürfen dabei nicht kleiner als durch die Technologie vorgegebenen Minimalbreiten sein. Die Begrenzung des IR-Drop führt ausgehend von Gleichung (1) mit dem Flächenwiderstand $R_{s\,\square}$ zu den Nebenbedingungen aus Gleichung (4).

$$\max\left(\sum_{s \in P} \boldsymbol{I}_s \frac{l_s}{w_s} R_{s\,\square}\right) \leq U_{\max} \quad \text{für alle Pfade } P \tag{4}$$

Zur Berücksichtigung von EM und EOS kommen die Randbedingungen aus Gleichung (2) hinzu.

Formulierung als Optimierungsproblem

Die genannten Gleichungen bilden ein Optimierungsproblem zur Bestimmung der Segmentbreiten, mit einer linearen Zielfunktion und sowohl linearen als auch nichtlinearen Nebenbedingungen:

$$\text{Minimiere} \qquad \sum_{s \in N} l_s w_s \tag{5}$$

$$\text{unter den Nebenbedingungen} \qquad w_s \leq w_{\min}(s) \tag{6}$$

$$\sum_{s \in P} \boldsymbol{I}_s \frac{l_s}{w_s} R_{s\,\square} \preceq U_{\max} \qquad \text{für alle } P \tag{7}$$

$$\frac{I_{s\,\text{Avg max}}}{w_s \cdot d_s} \leq J_{\text{Avg max}} \tag{8}$$

$$\frac{I_{s\,\text{RMS max}}}{w_s \cdot d_s} \leq J_{\text{RMS max}} \tag{9}$$

$$\frac{I_{s\,\text{Peak max}}}{w_s \cdot d_s} \leq J_{\text{Peak max}} \tag{10}$$

Dabei ist $w_{\min}(s)$ die kleinstmögliche Breite des Segments s. Bei Gleichung (7) handelt es sich um einen elementweisen Vergleich zwischen dem resultierenden Vektor auf der linken Seite und dem maximal zulässigen IR-Drop $U_{\max}$.

Für alle Segmente gilt $l_s \geq 0$, $w_s \geq 0$, $d_s \geq 0$ und $R_{s\,\square} \geq 0$. Weiterhin wird für die Grenzwerte der Stromdichte $J_{\{\text{Avg, RMS, Peak}\}\,\max} \geq 0$ angenommen. Durch die Verwendung des betragsmäßigen Maximums der Stromparameter I_{Avg}, I_{RMS} und I_{Peak} in den Gleichungen (8)–(10) werden diese Funktionen der Leitbahnbreiten w_s konvex, d. h. es gilt

$$f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y) \tag{11}$$

für alle $x, y \in \mathbb{R}$ und alle $\alpha, \beta \in \mathbb{R}$ mit $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$ [2].

Die verbliebene nichtlineare Nebenbedingung aus Gleichung (7) wird konvex, falls alle Elemente der Stromvektoren $\boldsymbol{I}_s$ positiv sind. Dies lässt sich durch (1) die Vorgabe einer Baum-Topologie als erste Version der Verdrahtung und (2) die Annahme, dass die von den Modulen aufgenommenen Ströme zu jedem Zeitpunkt positiv sind, erreichen. Beide Forderungen stellen besonders in integrierten analogen Schaltungen keine bedeutenden Einschränkungen dar. Sie führen dazu, dass die über den Netzsegmenten abfallenden Spannungen auf allen Pfaden vom Pad bis zu den Modulen immer positiv sind.

Das Ergebnis dieser Überlegungen ist ein konvexes Optimierungsproblem, da sämtliche nichtlinearen Nebenbedingungen konvex sind. Der Vorteil dieser Art der Formulierung ist, dass jedes lokale Optimum gleichzeitig auch ein globales Optimum ist. Weiterhin existieren sehr effiziente Lösungsmethoden, z. B. Innere-Punkte-Verfahren [2].

Zusammenfassung und Ausblick

Bisher wird im industriellen Umfeld die Einhaltung des maximal zulässigen IR-Drop erst nach der Verdrahtung durch Simulationen überprüft. Dieser Beitrag beschreibt eine neuartige Methodik für die Verdrahtung von Versorgungsnetzen in Baum-Topologie in analogen integrierten Schaltkreisen unter Beachtung des zulässigen Spannungsabfalls. Gleichzeitig werden auch Entwurfsregeln zur Berücksichtigung von Elektromigration und elektrischer Überlastung beachtet. Es wird ein Optimierungsproblem formuliert, dessen Lösung die Segmentbreiten des Netzes repräsentiert. Die Formulierung berücksichtigt dabei die von den versorgten Modulen aufgenommenen Ströme bei unterschiedlichen Parametersätzen. Dadurch findet eine Abbildung der Schwankungen beim Herstellungsprozess und unterschiedlicher Umgebungsbedingungen statt.

Ausgehend von diesen Ergebnissen ist die experimentelle Verifikation des Verfahrens Gegenstand der weiteren Forschung.

Literatur

[1] Adler, T., H. Brocke, L. Hedrich und E. Barke: *A Current Driven Routing and Verification Methodology for Analog Applications*. In: *Proc. 37th Design Autom. Conf.*, DAC, 2000, S. 385–389.

[2] Boyd, S. und L. Vandenberghe: *Convex Optimization*. Cambridge University Press, 2004.

[3] Boyd, S., L. Vandenberghe, A. El Gamal und S. Yun: *Design of Robust Global Power and Ground Networks*. In: *Proc. Int'l Symp. on Phys. Design*, ISPD, 2001, S. 60–65.

[4] Chowdhury, S.: *An Automated Design of Minimum-Area IC Power/Ground Nets*. In: *Proc. 24th Design Autom. Conf.*, DAC, 1987, S. 223–229.

[5] Chowdhury, S. und M. A. Breuer: *The construction of minimal area power and ground nets for VLSI circuits*. In: *Proc. 22nd Design Autom. Conf.*, DAC, 1985, S. 794–797.

[6] Chowdhury, S. und M. A. Breuer: *Minimal Area Design of Power/Ground Nets Having Graph Topologies*. IEEE Trans. on Circuits and Systems, 34(12):1441–1451, Dez. 1987.

[7] Chowdhury, S. und M. A. Breuer: *Optimum Design of IC Power/Ground Nets Subject to Reliability Constraints*. IEEE Trans. on CAD, 7(7):787–796, Juli 1988.

[8] Dutta, R. und M. Marek-Sadowska: *Automatic Sizing of Power/Ground (P/G) Networks in VLSI*. In: *Proc. 26th Design Autom. Conf.*, DAC, 1989, S. 783–786.

[9] Jerke, G. und J. Lienig: *Early-Stage Determination of Current-Density Criticality in Interconnects*. In: *Proc. 11th Int'l Symp. on Quality Electronic Design*, ISQED, 2010, S. 667–674.

[10] Kolla, R.: *A Dynamic Programming Approach to the Power Supply Net Sizing Problem*. In: *Proc. European Design Autom. Conf.*, EDAC, 1990, S. 600–604.

[11] Tan, S. X. D., C. J. R. Shi und J. C. Lee: *Reliability-Constrained Area Optimization of VLSI Power/Ground Networks Via Sequence of Linear Programmings*. IEEE Trans. on CAD, 22(12):1678–1684, Dez. 2003.

[12] Wu, X., X. Hong, Y. Cai, C. K. Cheng, J. Gu und W. Dai: *Area Minimization of Power Distribution Network Using Efficient Nonlinear Programming Techniques*. In: *Proc. Int'l Conf. on CAD*, ICCAD, 2001, S. 153–157.

SystemC-AMS unterstütztes Design eines eingebetteten Analog-/Mixed-Signal-Systems mit kostengünstigem Mikrocontroller

Dipl.–Ing. Komla Agla
komla.agla@imms.de

Institut für Mikroelektronik- und Mechatronik-Systeme gemeinnützige GmbH
Ehrenbergstraße 27, 98693 Ilmenau, Deutschland

Kurzfassung

In der vorliegenden Arbeit wird ein neues Konzept zum Entwurf von eingebetteten Analog Mixed-Signal Systemen, basierend auf dem Einsatz der Hardwarebeschreibungssprache SystemC-AMS präsentiert. Die vorgeschlagene Methodik wurde für die Modellierung eines intelligenten Thermoelement-Smart-Sensor-Systems verwendet. Der Fokus bei der Modellierung liegt auf der frühzeitigen Implementierung, Verifikation und Validierung eines effizienten Mikrocontroller-Programms, das den Kennlinienkorrekturalgorithmus des Thermoelements nachbildet. Durch den Einsatz von SystemC-AMS kann die Simulation des Komplettsystems im Vergleich z.B. zu einer Simulation auf RTL-Ebene beschleunigt werden.

Die Ergebnisse demonstrieren die Vorteile von SystemC-AMS für die Modellierung von eingebetteten Analog-/Mixed-Signal-Systemen mit kostengünstigem Mikrocontroller.

1. Einleitung

Der Entwurf von komplexen eingebetteten Analog/Mixed-Signal (E-AMS) Systemen ist sehr kosten- und zeitaufwändig. Mit dem Ziel, Entwicklungszeit und -Kosten zu reduzieren, wurde daher ein Designflow analog-digitaler Hardware-Software-Systeme entwickelt. Die Analyse zeigte, dass die Wahl einer geeigneten Systembeschreibungssprache die „time-to-market“ stark reduzieren kann. Eines der bekanntesten und meist verwendeten Modellierungs- und Simulationswerkzeuge ist MATLAB/Simulink[2]. In der vorliegenden Arbeit wird ein neues Designflow-Konzept zum Entwurf von komplexen Analog/Mixed-Signal-Systemen basierend auf dem Einsatz der seit kurzem als Standard anerkannten Hardwarebeschreibungssprache SystemC-AMS präsentiert. Im Anschluss wird das vorgestellte Designflow-Modell mit anderen Entwicklungsmodellen (wie z.B. dem V-Modell) verglichen.

Durch die Möglichkeit einer gemeinsamen Modellierung und Simulation des Gesamtsystems erlaubt SystemC-AMS die Berücksichtigung und Bewertung physikalischer Einflussgrößen bereits auf Systemebene, die frühzeitige Evaluation unterschiedlicher Designentscheidungen bzgl. Architektur, Algorithmen und Hardware-Software-Partitionierung sowie die Identifikation und Analyse der resultierenden, wechselseitigen Anforderungen und Auswirkungen im Rahmen des komplexen Gesamtsystemkonzepts. Auf diese Weise unterstützt SystemC-AMS als Schnittstelle den Designflow, die Kommunikation zwischen den Entwicklerteams und begünstigt somit den Entwicklungsprozess bzgl. Zuverlässigkeit und Geschwindigkeit.

Exemplarisch wurde die vorgeschlagene Entwurfsmethodik bei der Modellierung eines intelligenten Smart-Sensor-Systems erprobt. Der Fokus bei der Modellierung liegt auf der frühzeitigen Implementierung, Verifikation und Validierung eines Programms für einen effizienten low-cost Mikrocontroller, das den Kennlinienkorrekturalgorithmus eines Thermoelement-Sensors umsetzt. Neben den sich wechselseitig beeinflussenden Anforderungen zwischen analogem Frontend, digitalem Backend und den in Software realisierten Algorithmen spielen dabei vor allem die Einflüsse physikalischer Störgrößen eine wesentliche Rolle. Eine Leistungsbewertung (Simulationsgeschwindigkeit, Genauigkeit) des SystemC-AMS-Modells des Komplettsystems wurde gegenüber einer äquivalenten MATLAB/Simulink-Variante durchgeführt. Die Unterstützung der Bottom-up-Verifikation wird im Beispiel durch eine

digitale Abstraktion gezeigt, bei der das VHDL-RTL-Level-Modell eines Mikrocontrollers in ein SystemC-ESL-Level-Modell übersetzt und im Systemkontext validiert wurde. Durch den Einsatz von SystemC-AMS kann die Simulation des Komplettsystems im Vergleich z.B. zu einer Simulation auf RTL-Ebene beschleunigt werden.

Abschließend werden durch die Darstellung der gewonnenen Ergebnisse wesentliche Vor- und Nachteile von SystemC-AMS für die Modellierung von eingebetteten Analog-/Mixed-Signal-Systemen mit kostengünstigem Mikrocontroller zusammengefasst.

2. Entwurf mit SystemC-AMS

Die Hardwarebeschreibungssprache SystemC-AMS

SystemC-AMS bietet eine Plattform für funktionale Modellierung, Architekturexploration, Integrations-Validierung und Virtual-Prototyping von eingebetteten Analog-/Mixed-Signal Systemen, wie in **Bild 1** dargestellt. Diese Anwendungsgebiete verlangen eine Modellierungs- und Simulationsmethodik, die noch abstrakter als existierende HDLs für Analog- /Mixed-Signal Systeme ist. Gleichzeitig sollen dabei AMS-Komponenten und die Interaktion mit Hardware/Software Systemen modelliert werden können. Eine signifikante Anforderung bei der Modellierung mit SystemC-AMS ist die Einhaltung einer akzeptablen Simulationsperformanz bei der Modellierung des Architekturverhaltens mit hinreichender Genauigkeit. Die ausführbare Spezifikation als Anwendungsfall steht im Mittelpunkt eines modellbasierten Designs, da sie als Basis für sämtliche Arbeiten bei der Systementwicklung dient und dazu ermöglicht, das korrekte Verständnis der System-Spezifikation durch Simulation zu überprüfen. Die ausführbare Spezifikation bringt Strukturen und Algorithmen ein, die im späteren Entwicklungsstadium nur noch sehr schwer zu ändern sind. Daher sollten architekturabhängige Strukturen und die Algorithmen sorgfältig ausgewählt werden[1]. SystemC-AMS bietet 3 MoCs („*Model of Computation*“). Diese sind TDF (*Timed Data Flow*), LSF (*Linear Signal Flow*) und ELN (*Electrical Linear Networks*) und definiert somit den wesentlichen Modellierungsformalismus.

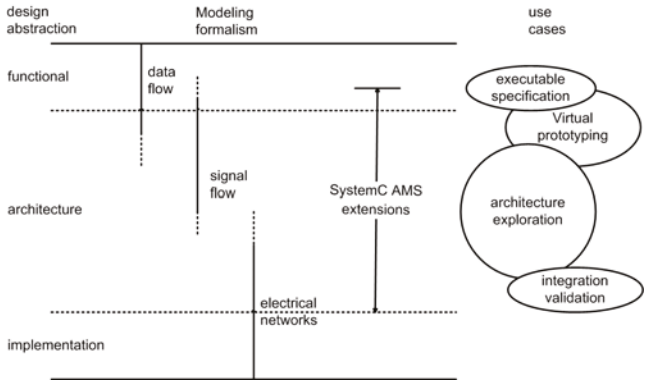


Bild 1: Modellierungsformalismus und die Anwendungsfälle zwischen funktionaler Ebene und der Implementierung.

Entwurfsablauf

Ein möglicher Entwurfsablauf basierend auf dem Einsatz von SystemC-AMS wird durch **Bild 2** illustriert. Ausgehend von den Systemanforderungen wird als erster Schritt des Entwurfs eine Architektur des gesamten Systems konzipiert. SystemC-AMS als Modellierungs- und Simulationswerkzeug erlaubt schon auf dieser Abstraktionsebene, die Architektur als ein Simulationsmodell nachzubilden, ein wichtiger Schritt für weitere Designentscheidungen. Beim Design, speziell von E-AMS-Systemen, ist die Einhaltung folgender Anforderungen notwendig, wie in [6] ausgedrückt ist: „*Low-cost, Short time to market, Small size to market, Real-time constraints, latency, throughtput, Lowpower & low energy consumption, Safety, Data accuracy, Robusteness, Flexibilty in developping new applications*“.
Die Beachtung dieser Anforderungen hilft bei der Wahl geeigneter Komponenten für das gesamte System: einerseits die Analogkomponenten (Analog Front-end) und andererseits die

digitalen (HW/SW) Komponenten. Wie oben schon erwähnt, wird bei der Modellierung als erstes eine ausführbare Spezifikation (SystemC-AMS Modell) des gesamten Systems realisiert, wobei die Analog Mixed-Signal-Systemkomponenten ideal (ohne Fehlerquellen) beschrieben sind. Dadurch wurde ein sehr schnelles Simulationsmodell des Systems erstellt. Mit diesem Modell werden wichtige Designentscheidungen getroffen; z.B. Analyse und Auswahl des Algorithmus, der später auf dem Mikrocontroller ausgeführt wird und die Auswahl eines geeigneten Mikrocontrollerkerns ausgewählt. Die Anforderungen an das AFE (Analog Front End) sowie die notwendige Verstärkung des Operationsverstärkers, Auflösung des ADC und andere wichtige Parametern wurden mit Hilfe des Modells ermittelt.

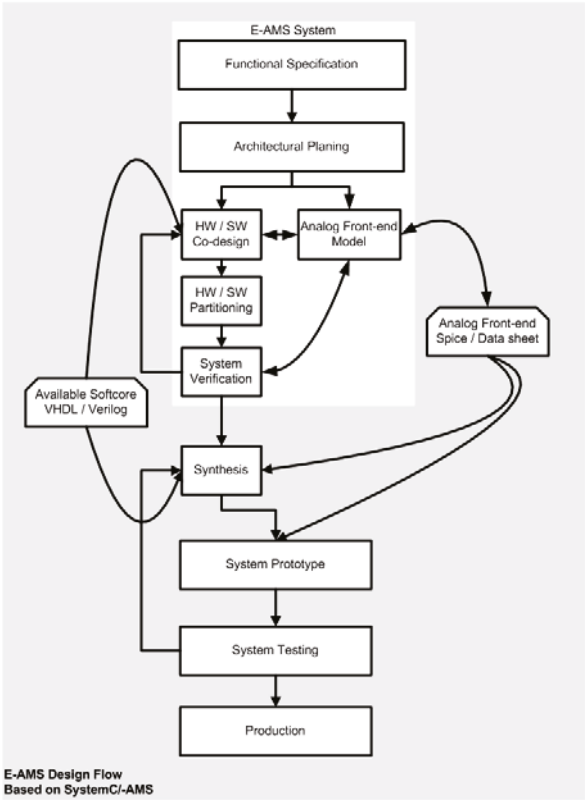


Bild 2: E-AMS-Design-Ablauf mit SystemC-AMS.

Die meisten verwendeten Mikrocontroller (Softcore) existieren als HDL-Modelle (RTL-Modelle VHDL/Verilog), wenige existierende sind SystemC-Modelle (ISS, „*Instruction Set Simulator*“)[7]. Für ein erfolgreiches Hardware-Software-Codesign mit SystemC und dem Ziel einer frühzeitigen Implementierung und Validierung des Mikrocontrollers ist es notwendig, einen Mikrocontroller-Kern auf diese Systemebene zu transformieren. Einige komplexe Mikrocontroller-Kerne, wie MicroBlaze, wurden als SystemC/TLM-Modelle übersetzt[7], ansonsten ist es notwendig, diese manuell oder automatisch in SystemC zu übersetzen.
Der Software-Entwickler realisiert schon auf dieser Abstraktionsebene mit dem ISS seine Software-Entwicklung. Dazu ist es notwendig, das Analog Frontend (z.B. durch Hinzufügen von Störgrößen, wie Rauschen, Offset- und Verstärkungsfehlern, Temperaturdrift etc.), zu verfeinern. Diese Verfeinerung wird anhand des Datenblatts bzw. des SPICE-Modells des AFE realisiert. Das Systemlevel-Modell wird dann durch Simulation verifiziert; bei einer erfolgreichen Validierung der Verifikation erfolgt die Synthese auf einer niedrigeren Abstraktionsebene des Designflows. Der vorgestellte Entwurfsstil ist ein Top-Down-Design mit Bottom-Up-Methoden zur Verfeinerung, Verifikation und Validierung der SystemC-AMS-Modelle. Die Tatsache, dass der HDL-Kern (RTL-Ebene) auf höherer Abstraktionsebene (Systemebene) synthetisiert werden muss, macht den Unterschied zu einem normalen Entwicklungsprozess, wie dem V-Modell[11].

Dieser Entwurfsstil wird anhand eines Beispiels eines E-AMS-Systems basierend auf einem Smart Sensor mit Thermoelementen erprobt und bestätigt.

3. Implementierung

Die Anforderung

Die am häufigsten gemessene Prozessgröße im Automatisierungsumfeld ist die Temperatur. Bekannte Bauelemente, wie Widerstandstemperatursensoren, Thermistoren, Thermoelemente oder Dioden, werden verwendet, um die absolute Temperatur sowie die Temperaturänderung zu ermitteln. Exemplarisch wird ein eingebettetes Analog-/Mixed-Signal-System genutzt, das ein Temperatur-Mess/-Sensor-System basierend auf Thermoelementen modelliert. Thermoelemente werden sowohl im industriellen als auch im kommerziellen Bereich eingesetzt und können je nach Typ Temperaturen bis zu 1300°C messen. Der Mikrocontroller soll eine Kennlinienkorrektur des Thermoelements (ausgewählter Typ: N) durchführen, mit den Anforderungen, dass der Temperaturapproximationsfehler innerhalb einer zulässigen Grenze (<1K) liegt und der Mikrocontroller-Speicher-Verbrauch und die Prozessorrechenleistung minimal sein sollen.
Die Modellierung erfolgte mit SystemC-AMS für die Analogkomponenten und SystemC für die Digital- und Softwarekomponenten. Das Modell auf Basis der oben genannten Anforderungen wird mit verschiedenen Simulationskonfigurationen der Systemkomponenten simuliert und verifiziert.

Das System

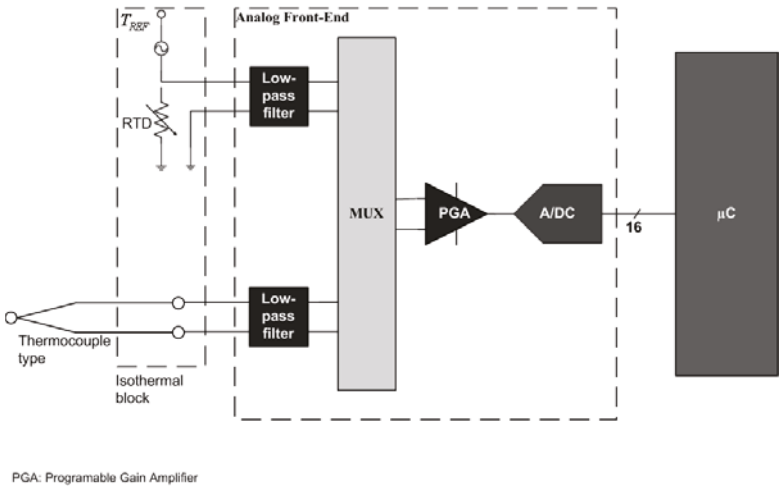


Bild 3: Funktionales Blockdiagramm eines Thermoelement-Messfühlers[5].

Der Leitkreis eines Thermoelements besteht aus zwei unterschiedlichen Metallen bzw. Halbleitern, die an einem Ende (Meßstelle) miteinander verbunden sind. Für die Temperaturmessung müssen die beiden offenen Enden (Vergleichsstelle) auf konstanter Temperatur gehalten werden. Durch Thermodiffusion entsteht eine Spannung (Seebeck-Effekt), die Thermospannung (durch Kennlinie realisiert). Jede Temperaturmessung mit Thermoelementen ist im Prinzip eine Differenzmessung zwischen Messstellen und Vergleichsstellentemperatur (**Bild 3**). Zur Bestimmung der Vergleichsstellentemperatur wird gewöhnlich ein Widerstandstemperatursensor (RTD, „*Resistance Temperature Detector*“) eingesetzt. Der Platin-RTD (Pt100/PT1000) wird am häufigsten verwendet, da er einen breiten (-220°C…+850°C) Temperaturbereich, eine sehr hohe Meßgenauigkeit (bis zu 0.01K) und eine fast lineare Kennlinie besitzt [8]. Ein typisches System zur Messung von Temperatur anhand eines Widerstandstemperatursensors besteht aus einem Tiefpass, einem Operationsverstärker einem Analog-Digital-Umsetzer und einem Mikrocontroller (**Bild 3**) [3].

Der Algorithmus

Die gemessene Thermospannung (Umeas) in **Bild 3** ist von der Temperatur des Thermoelements (T_{TC}), der Vergleichsstelle (T_{REF}) und von dem verwendeten Thermoelement-Typ abhängig. Die Thermospannung ist in Bezug auf die Temperaturdifferenz zwischen Messstelle und Vergleichsstelle[3] nichtlinear. Das Thermoelementverlaufsverhalten ist durch ein direktes bzw. ein inverses Polynom nachgebildet[9]. Durch die Anwendung der inversen Polynomfunktion auf die gemessene Spannung des Thermoelements wird die gesuchte Temperatur des Thermoelements ermittelt.

Die Modellierung

Zur Untersuchung des Algorithmus wurde das gesamte System (ohne ISS) mit SystemC-AMS modelliert. Um das erstellte SystemC-AMS Modell zu bewerten, wurde eine vergleichbare MATLAB/Simulink Version modelliert. Dies erfolgte unter Zuhilfenahme der vorhandenen Modelle (z.B. ADC) aus dem Werkzeug. Deshalb konnte das Modell in sehr kurzer Zeit erstellt werden, während mit SystemC-AMS eine Erweiterung von C++, die Implementierung mehr Zeit in Anspruch genommen hat. Die SystemC-AMS-Variante mit TDF beansprucht deutlich weniger Rechenzeit (2.47min) als die MATLAB/Simulink-Implementierung (34.7min) auf einem Intel Core(TM)2 Duo CPU T7250 @ 2x2.00GHz Prozessor mit 2GB RAM.

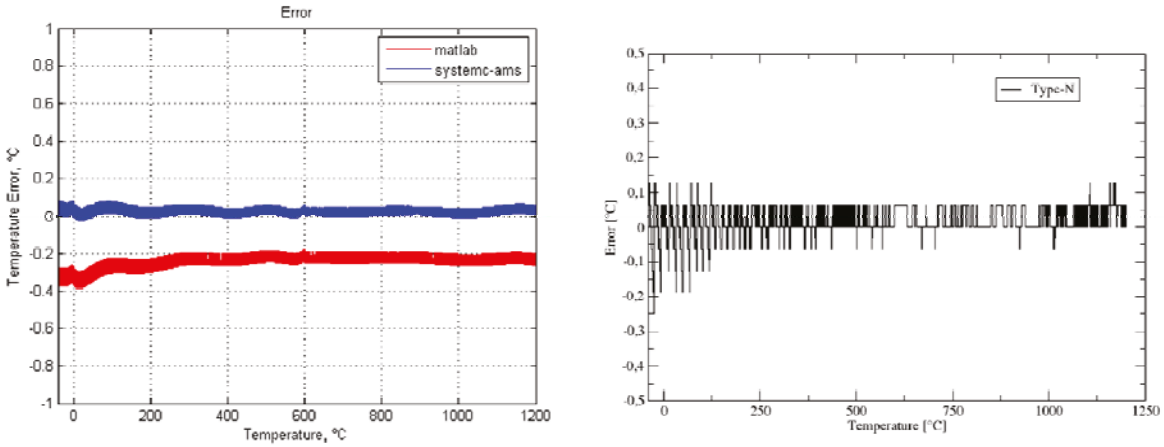
Das erste entworfene Modell erlaubte nur, das Kennlinienkorrekturverfahren zu überprüfen, aber noch nicht alle verlangten Anforderungen (Betrachtung von Störgrößen) zu realisieren. Eine direkte Implementierung der Polynome (PT1000 bzw. Thermoelement) auf dem Mikrocontroller ist sehr schwierig zu realisieren und würde die Mikrocontrollerressourcen und Leistung übersteigen, da die verwendeten Datentypen (Double bzw. Float) nicht für Low-Cost Mikrocontroller (hier Intel 8051 als ausgewählter Prozessor) geeignet sind. Eine Möglichkeit, das Problem zu umgehen, ist die Benutzung einer Lookup-Tabelle [10]. Zwei Lookup-Tabellen wurden zur Vereinfachung der Berechnung des Controllers erstellt. Das PT1000-LUT beinhaltet die Thermospannungswerte und das Thermoelement-LUT die Temperaturwerte, beide im 16 Bit Format. Zwischen zwei aufeinanderfolgenden Stützpunkten wird interpoliert.

Als Analog-Frontend wurde ein ADS1148 ausgewählt und die wichtigsten Funktionen mit SystemC-AMS (TDF) modelliert. Der ADS1148 ist 16-Bit Analog-to-Digital Converter von Texas Instrument, besitzt unter anderen ein PGA („Programable Gain Amplifier“) und kann die Signale von bis zu 4 Sensoren gleichzeitig verarbeiten. Wie im Designkonzept beschrieben, wurde der frei verfügbare Mikrocontroller-Kern Oregano mc8051(VHDL), der mit dem Original 8051 Befehlssatz kompatibel ist, manuell in SystemC übersetzt[12]. Das übersetzte Modell wurde anhand des VHDL-Modells in der Cadence IUS Entwicklungsumgebung verifiziert und validiert. Als Entwicklungsumgebung wurde microVison von Keil verwendet. C und Assembler werden damit kompiliert und ein Intel-HEX-File wird dabei generiert. Die generierten LUTs liegen als C-File vor und werden direkt im ROM des Mikrocontrollers implementiert.

4. Simulationsergebnisse

Bild 4a zeigt die Simulationsergebnisse des ersten Entwurfs. Der dabei entstandene Offset zwischen dem SystemC und dem MATLAB Modell wird durch die unterschiedliche Realisierung der PT1000 Kennlinie verursacht. Der mittlere Fehler der beiden Modelle ist annähernd gleich. Somit wurde das entworfene SystemC-AMS Modell mit dem MATLAB Modell verifiziert.

Nach verschiedenen Konfigurationen des AFE und Simulationen des gesamten Systems (ISS-Version) wurden die Parameter ermittelt, die die Systemanforderungen erfüllen. **Bild 4b** zeigt eine Temperaturapproximationsfehlerausgabe für das Thermoelement Typ-N. Der maximale Fehler ist kleiner als 1K und der Ressourcenverbrauch beträgt weniger als 1 kBytes (348 Bytes für die LUTs).



a) b)

Bild 4: Temperaturfehler-Ausgabe für Thermoelement Typ-N **a)** Ohne ISS, **b)** mit ISS

5. Zusammenfassung

Die vorliegende Arbeit präsentiert einen Ansatz zur Nutzung von SystemC/-AMS im Entwicklungsprozess von eingebetteten Analog-Mixed-Signal Systemen. Der Einsatz von SystemC-AMS auf Systemebene erlaubt eine vollständige Implementierung, Verifikation und Validierung des Software-Programms. Dafür war es notwendig, Bottom-Up-Methoden für die Verfeinerung von einigen Systemkomponenten zu benutzen.

Durch die Erprobung der heterogenen Modellierungs- und Simulationsplattform SystemC-AMS wurden neue Erkenntnisse gewonnen, welche die Entwurfs- und Verifikationszeit von komplexen Analog/Mixed-Signal-Systemen erheblich reduzieren. Die Eignung von SystemC-AMS für die Modellierung von eingebetteten Analog-/Mixed-Signal-Systemen mit kostengünstigem Mikrocontroller wurde dargestellt.

Literatur

[1] C. Grimm, M. Barnasconi, A. Vachoux, K. Einwich, „An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS Extensions", June-2008.

[2] M. Agostinelli, R. Priewasser, M. Huemer, S. Marsili, D. Straeussnigg „SystemC-AMS modeling and simulation of digitally controlled DC-DC converters", 2010.

[3] B. Boris, Z. Hocenski, L. Cvitas, „Optimal Approximation Parameters of Temperature Sensor Transfer Characteristic for Implementation in Low Cost Microcontroller Systems", July-2006

[4] H. Hang, J. Song, F. Yang, X. Fan, „Design of Smart Temperature Transmitter", 2009.

[5] B. Schuh, Watlow, „Smart Thermocouple System for Industrial Temperature Measurement", 2001

[6] A. Doboli, „Introduction to Mixed-Signal, embedded Systems Design", 2006.

[7] www.soclib.fr.

[8] B. C. Baker, "Precision Temperature-Sensing with RTD Circuits", 2008.

[9] www.omega.com, „ITS-90 Thermocouple Direct and Inverse Polynomials".

[10] C. Kosmatopoulos, N. Tsagourias, „Using a Look-Up Table for Code Execution in Small Microcontrollers System with Dash Memory" 2002.

[11] K. Reif, „Automobilelektronik: Eine Einführung für Ingenieure Von Konrad Reif", 2007.

[12] G. Nitsche, K. Agla, „SystemC(-AMS) zur frühzeitigen Validierung & Optimierung des Systemkonzepts komplexer Smart-Sensor-System", Silicon Saxony Day, 2010.

Complex Networked Avionics Systems Design at Early Conceptual Architecture Level

Nils Fischer, Horst Salzwedel
nils.n.fischer@airbus.com , horst@mldesigner.de
Airbus Deutschland GmbH, Mission Level Design GmbH

Abstract

Current aircraft development is driven by a whole set of complex tasks. As new aircraft manufacturers introduce a stronger competition to the overall market, established aircraft manufacturers are under considerable pressure to shorten lead times for aircraft development. Today, during concept definition and specification development, a high degree of uncertainty in the system requirements arises which is propagated through various stages of the development process. Pure requirement based engineering alone is not sufficient to handle the complexity of new aircraft design challenges. Since current design approaches do not consider the impact of this uncertainty on integrated systems, the design specifications are error prone and not validated [1]. In this paper, a method is proposed which makes complex high level systems executable during the very early levels of system design. Generic model components are used to enable system designers to couple functions, architecture elements, resources and performance parameters. As a result, holistic executable specifications on so called Early Conceptual Architecture Level can be derived. A virtual test environment was created to be used for early coverage analyses and testing. The method is demonstrated for the development of a simplified aircraft communication system.

Introduction

With the introduction of networked digital electronics executing software with limited resources and information arriving at different times, it is no longer possible to separate functional behaviour of an integrated aircraft from the resources of its architecture which processes and transports all information. Each airline may specify their own architecture and configuration or even purchase several systems independently of the aircraft manufacturer. The development of these highly complex, rapidly changing systems with a variety of configurations can only be understood and developed by dedicated teams with specialized skills, experience, and knowledge about the different environments in which these systems have to be operated and maintained. The individual engineers make assumptions, many based on experience with less complex or other systems. Often, it is not possible for these engineers to understand the overall impact of their decisions on these systems or the overall system-of-systems [2].

Several attempts, such as Requirement-based Engineering (RBE) and Concurrent Engineering (CE), have been made to mitigate the development risk. Even some model-based techniques, such as SysML/UML have been introduced in order to improve the quality of the design resulting in a reduction the overall risk. However, all of these attempts have failed to deliver on their promise. The success or failure of a product is determined by the ability to provide the customer with the best solution for the intended application or mission. A functional model that does not describe a system architecture and fails to account for the overall system mission and performance will lead to an insufficient design space exploration. Such a model does not permit to optimize the overall goal of a project, i.e. the application or mission for which the product is intended [3].

Since specifications are not validated during early design levels, many projects suffer from severe problems during integration. Therefore, system designers need methods and tools to be able to couple architecture including resources (often called performance or non-functional parameters), logical functions / behaviour, application / mission, environment and if necessary usability aspects.

Model Based Aircraft Development: A Challenge for Future System Designers

Figure 1 shows the currently used design approach together with uncertainty and cost change for the development of complex systems. During system-of-systems development, mostly written specifications are being developed, typically using office applications according to DoDAF [5]. As the resulting documents are not executable, they do not permit to validate the behaviour of coupled aircraft systems on aircraft level.

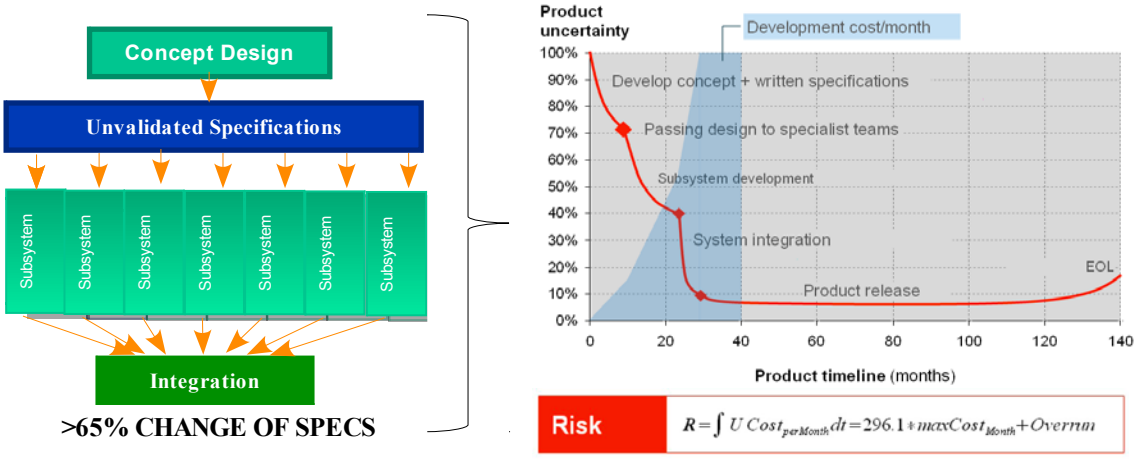


Figure 1: Specification changes, uncertainty and cost in system development

Specifications with an average uncertainty of about 65% are passed on to subsystem development. Because there is no over-all system model, pre-integration tests and global optimization cannot be performed. Subsystem developers are thus developing their subsystems without the ability to analyse the impact of their design on the coupled overall system. Software is being written without detailed knowledge of architecture, time and resource dependencies. Therefore most conflicts and uncertainties in specifications have to be resolved during system integration, causing a major financial impact. In order to minimize the development risk the product uncertainty has to be minimized as early as possible, usually during concept development when cost is low. The overall intention should be: Solve integration problems during design instead of solving design problems during integration.

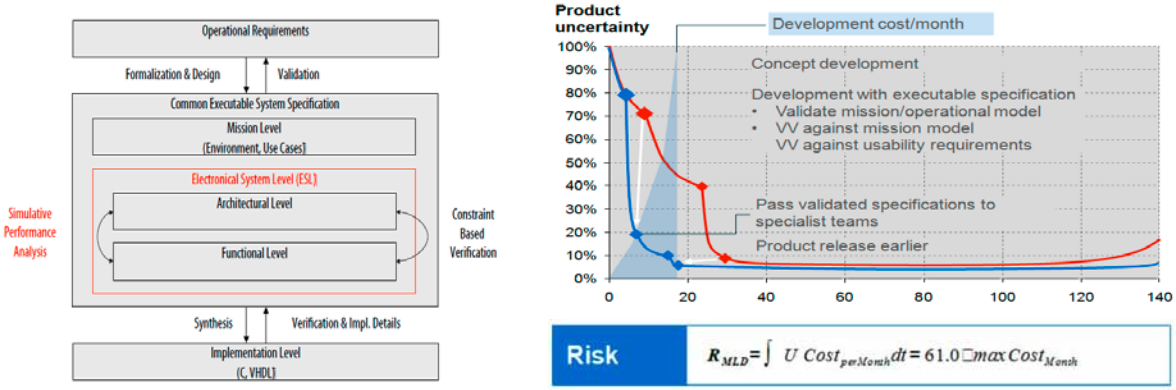


Figure 2: Holistic model design [4] and early risk reduction development process

Figure 2 shows such a development process which reduces product uncertainty much earlier within the development process (blue line). Executable specifications are based on a combined application (i.e. mission) and architecture level design flow which is illustrated on the right hand

side. To achieve such a development process we need to integrate a top-down design with bottom-up development as shown in Figure 3.

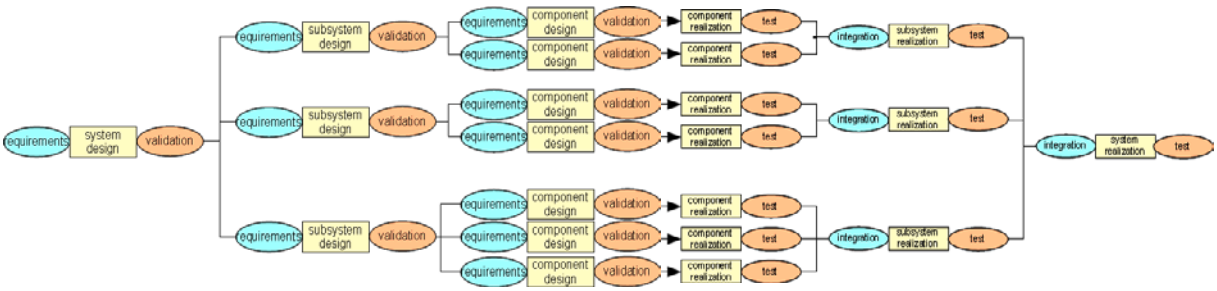


Figure 3: Top down and meet in the middle system-of-systems design approach

Aircraft Communication System

Modern civil aircraft are equipped with integrated communication systems that perform many different functions. While there are dedicated systems for voice communication between flight deck and air traffic control, as well as for navigation purposes, the system considered here facilitates communication between different aircraft systems for the purpose of operation and maintenance [3]. Figure 4 shows the extended system context of a simplified communication system equipped with three main units and some specific functions as well as other possible architecture elements and sub-elements.

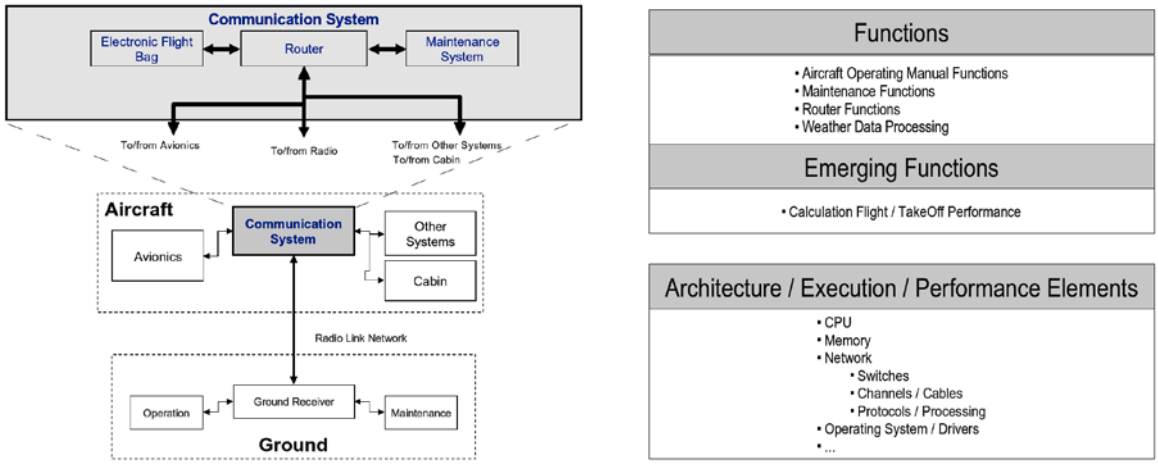


Figure 4: Upper level context and system overview of a Communication System

An Electronic Flight Bag (EFB) is a system element that supports the flight crew when performing specific tasks related to flight operation such as the Aircraft Operation Manual (AOM), calculation of take-off performance and weather data processing. Since nearly all modern aircraft use IP-based communication for non-critical systems, a router element is required to provide a certain Quality of Service for all connected systems. Civil aircraft use so called Built-in Test equipment to constantly monitor connected components and report any unusual behaviour to the Centralized Maintenance System. Most systems will also send part of the information about the state of other systems to the ground via a maintenance element. An executable specification of the civil aircraft communication system has been created with the system design tool MLDesigner to demonstrate how logical behaviour (Function), architectural and performance aspects (Elements) can be used in a combined multi-domain simulation.

Conceptual Model Design for Executable Specifications

To guarantee a coherent simulation, it was necessary to implement: (a) an overall synchronicity method that encompasses all model components, (b) a consistent data base and standardized, coherent data exchange management between elements and finally (c) to use ranges for parameter values during simulation to express uncertainty. In a conceptual model, functions are allocated to so called elements. A function block not only describes logical behaviour based on discrete event models but also contains a generic Finite State Machine (FSM) to represent several possible internal function states. For instance a function can be in normal mode, switched off or failed. Depending on its internal state, a function can adjust its behaviour during simulation, e.g. a function is deactivated or activated. To add probabilistic behaviour and model uncertainty, the state machine of a function can be linked to a failure inducing module that fails, enables or stimulates a function. Figure 5 depicts both, function and element features.

An element is a container for functions and other elements representing architectural system features like network components, CPUs, memories or entire electronic control units (ECUs) or line-replaceable units / modules (LRUs / LRMs) which can contain further elements. As with functions, elements also contain a generic FSM for internal state control which can be further detailed through hierarchical decomposition of its states. For instance a Normal state could host several sub-states like Maintenance- or Test-Mode. If, for example, an element or an entire system is in Test Mode, only specific allocated functions are available to be performed.

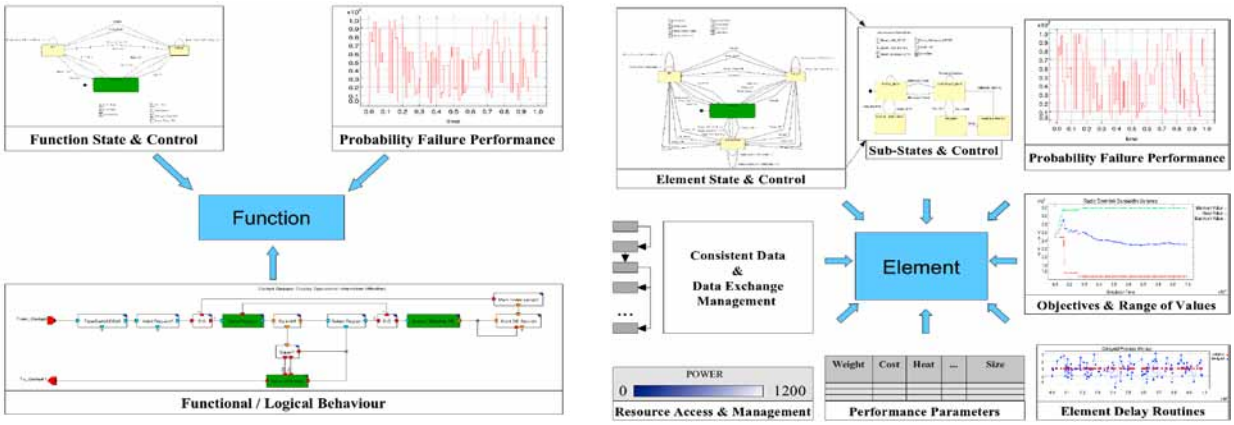


Figure 5: Features of functions (left) and elements (right) within executable conceptual models

By default but not limited to, elements have four states. Normal, Failed, Off and Unpowered. In combination with internal function states it is now possible to let an element behave normal, failed or partially failed. It can also be switched off through other elements and functions if necessary. Probabilistic control routines are used to emulate failures as they happen in real world electronic units. So called Objectives have been introduced which have to be met by the overall system or some of its elements and functions. Objectives are tuples of weighted parameter value ranges and can be used for either system validation or as parameter input values. Resources have been defined, which are directly coupled with element and function execution. If a global or local resource like power is not available or does not fulfil the demand given by an element objective, the element will stop its execution by means of its internal FSM. As a result, all of its allocated functions fail until all required resources are available. Since all elements communicate through standardized data structures, it is possible to use generic delay routines within each element. These routines delay allocated functions and data generated or processed within an element. Possible delay values can depend on available bandwidth, cable length, CPU and memory execution delays to name just a few. In early design phases, when such time aspects are only vaguely known, system designers can use more abstract element and

function delays to figure out the maximum amount of time which still enables an element or system to perform all functions while simultaneously fulfilling necessary time constraints or other objectives.

On system or extended system level a control module was implemented to manipulate system, element and function behaviour directly during simulation. A prototypical graphical user interface (GUI) was implemented to show and change the state of each model component by observation and manipulation of its generic FSM. Any function, element or sub-system failure is detected and shown. If for instance a subsystem is deactivated or fails, all of its elements and functions are automatically deactivated through GUI controlled program routines. Figure 6 depicts how manipulation of the element EFB, via the graphical user interface on the right hand side, affects several elements (ECUs, networks) as well as specific functions (left hand side, all marked in red) within other areas of the system during simulation.

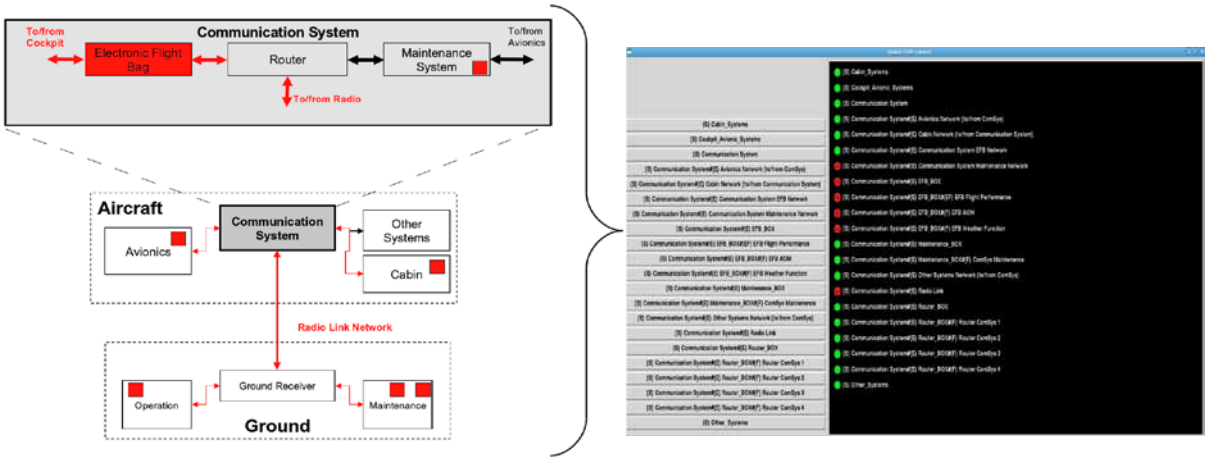


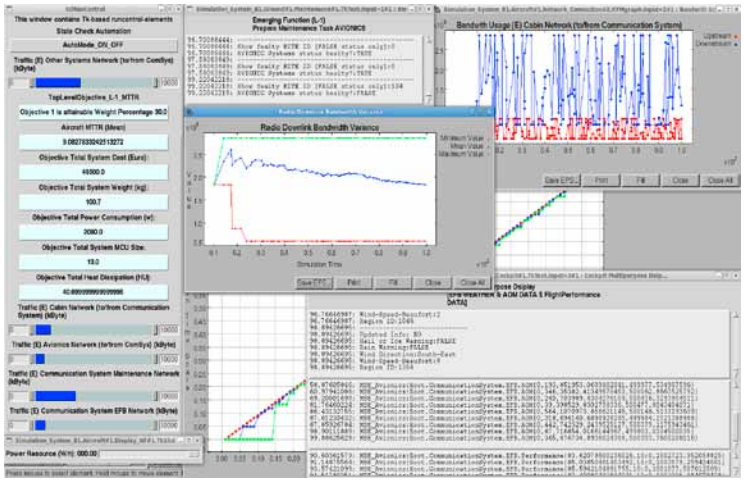
Figure 6: Global Function and Element Control in MLDesigner via Tcl/TK Interface

In addition, an auto mode is available during simulation which can automatically test different internal FSM state permutations for functions, elements and systems. To determine a possible maximum delay value for all elements of a system which also satisfies all objectives, it is also possible to auto iterate simulations. The system will gradually adapt all element delay parameters until a predefined delay precision is reached. With the abilities shown, it is now possible to analyse the direct impact of functions, elements or whole (sub-)systems on the overall system architecture and performance during system execution.

In the example system, an oversized periodic maintenance function could be isolated which was sending data continuously via the radio link network, thus generating a high and constant utilization. Because of this oversized utilization, the performance of other functions and elements of the system and its related systems was reduced to a non acceptable degree. The function specification was modified to adjust to given system architecture parameters thus resolving the conceptual problem at a very early stage of design.

Another important application for the described simulation control features are safety and reliability critical aircraft systems. By activation / deactivation of functions, elements or systems one can determine the robustness of a whole system architecture. Questions like “Will a system still work as required if one particular function or one element or a coupled system fails?” or “Will all redundancy mechanisms work?” can now be checked before system integration has even started. Several statistics, as depicted in Figure 7, are created during and after simulation showing performance results regarding dynamic system behaviour and resource utilization, non-functional and functional parameter value ranges. On the right hand side, a set of simulation

results of key objectives for two different conceptual architectures (B1 and B2) is depicted. This data is used to compare both architectures and to make a design decision, based on objective weights, for a (Pareto-) optimal solution.



Objective	Variant 1	Variant 2
T Maintenance (ms)	1,1036	1,1131
T Flight Performance (ms)	3,6616	3,6142
T AOM (ms)	5,1281	5,4920
T Weather Data (ms)	4,1515	3,7379
MTTR (h)	9,9107	9,8737
Total System Cost ($)	49500	43000
Total System Weight (kg)	101	86
Total Power Consumption (W)	2090	1810
Total System Size (MCU)	13	10
Total Heat Dissipation (units)	40,6999	31,8999
All Objectives Attainable	Yes	Yes

Figure 7: Several simulation results in MLDesigner and different conceptual architecture values

Summary

An executable aircraft communication system model was developed using the design tool MLDesigner. The model describes the system and its context (aircraft, ground entities) as well as coupled functional, architecture and performance components. A virtual test environment was created to survey and manipulate the operational status of each model component manually. Since MLDesigner is able to perform Monte-Carlo-Simulations it is also possible to simulate with parameter value ranges and therefore including model uncertainty. FSM based control structures embedded in functions, elements and sub-systems are used to simulate different internal states which affect the overall system behaviour and performance. These control structures also allow automated, program controlled coverage analyses for all parts of a system model. An execution of several test scenarios or pre-defined coverage tests is possible via configuration files. With this, system designers can perform early concept validation and simulate different architecture alternatives to optimize the overall system application before implementation and integration have started.

References

[1] John Hines, We Don't Do Design Correctly!, Keynote speech at IEEE 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS 2001, Cincinati, Ohio, Aug. 15-18, 2001.

[2] H. Salzwedel, N. Fischer, and T. Baumann, Aircraft level optimization of avionics architectures, in 27th Digital Avionics Conference – DASC 2008, ser. AIAA-IEEE DASC. St. Paul, Minnesota, United States: Digital Avionics Systems Conference, October 2008.

[3] S. Marwedel, N. Fischer, H. Salzwedel, Improving the design quality of complex networked systems using a model–based approach, in 3rd International Conference on Model–based Systems Engineering, International Council on Systems Engineering, Fairfax, Virginia, USA (2010).

[4] 50th Anniversary Symposium & Celebration, 2008, Stanford University, Model Based Design, Analysis and Validation of STEP using MLDesigner and SatLab

[5] DOD Architecture Framework (DoDAF), V1.5, Vols I, II, and III, Version 1.5 ed. Department of Defense, April 2007.

Engineering fehlerfreier Steuerprogramme: Grafischer Programmentwurf, Modelica-basierte Verifikation, EC61131-Code-Generierung, Software in the Loop

Stephan Seidel, Ulrich Donath
{Stephan.Seidel | Ulrich.Donath}@eas.iis.fraunhofer.de
Fraunhofer Institut für Integrierte Schaltungen, Institutsteil Entwurfsautomatisierung

Kurzfassung

Gegenwärtig beginnt die Formalisierung der Steuerprogrammentwicklung zumeist mit der Programmierung des Steuerprogramms selbst. Dabei führen Inkonsistenzen oder Fehldeutungen der Aufgabenstellungen zu Fehlern, die in aufwendigen Software-in-the-Loop Tests aufzuklären sind. Die Verfasser zeigen, dass die Formalisierung der Programmentwicklung bereits in der Entwurfsetappe beginnen kann. Sie nutzen dazu grafische Editoren. Die erzeugte Systembeschreibung, inklusive Steuerprogramm, wird in Modelica-Code konvertiert. Auf Modelica-Ebene erfolgt die Verifikation des Systems, indem das aktuelle Verhalten mit den Zielvorgaben verglichen wird. Nach diesem Schritt wird aus der grafischen Beschreibung des Steuerprogramms der Zielcode generiert, in diesem Fall ist dies Strukturierter Text nach IEC 61131. In Software-in-the-Loop Tests wird die Echtzeit-Tauglichkeit des Codes validiert.

Einleitung

In der konventionellen Steuerprogramm-Entwicklung entsteht häufig die Notwendigkeit, in umfangreichen Software-in-the-Loop Tests Ursachen für die fehlerhafte Wechselwirkung einzelner Komponenten aufzuklären. Als Quellen lassen sich letztendlich Inkonsistenzen in der Definition der Aufgabenstellung finden [1]. Diese hätten in einem intensiven papier-dominierten Review-Prozess identifiziert werden können. Eine Alternative ist das „Vorziehen“ der Formalisierung von der Codierung auf den Entwurf. Die Formalisierung des Entwurfs besteht in der Konstruktion von Modellen, die sowohl die strukturelle Gliederung des Systems als auch die Funktionsdefinitionen der einzelnen Komponenten erfassen. Sind die entwickelten Modelle ausführbar, ermöglichen sie in Simulationen sowohl den funktionalen Test der Einzelkomponenten als auch den Test ihres Zusammenwirkens im Gesamtsystem. Die Testergebnisse werden mit den Anforderungen in der Definition der Aufgabenstellung verglichen. Bei Differenzen erfolgt iterativ die Korrektur der Modelle oder es wird die Aufgabenstellung qualifiziert und der Entwurfsprozess wiederholt. Erfüllt die Funktionalität des Gesamtmodells die gestellten Anforderungen, so erfolgt die automatisierte Generierung des Zielcodes der Steuerkomponenten. Zur abschließenden Verifikation werden Software-in-the-Loop Tests dieser Komponenten ausgeführt. Dabei wird das reale Steuerungsprogramm mit Modellkomponenten verkoppelt und durch deren Simulation mit Testgrößen beaufschlagt.

Ausgangspunkt der im Folgenden vorgestellten Steuerprogramm-Entwicklung ist ein Modell der Maschine [2], das in die Partitionen Maschinenmodell, Steuerungsmodell und Operatormodell gegliedert wird.

Das Maschinenmodell wird aus den Modellen der einzusetzenden elektrischen, mechanischen und/oder hydraulischen Maschinenkomponenten zusammengesetzt. Diese Modelle liegen in Modelica-Codierung vor und sind Bestandteil einer Bibliothek. Sie werden grafisch in einer Modellansicht instanziiert und verkoppelt. Das Maschinenmodell fungiert in seiner Gesamtheit als das physikalische Umgebungsmodell für die Steuerung.

Das Steuerungsmodell beschreibt die Operationen der Steuerung für einen geforderten Maschinenablauf. Diese Operationen sind im einfachen Fall lineare Anweisungsfolgen, im komplexen Fall Algorithmen, die einen Zustandsautomaten abbilden. Für die grafische

Beschreibung der Zustandsautomaten wird ein UML-Profil [3] genutzt. Zur Bildung des ausführbaren Modells werden die Zustandsdiagramme in Modelica-Code übersetzt. An die Zustände oder/und Zustandsübergänge sind bereits im Entwurf der Diagramme Aktionen in Modelica-Code geknüpft worden.

Das Operatormodell fasst die Bedienoperationen der Maschine zusammen. Das sind in der Regel lineare Anweisungsfolgen, die vom Produktionsplan abgeleitet werden. Sie werden in der Zielsetzung der Gesamtsystem-Simulation in Modelica-Code formuliert.

Die einzelnen Schritte in der Entwicklung des Steuerprogramms

- Grafischer Programmentwurf
- Test und Verifikation auf Modelica-Ebene
- IEC61131-Code-Generierung
- Software-in-the-Loop Test

werden am Beispiel der Einschiebeeinrichtung einer Verpackungsmaschine demonstriert.

Grafischer Programmentwurf

Die Verpackungsmaschine realisiert die Endverpackung von Beutelware in Kartons. In der Einschiebeeinrichtung (Abb. 1) werden die Beutel vor dem Einschieber auf einer Zunge positioniert. Der Karton wird durch ein Klemmband direkt vor dem Einschieber in die Befüllposition gebracht. Ist diese Position erreicht, wird die Zunge ausgefahren und der Beutel auf der Zunge durch den Einschieber in den Karton geschoben. Danach sind Zunge und Einschieber zurückzufahren und der Karton ist in die neue Füllposition zu bringen.

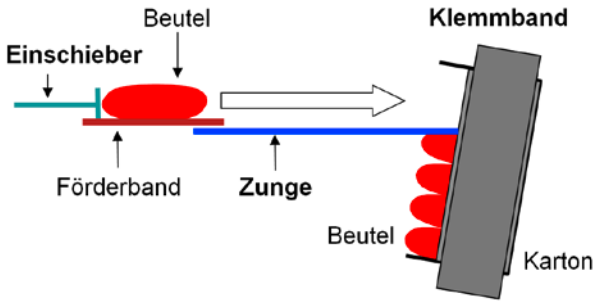


Abb. 1: Schema der Einschiebeeinrichtung

Zunächst werden nacheinander die Modelle *Zunge (Tongue TO)*, *Einschieber (Pusher PU)* und *Klemmband (Belt BE)* entworfen, wobei das jeweilige Modell der Gerätekomponente vom Modell der Steuerung getrennt wird.

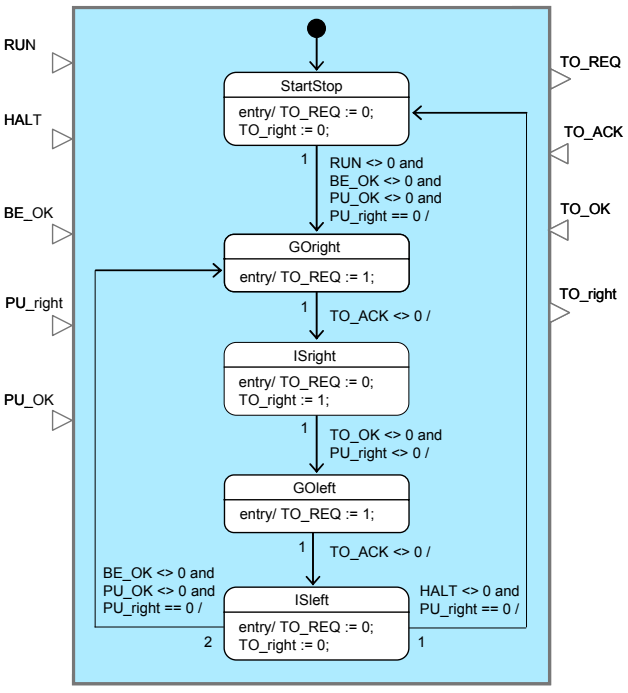


Abb. 2: Zustandsdiagramm der Zungensteuerung

Die Gerätekomponentenmodelle von Zunge und Einschieber besitzen einen identischen inneren Aufbau (Antrieb, Masse, Reibung) und ein identisches Interface. Das Interface der Steuerung wird spiegelbildlich zum Interface der Gerätekomponente definiert. Die Zungen- und die Einschiebersteuerung werden als Zustandsautomaten realisiert (Abb. 2). In den Zuständen wird das Steuersignal für den Antrieb gesetzt oder gelöscht. Der Antrieb fährt daraufhin alternierend die Zunge bzw. den Einschieber aus und ein. Die Transitionen zwischen den Zuständen werden durch die Rückmeldungen des Antriebs und die Operatorkommandos RUN und HALT getriggert.

Das Modell der Gerätekomponente Klemmband beschreibt die Auf- und Abwärts-Bewegung von Klemmband und Karton. Neben Antrieb, Masse und Reibung wird in dem Modell die Gewichtskraft erfasst, die mit der Füllung des Kartons zunimmt. Die Steueroperationen werden ebenfalls durch einen Zustandsautomaten realisiert. Dieser untergliedert die Bewegung des Klemmbands in einen Anfahrvorgang und in einen zyklischen Positioniervorgang zum Befüllen des Kartons. In den Zuständen werden die Klemmband-Geschwindigkeiten SPEED angewiesen und die Zielpositionen berechnet. Der Zyklus des Positionierens wird bei vorgegebenem Füllstand des Kartons unterbrochen.

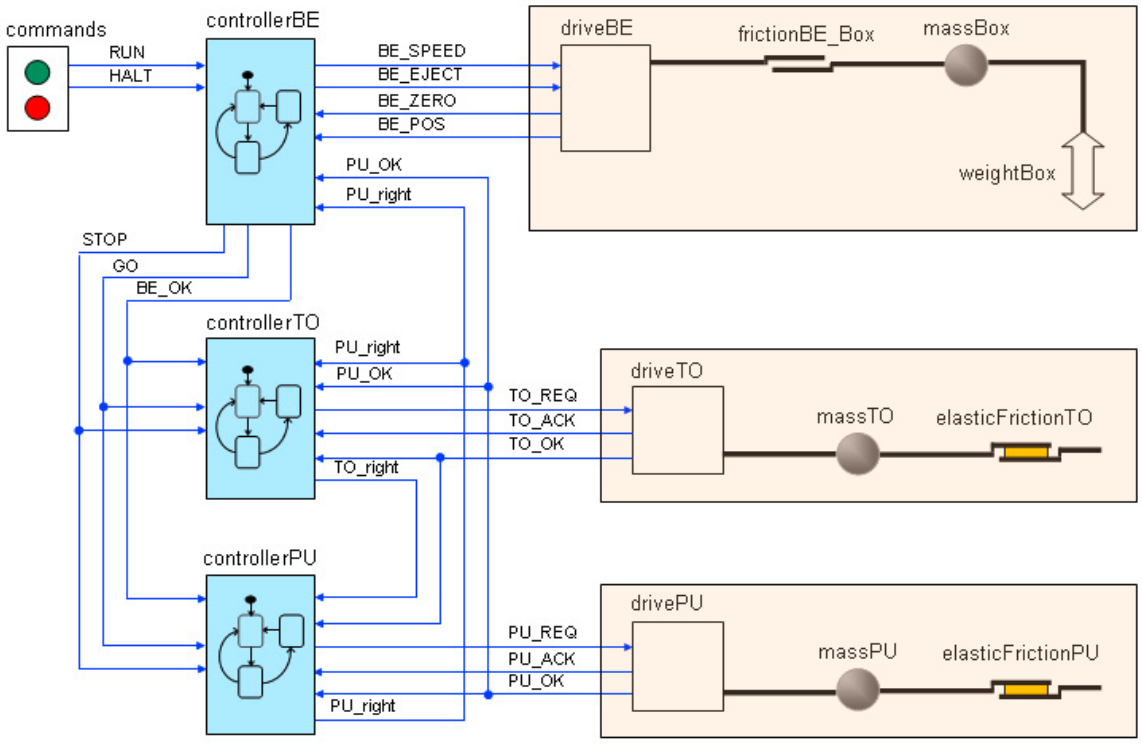


Abb. 3: Gesamtmodell der Einschiebeeinrichtung

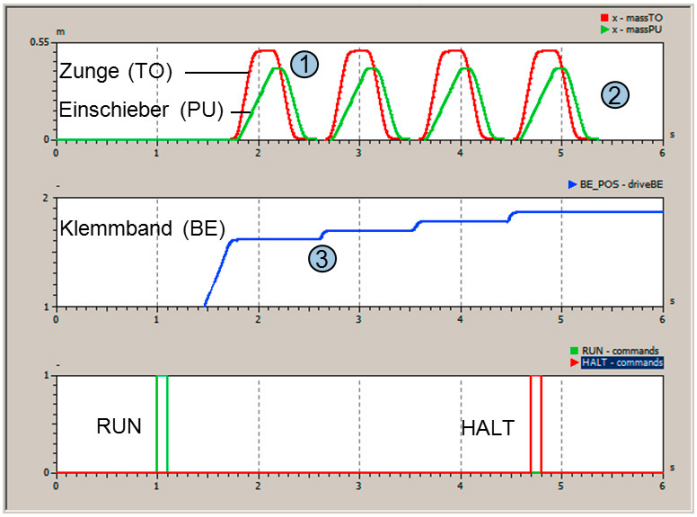
Nach dem Entwurf der einzelnen Modelle Zunge, Einschieber und Klemmband werden diese gemeinsam in einer Modellansicht instanziiert und verkoppelt (Abb. 3). Der Klemmband-Controller übernimmt im Gesamtmodell eine Master-Funktion, indem er die STOP- und GO-Signale sowie nach abgeschlossener Positionierung das OK-Signal für die untergeordneten Controller von Zunge und Einschieber bildet.

Test und Verifikation auf Modelica-Ebene

Die Modelle der Gerätekomponenten werden mit physikalischen Elementen und Signalgliedern gebildet, die in Bibliotheken enthalten sind und bereits eine Repräsentation in Modelica besitzen. Die Diagramme der Steuerkomponenten werden automatisiert jeweils in ein (Modelica-) *model* mit entsprechenden Typ-, Variablen-, Parameter- und Signal-Deklarationen und einen (Modelica-) *algorithm* übersetzt. Das Modelica-Gesamtmodell entsteht aus der Verknüpfung aller Modellkomponenten gemäß Abbildung 3.

Zur Verifikation der Steuerung werden zunächst die Teilmodelle Zunge und Einschieber bezüglich der geforderten Funktionen betrachtet. Abbildung 4 zeigt die Weg-Zeit-Diagramme von Zunge und Einschieber in Relation zu den Befehlen RUN und HALT.

Als Fazit der Simulation zeigen sich folgende Ergebnisse. Die Zunge fährt wie gefordert vor dem Einschieber aus und ein und HALT bewirkt einen Stopp nachdem beide Komponenten eingefahren sind und sich wieder in Nullposition befinden.



1. Die Zunge wird aus dem Karton zurückgefahren, bevor der Einschieber zurückfährt
2. Zunge und Einschieber fahren ein und bleiben in ihrer Ausgangsposition nachdem das HALT Signal empfangen wurde
3. Das Klemmband bewegt sich nur, wenn Zunge und Einschieber in der Ausgangsposition sind

Abb. 4: Simulationsergebnis der Einschiebeeinrichtung

Die Funktionen der Klemmbandsteuerung werden zunächst ebenfalls einzeln im Teilmodell getestet und anschliessend im Gesamtmodell der Einschiebeeinrichtung nach Abbildung 3. Das Simulationsergebnis in Abbildung 4 zeigt die Klemmbandpositionierung in Relation zur Zungen- und Einschieber-Bewegung. Bestätigt wird die Forderung, dass sich das Klemmband nur bewegt, wenn Zunge und Einschieber eingefahren sind (Nullposition). HALT, RUN und die vollständige Befüllung des Kartons führen zu einem koordinierten Stopp und Restart der Abläufe.

IEC61131-Code-Generierung

Die (Modelica-) *algorithms* der Zustandsautomaten umfassen i. A. folgende Blöcke:

- Event Generation:
  when-statements mit Relationen von Signalen oder Zeiten und Toggeln von Flags
- Entry Activities:
  when-elsewhen-statements mit Check von diskreten Zustandsvariablen und *signal assignments*
- Do Activities:
  if-elseif-statements mit Check von diskreten Zustandsvariablen und *simple signal assignments*
- Transitions:
  if-elseif-statements mit Check von diskreten Zustandsvariablen und Boolescher Übergangsbedingungen sowie Wertzuweisungen an die Zustandsvariablen und *signal assignments*.

Die Blöcke *Entry Activities*, *Do Activities* und *Transitions* werden entsprechend der hierarchischen Gliederung der Zustandsdiagramme mehrfach gebildet.

Die Semantik des Modelica-Codes bildet das Verhaltensnormal für den zu implementierenden Zielcode. Als Zielcode wird exemplarisch Strukturierter Text (ST) nach IEC 61131 gewählt, welcher die Programmierung von speicherprogrammierbaren Steuerungen (SPS) in einer Hochsprache ähnlich Pascal ermöglicht. In Analogie zum Modelica-Code des Zustandsautomaten wird der Code in ST in die Funktionen *FC Event Generation, FC Entry Activities, FC Do Activities* und *FC Transitions* unterteilt (Abb. 5). Die Organisation der SPS erfordert darüber hinaus die Trennung von Deklarationen und auszuführenden Funktionen.

E/A- und globale Variablen sowie die Namen der verwendeten Funktionen und Funktionsbausteine werden in einer Symboltabelle definiert. Diese Tabelle wird im Rahmen der

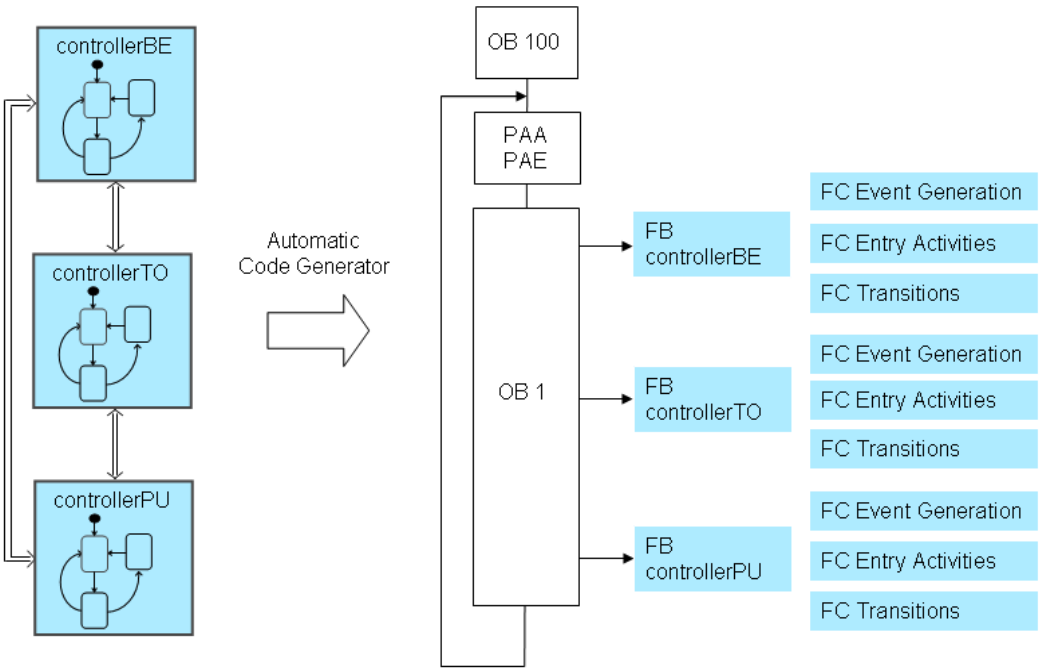


Abb. 5: Struktur des SPS-Codes

Codegenerierung [4] automatisch erzeugt. Zusätzlich wird ein Datenbaustein generiert, welcher die internen Variablen des Zustandsautomaten bereitstellt.

FC Event Generation (EG) führt die Erkennung von Signal-Events aus. Da ein *when*-statement in ST nicht existiert, wird eine Flankenerkennung unter Verwendung einer zusätzlichen Merker-Variablen in einem *if*-statement realisiert.

In *FC Entry Activities (EA)* werden die Entry-Aktivitäten der Zustände codiert. Ein *case*-statement wählt zunächst den aktiven Zustand aus. Selektiert durch einen Entry-Merker innerhalb eines *if*-statements erfolgt danach die einmalige Ausführung der Entry-Aktivität. Die Do-Aktivitäten in *FC Do Activities (DA)* werden analog implementiert, enthalten jedoch kein *if*-statement mit Entry-Merker. Damit werden die Do-Aktivitäten im Gegensatz zu Entry-Aktivitäten in jedem SPS-Zyklus ausgeführt.

FC Transition (TR) enthält die Codierung der Transition-Aktivitäten. Der aktive Zustand wird in einem *case*-statement erkannt. In jedem Zweig dieses Statements ist in einem *if*-statement mit korrespondierendem Transition-Trigger die Transition-Aktivität notiert. Bei aktivem Zustand und erfülltem Transition-Trigger wird die entsprechende Transition-Aktivität ausgeführt und die Zustandsvariable auf den Folgezustand gesetzt.

Alle oben beschriebenen Funktionen werden jeweils vom Funktionsbausteinen *FB controllerXY* in der Sequenz *EG, EA, DA, TR* ausgeführt. Zusätzlich erfolgt in diesem Funktionsbaustein die Timer-Organisation des Zustandsautomaten. Die *FB controllerXY* werden im Organisationsbaustein *OB1* in der in Abbildung 5 dargestellten Reihenfolge aufgerufen. *OB1* wird innerhalb des SPS-Zyklus nach der Aktualisierung der E/A-Variablen ausgeführt.

Bei Abbildung eines Zustandsdiagramms in Strukturierten Text ist zu organisieren, dass aufgerufene Bausteine vor den aufrufenden Bausteinen im Quellcode stehen [5]. Die Reihenfolge der Bausteine in ST muss somit wie folgt lauten: (1) Datenbaustein mit internen Variablen, (2) Funktionen *Event Generation*, *Entry Activities*, *Do Activities*, *Transitions,* (3) Funktionsbaustein *controllerXY,* (4) Organisationsbaustein *OB1,* (5) Organisationsbaustein *OB100.*

Besteht die Steuerung des Automatisierungssystems aus mehreren Zustandsautomaten, so erzeugt der Codegenerator zunächst für jeden Zustandsautmaten separate Dateien für Symboltabelle und ST-Programmcode. Ein AddOn-Tool liest diese Daten und erzeugt daraus eine

gemeinsame Symboltabelle und eine gemeinsames ST-Programm, welches alle Bausteine für jeden Zustandsautomaten enthält und zusätzlich einen gemeinsamen zyklischen Baustein *OB1* und Initialisierungsbaustein *OB100*.

Software-in-the-Loop Test

Nachdem das SPS-Programm in Strukturiertem Text erzeugt wurde, wird eine Co-Simulation von SPS-Programm und Maschinenmodell durchgeführt. Dazu wird eine Simulations-SPS verwendet, wie sie beispielweise unter dem Namen PLCSIM im Umfang des SPS-Programmierwerkzeuges Step7 von Siemens enthalten ist. PLCSIM läuft als separater Prozess auf einem Windows-Rechner und ermöglicht die Ausführung eines SPS-Programms äquivalent zu einer Hardware-SPS.

Zur Durchführung einer Co-Simulation ist es notwendig, den Ein- und Ausgängen der Simulations-SPS die korrespondierenden Signale des Maschinenmodells zuzuordnen und diese während des Simulationslaufs zyklisch zu aktualisieren. Dazu werden im Modelica-Modell alle Zustandsdiagramme des Steuerungsmodells aus dem Modell entfernt und durch ein Koppelelement ersetzt, welches über die gleichen E/A-Signale wie die Zustandsdiagramme verfügt. Dieses Koppelelement ist mit Schnittstellen zu beiden Simulatoren ausgestattet. Zum einen ist das für den Modelica-Simulator eine TCP-Socket Schnittstelle, zum anderen für die Simulations-SPS ein COM-Objekt. Da die Modelica-Simulation in virtueller Zeit abläuft, die SPS jedoch in Echtzeit voranschreitet, ist eine Synchronisation der Simulations-SPS notwendig. Dazu werden die im SPS-Programm verwendeten Timer durch das Koppelprogramm zu Beginn jedes SPS-Zyklus aktualisiert. Der Fortschritt der virtuellen Modelica-Simulationszeit bestimmt damit den Ablauf der SPS-Timer.

Zusammenfassung

In diesem Artikel wurde der modellbasierte Entwurf und die Verifikation von Automatisierungssystemen vorgestellt. Am Beispiel einer industriellen Verpackungsmaschine erfolgt der Entwurf der Steuerung mit einem grafischen Editor für Zustandsdiagramme, aus welchen zur Simulation des Systems auf Modellebene Modelica-Code generiert wird. Mit diesem wird eine Gesamtsystemsimulation in einem Modelica-Simulator ausgeführt und die Bewertung der generellen Funktion sowie der Leistungsparameter des Systems vorgenommen. Im Anschluss daran wird aus den Zustandsdiagrammen Code als Strukturierter Text nach IEC 61131 für speicherprogrammierbare Steuerungen generiert, welcher im Rahmen einer Software-in-the-Loop Simulation ebenfalls validiert wird. Es entfällt die zeitaufwändige und fehlerträchtige Prozedur des manuellen Codierens des Steuerungsprogramms. Der gesamte Entwurfprozess gestaltet sich durch den modellbasierten Entwurf, Simulation und Verifikation wesentlich effizienter als derzeit in der industriellen Praxis angewandte Prozeduren.

Literatur

[1] Schwabe, S.: Modellbasierter Systems-Engineering-Prozess. Economic Engineering 3/2009, S. 58-59

[2] Haufe, J., Donath, U., Lantzsch, G.: Modellbasierter Entwurf von Steuerungen in der Automatisierungstechnik. Dresdner Arbeitstagung Schaltungs- und Systementwurf (DASS), Dresden, 2009

[3] Donath, U.; Haufe, J.; Blochwitz, T.; Neidhold, T.: A new approach for modeling and verification of discrete control components within a Modelica environment. Proceedings of the 6th Modelica Conference, Bielefeld, March 2008, p. 269-276

[4] Lindner L.: Rapid Control Prototyping by Transformation of Hierarchical State Machine Control Models into IEC 61131 PLC Code. Diploma thesis, TU Dresden, 2009

[5] Siemens AG. S7-SCL V5.3 for S7-300/400 Manual, 2005

Modellbasierter Entwurf von Echtzeit-Applikationen für 8-, 16- und 32-bit-Mikrocontroller

Holger Priwitzer, Olaf Enge-Rosenblatt, Peter Schneider
{holger.priwitzer, olaf.enge, peter.schneider}@eas.iis.fraunhofer.de
Fraunhofer-Institut für Integrierte Schaltungen Institutssteil Entwurfsautomatisierung (IIS/EAS),
Zeunerstraße 38, 01069 Dresden

Kurzfassung

Im vorliegenden Beitrag wird der Einsatz modellbasierter Entwurfstechniken für die Entwicklung von Echtzeit-Applikationen diskutiert. Im Fokus der Betrachtungen liegt ein verallgemeinerter Ansatz zur Überführung von Modellen aus Matlab/Simulink auf unterschiedliche Mikrocontrollerarchitekturen. In diesem Zusammenhang wird ein am IIS/EAS entwickelter Embedded Target sowie dessen auf Mikrocontrollerapplikationen zugeschnittenes Simulink-Blockset vorgestellt. Anhand von Beispielapplikationen für 8-bit-Mikrocontroller werden Vor- und Nachteile des modellbasierten Ansatzes sowie der konkreten Umsetzung diskutiert.

Einleitung

Modellgestützte Entwurftechniken werden heute in vielen Bereichen von Forschung und Industrie eingesetzt, um der stetig wachsenden Komplexität der Systeme und den damit verbundenen Risiken einer Fehlentwicklung entgegenzuwirken. Dabei zeichnen sich heutige Entwurfswerkzeuge insbesondere durch ihre Durchgängigkeit während des kompletten Entwurfsprozesses aus. Dadurch ist der Übergang von der abstrakten Systembeschreibung in Form von Modellen bis hin zur Implementierung der serienreifen Software nahezu vollständig automatisierbar. Aufgrund des vermeintlich höheren Aufwands der modellgestützten Systembeschreibung und den damit verbundenen Zusatzkosten beschränken sich die Anwendungsgebiete jedoch häufig auf Produktentwicklungen mit hohen Auftragsvolumina, Anwendungen mit sicherheitskritischen Aspekten [1] oder die Entwicklung von Spitzentechnologien [2].

Mit dem Ziel, die Möglichkeiten aktueller Entwurfswerkzeuge in einem breiteren Anwendungsspektrum nutzen zu können, soll im vorliegenden Beitrag ein auf Matlab/Simulink basierender Ansatz zur Entwicklung von Mikrocontrolleranwendungen vorgestellt werden.

Codegenerierung mittels Simulink

Die Simulink-Entwurfsumgebung bietet neben der modellgestützten Systembeschreibung und Simulation auch die Möglichkeit der plattformspezifischen Codegenerierung (C/C++). Auf Basis der Werkzeuge Real-Time Workshop (RTW) und Real-Time Workshop Embedded Coder (EC) lassen sich (Teil-) Modelle inklusive der zur Berechnung von diskreten/kontinuierlichen Zustandsvariablen notwendigen Lösungsverfahren per Knopfdruck exportieren. Für einfache bis mittlere Anwendungen (Eingrößenregelungen/-steuerungen, einfache Zustandsregler) ist der generierte C-Code dabei effizient genug, um diese selbst auf kleinen 8-bit-Mikrocontrollern zu implementieren. Durch zusätzliche Konfigurationsoptionen können die Modelle bezüglich ihrer späteren Implementierung optimiert werden (z.B. Fixed-Point-Datentypen) und die daraus resultierenden Effekte (z.B. Wertebereichsüberschreitung) bereits während der Simulation überprüft werden.

Die Anpassung der Modelle an eine spezielle Zielhardware erfolgt typischer Weise über plattformspezifische Targets, welche deren funktionale Komponenten (z.B. GPIO, ADC, …) als Bibliothek innerhalb Simulink bereitstellen. Diese Targets stellen außerdem notwendige Informationen zur jeweils verwendeten MCU (8-, 16-, 32-bit, little/big endianness, …) bereit und übernehmen die Einbindung der plattformspezifischen Werkzeuge (Compiler, Linker, …). Die targetspezifische Konfiguration des Codegenerators erfolgt über ein System Target File (STF). Dieses basiert auf einer Metasprache – der Target Language – und dient dem Target Language Compiler (TLC) als Einstiegspunkt für die Überführung der Modellbeschreibung in einen ausführbaren Programmcode (typischer Weise C oder C++). Durch die Festlegung globaler TLC-Variablen innerhalb des STF wie der Target-Klasse (GRT – Generic Real-Time Target

oder ERT – Embedded Real-Time Target) oder des Codeformats (Real-Time, Embedded Real-Time, …) sowie durch die Einbindung vordefinierter oder targetspezifischer TLC-Bibliotheken wird die konkrete Implementierung der Modelle für das jeweilige Zielsystem gesteuert.

Verglichen mit der Vielzahl am Markt erhältlicher Mikrocontroller werden derzeit jedoch nur wenige Targets mit Unterstützung für ausgewählte Derivate kommerziell angeboten (The Mathworks, einige Drittanbieter). Der Aufwand für die Erstellung eines benutzerdefinierten Targets, wie es in [3] beschrieben wird, stellt für den Nutzer in der Regel einen unverhältnismäßig hohen Aufwand dar. Um diesem Problem entgegenzutreten wurde am Fraunhofer-Institut IIS/EAS ein Embedded Target („EAS-Target“) entwickelt, der aufgrund seines verallgemeinerten Ansatzes für die Erweiterung und Verwendung unterschiedlicher Mikrocontrollertypen ausgelegt ist.

EAS-Target – Simulink-Blockset für die Mikrocontrollerprogrammierung

Der EAS-Target ist ein für Matlab/Simulink entwickelter Embedded Target, welcher sich vollständig in die Entwurfsumgebung integriert (vgl. Abb. 1). Vor dem Hintergrund der Verwendung unterschiedlicher Mikrocontroller sieht das Konzept des EAS-Target die Abstraktion zwischen den im Modell verwendeten Treiberblöcken und der konkreten Implementierung im Mikrocontroller vor. Aufgrund dieser Abstraktion stehen dem Nutzer bei der Parametrisierung zwar nicht alle Features der jeweiligen Ressource zur Verfügung. Jedoch ergeben sich dadurch auch weitreichende Vorteile. Diese sind einerseits die leichtere Portierbarkeit von Modellen für unterschiedliche Controller, deren Auswahl man an einer zentralen Stelle im Modell – dem Master-Block – vornimmt. Andererseits wird der Aufwand für die Unterstützung weiterer Mikrocontrollertypen auf ein Minimum reduziert, da sich dieser im Wesentlichen auf die konkrete Implementierung der Hardwaretreiber und nicht auf deren Integration in Simulink konzentriert.

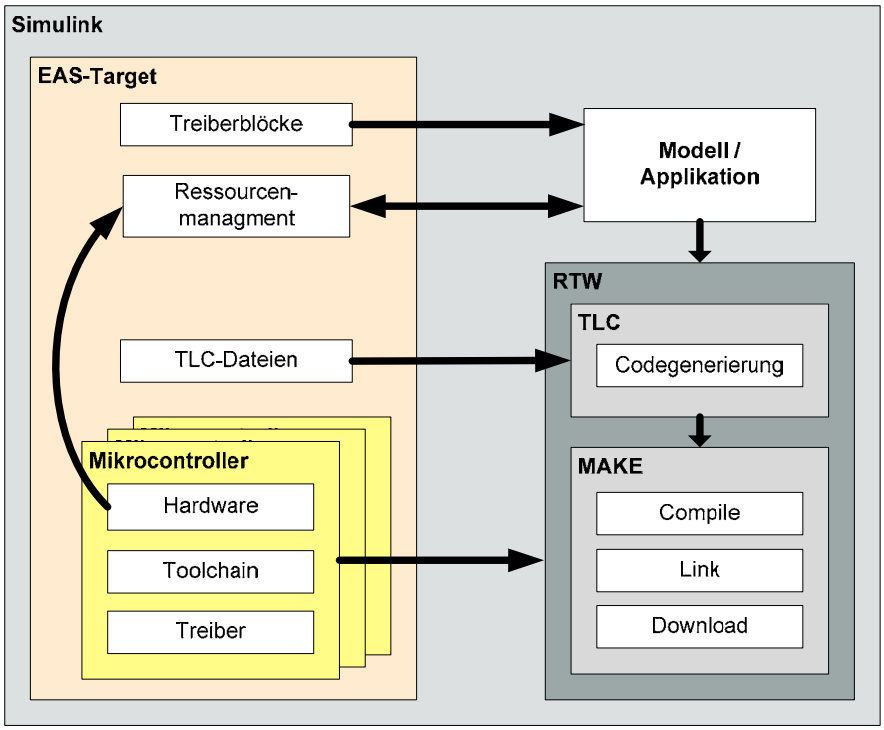


Abb. 1 Integration des EAS-Target in die Simulink Entwurfsumgebung

Einen entscheidenden Bestandteil des EAS-Target bildet die zugehörige Simulink-Bibliothek mit Treiberblöcken für typische Mikrocontrollerperipherie (vgl. Abb. 2). Jeder dieser Blöcke ist als S-Funktion [4] mit individueller Blockmaske implementiert und erlaubt somit die Vereinbarung und Parametrisierung der jeweiligen Hardwareressource im Modell. Jedem Treiberblock zugehörig existieren TLC-Dateien, welche die notwendigen Funktionsaufrufe zur Initialisierung, Terminierung sowie für den schreibenden und/oder lesenden Zugriff auf die jeweilige Hardwareressource in den Modellcode generieren. Durch die Blöcke des EAS-Target werden neben einfachen digitalen IO’s, Zählern, ADC’s und PWM’s auch Busschnittstellen wie UART, I^2C und SPI mit einer hohen Flexibilität bezüglich der Protokolleinstellungen unterstützt. Die Verwendung eines internen EEPROM-Speichers für die permanente Speicherung von Konfigurations- oder Kalibrierungsdaten oder der Zugriff auf externen Speicher wird

durch entsprechende Schreib-/Leseblöcke unterstützt. Die bereitgestellten Blöcke zur Verwendung von Timern und externen Interrupt-Eingängen erlauben außerdem die asynchrone Berechnung ausgewählter Modellbestandteile.

Eine besondere Stellung innerhalb der EAS-Target-Blockbibliothek nimmt der Master-Block ein. Zum einen werden über die Maske dieses Blocks der zu verwendende Mikrocontrollers sowie globale Einstellungen wie der Systemtakt und treiberspezifische Parameter (z.B. Baudrate einer UART) vorgenommen. Zum anderen wird durch den Master-Block, abhängig vom gewählten Controller, die Verwendung der zur Verfügung stehenden Ressourcen innerhalb des Modells überwacht, so dass Konflikte aufgrund unzulässiger (Mehrfach-) Verwendung von Hardwareschnittstellen erkannt werden.

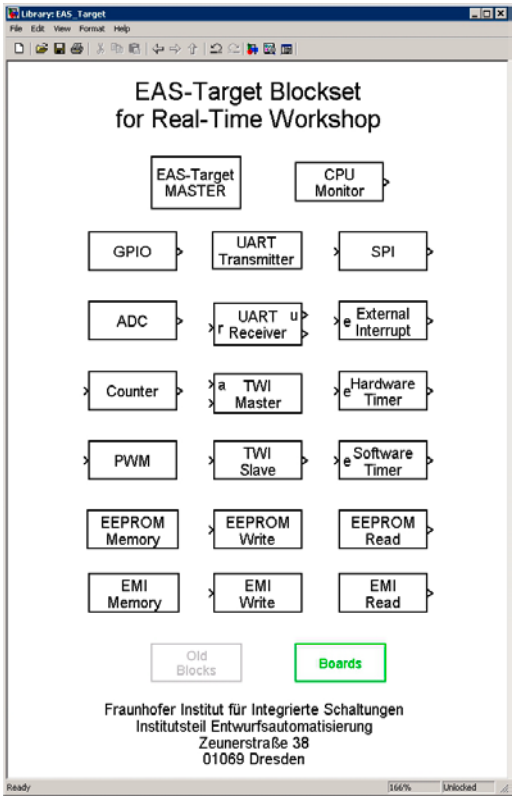


Abb. 2 Blockbibliothek des EAS-Target

Im Hinblick auf die Realisierung einer Mikrocontrollerapplikation ist es in der Regel notwendig, extern an den Controller angeschlossene digitale und/oder analoge Komponenten anzusteuern. Dies erfordert insbesondere bei komplexen, über digitale (Bus-) Schnittstellen angebundenen Funktionsbaugruppen (z.B. mehrkanaliger ADC mit programmierbarer Verstärkung oder Grafikdisplay mit eigenem Controller) das gezielte Schreiben/Lesen von Registern auf der externen Komponente, die Umsetzung eines proprietären Protokolls oder die exakte zeitliche Reaktion auf externe Interrupts. Die Flexibilität der durch den EAS-Target bereitgestellten Treiberblöcke reicht hierbei in vielen Fällen aus, um die applikationsspezifische Ansteuerung von extern an den Mikrocontroller angeschlossenen Komponenten direkt in Simulink zu beschreiben. Mit Hilfe von Simulink-Standard-Bibliotheken oder Erweiterungen wie der Stateflow-Toolbox zur Implementierung von Zustandsautomaten können komplexe Funktionen realisiert werden, deren funktionales und zeitliches Verhalten unter Verwendung von geeigneten Quellen und Senken bereits in der Simulation getestet werden können. Die Kapselung dieser funktional eng zueinander gehörenden Modellbestandteile in parametrisierbare Subsysteme erlaubt außerdem deren Wiederverwendung an unterschiedlichen Stellen im Modell sowie deren Bereitstellung als benutzerdefinierte Blockbibliothek. Die Abstraktion der Funktionsbeschreibung in Simulink von der konkreten Hardware-Implementierung bleibt hierbei vollständig erhalten, so dass man die einmal umgesetzte Funktionalität auch in zukünftigen Applikationen mit möglicherweise anderen durch den EAS-Target unterstützten Mikrocontrollern direkt wiederverwenden kann.

Beispiel 1: Programmierbare Präzisionsstromquelle

Im Rahmen der Inbetriebnahme und des Tests neuartiger am IIS/EAS entwickelter Bildsensoren wurde eine Vielzahl verschiedener Bias-Signale benötigt. Für diese in der vorliegenden Konfiguration acht Ströme und Spannungen im Bereich I = 1…500 µA, U = 0…3.3 V wurden frei programmierbare Präzisionsquellen entwickelt. Die Universalität, je nach Konfiguration bis zu 254 Kanäle und I = 0..100 mA und U = -15..+15V, ermöglicht eine Reihe von Anwendungen bei Test und Verifikation analoger und Mixed-Signal-Schaltkreise und Baugruppen. Somit ist es möglich definierte digitale und analoge Signale statisch an Chips anzulegen, Spannungen und Ströme vorzugeben bzw. zu begrenzen und gleichzeitig die entsprechenden an den Klemmen vorliegenden Werte zu messen.

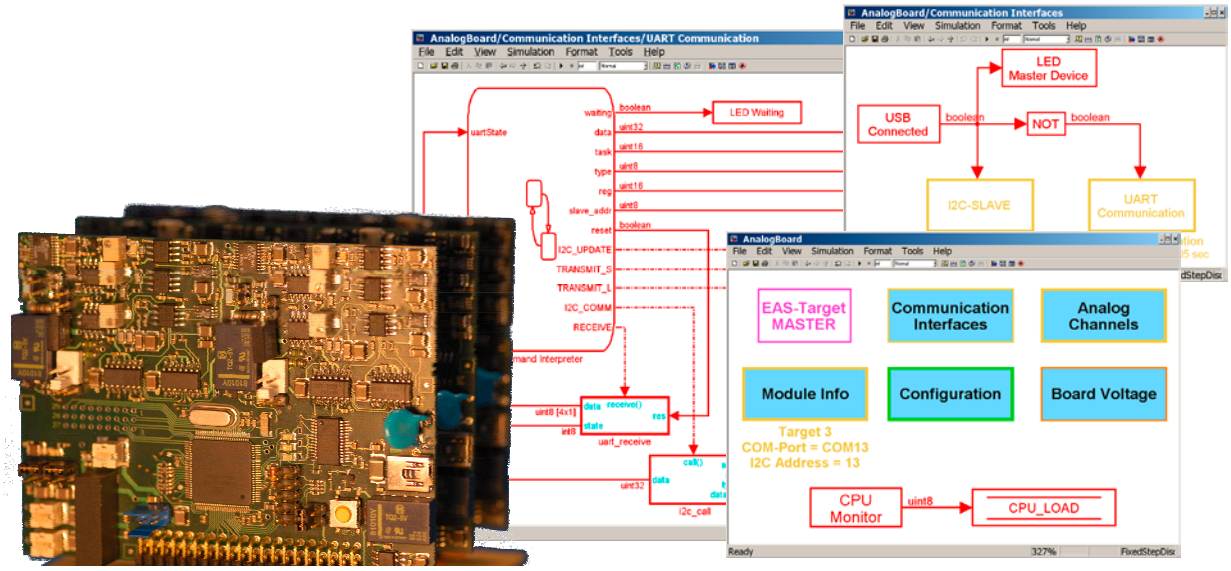


Abb. 3 Präzisionsquellen auf einer gemeinsamen Backplane (links) und Simulink-Modell der Applikationssoftware (rechts)

Auf den zweikanaligen Modulen wurde ein 8-bit-Mikrocontroller (ATmega2560) verwendet, welcher die Ansteuerung der über SPI-Bus angeschlossenen 16-bit-DAC's und -ADC's des Analogteils sowie des Stromreglers am Ausgang eines jeden Kanals übernimmt. Über eine UART des Controllers und einen Umsetzer auf USB können die Module an einen Messrechner angeschlossen und mit Strom versorgt werden. Dies ermöglicht es, basierend auf einer zugeschnittenen Kommandosprache, die Module zum Zweck der Konfiguration und Kalibrierung sowie zur Durchführung automatisierter Messabläufe über ein Terminalprogramm oder auch skriptbasiert anzusteuern. Hierbei können bis zu 127 dieser Module über I^2C-Bus miteinander verbunden und über ein einzelnes, frei wählbares „Mastermodul" angesteuert werden. Außerdem ist eine Synchronisation der Module über zusätzliche herausgeführte digitale Ein- und Ausgänge möglich.

Die softwareseitige Umsetzung erfolgte vollständig in Matlab/Simulink (vgl. Abb. 3 rechts), wobei alle notwendigen Hardware-Schnittstellen mittels der durch den EAS-Target bereitgestellten Treiberblöcke realisiert werden konnten. Mittels eines in Stateflow umgesetzten Kommando-Interpreters sowie der Implementierung der über den USB-Anschluss automatisch umschaltenden Master-/Slave-Funktionalität, wurde der schreibende/lesende Zugriff auf interne Variablen der Module (z.B. Mess- und Kalibrierungswerte, Statusflags) ermöglicht. Hierbei wurde, um die Übersichtlichkeit der Zustandsdiagramme zu wahren, teilweise auf die Verwendung von „Embedded Matlab"-Code zurückgegriffen. Die für ein jedes Modul und einen jeden Kanal individuelle Kalibrierung der analogen Ein- und Ausgänge konnte hierbei zu weiten Teilen automatisiert vorgenommen und unter Verwendung des internen EEPROM des Controllers dauerhaft gespeichert werden. Durch die Modularisierung in „Triggered"-, „Enabled"- und „Function Call"-Subsysteme, die nur bei definierten Ereignissen (z.B. externer Interrupt) oder bestimmten Zuständen aktiv sind, konnten die mit jedem Zeitschritt zu berechnenden Modellbestandteile auf ein Minimum reduziert werden. Somit konnten, trotz der für den verwendeten 8-bit-Mikrocontroller nicht unerheblichen Komplexität der entstandenen Software, Modell-Abtastzeiten von 5 ms sowie Abtastfrequenzen der analogen Kanäle von 2 kHz realisiert werden.

Beispiel 2: Beatmungssystem mit grafischer Bedieneinheit

Als zweites Beispiel soll hier ein auf Basis von Matlab/Simulink und der EAS-Target-Erweiterung entworfenes Beatmungsgerät vorgestellt werden. Das am Zentrum für Brennstoffzellen Technik (ZBT) in Duisburg entwickelte System [5] basiert auf einer PEM-Brennstoffzelle (Proton Exchange Membran Brennstoffzelle) und dient der Bereitstellung von Atemluft mit reduziertem Sauerstoffgehalt und definierter Luftfeuchte. Durch ein gegenüber konventionellen Brennstoffzellensystemen abweichendes Regelungskonzept wird hierbei die gezielte Konditionierung der dem System eingangsseitig zugeführten Umgebungsluft erreicht.

Für die Implementierung der Regelalgorithmen wurde das in Abb. 4 links dargestellte Steuergerät mit einem 8-bit-Mikrocontroller (ATmega128 von Atmel) entworfen [6]. Hierüber wird einerseits die Ansteuerung der für den Betrieb des Systems erforderlichen Aktoren (Pumpen, Lüfter, Ventile) realisiert, wofür über eine Vielzahl von Digital- und PWM-Ausgängen Lasten bis 20 A (Stromsenke) geschalten werden müssen. Andererseits werden über das Steuergerät unterschiedliche Sensoren (Analog-, I^2C-, SPI-Sensoren), die die für die Regelung und den sicheren Betrieb erforderlichen Momentanwerte (z.B. Temperatur, Luftfeuchte, …) des Systems erfassen, über entsprechende Schnittstellen angebunden. Die Bedienung und Konfiguration des Gerätes erfolgt über ein übersichtliches und für unterschiedliche Zielgruppen leicht anzupassendes grafisches Display mit Touch-Funktion, welches direkt auf der Oberseite des Steuergerätes angebracht wurde.

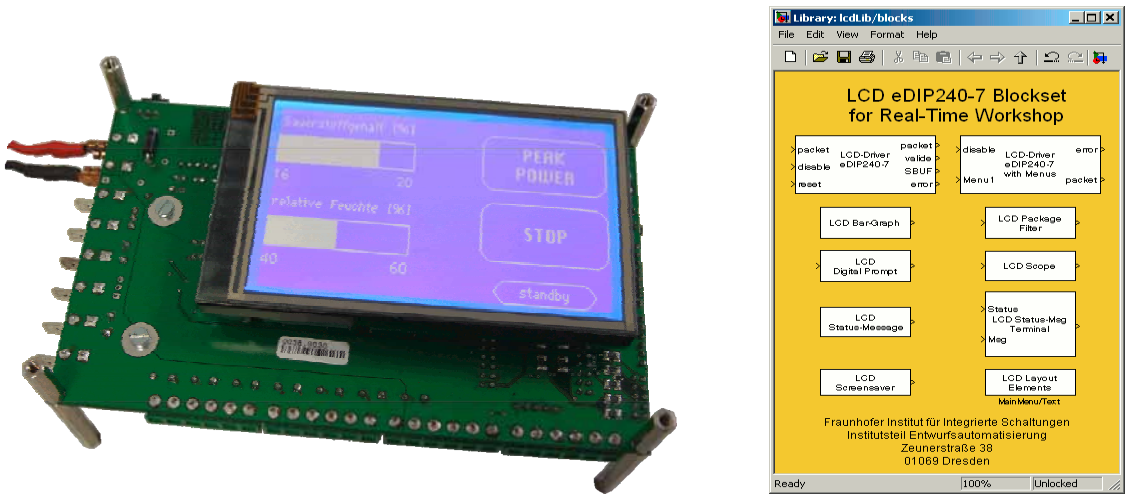


Abb. 4 Steuergerät mit Bedieneinheit (links) und Bibliothek für das grafische Touch-Displays (rechts)

Aufgrund der Komplexität des Systems und den zu berücksichtigenden sicherheitskritischen Aspekten wurde für den Softwareentwurf ein modellbasierter Ansatz gewählt. Das für die Systemsimulation benötigte Prozessmodell wurde in Matlab/Simulink erstellt, um die zu entwerfenden Regelalgorithmen unter Verwendung des C-Code-Exports direkt auf einem Mikrocontroller implementieren zu können. Die Stabilität der entworfenen Regelalgorithmen in unterschiedlichen Betriebszuständen des Systems sowie die zuverlässige Abschaltung des Systems bei Ausfall systemkritischer Komponenten konnten somit bereits in der Simulation untersucht und sichergestellt werden. Unter Verwendung des EAS-Target konnten die erforderlichen Schnittstellen zwischen dem Regler der Peripherie des Mikrocontroller vereinbart werden, so dass der für den Mikrocontroller generierte Programmcode bereits frühzeitig am realen System getestet werden konnte. Die gesammelten Erfahrungen konnten somit in effizienter Weise zur weiteren Optimierung der entworfenen Steuer- und Regelalgorithmen genutzt werden, wobei die vorgenommenen Änderungen am Modell stets durch Simulation des Gesamtsystems verifiziert werden konnten.

Für die schrittweise Integration des über eine serielle Schnittstelle (UART) angebundenen Grafikdisplays wurden auf Basis des EAS-Target spezielle Funktionsblöcke entwickelt (vgl. Abb. 4 rechts), welche einerseits den Entwurf der menübasierten grafischen Oberfläche und die Programmierung des Display-Controllers ermöglichen und andererseits die zur Interaktion mit dem Nutzer (Eingabe von Parametern, Visualisierung von Signalen und Systemzuständen) notwendige Funktionalität auf dem Mikrocontroller implementieren. Die nachträgliche Anpassung der Nutzerschnittstelle ist somit direkt aus dem Modell heraus möglich und steht auch bei zukünftigen Anwendungen zur Verfügung.

Die finale Version des Modells konnte mit einer eingestellten Abtastzeit von 20 ms auf dem 8-bit-Mikrocontroller implementiert werden, wobei für die Behandlung der durch Eingaben am Display ausgelösten Ereignisse noch hinreichend Reserve vorhanden ist.

Zusammenfassung

Basierend auf den Entwurfswerkzeugen Matlab/Simulink und Real-Time Workshop wurde der EAS-Target, ein Embedded Target für die plattformspezifische Codegenerierung und die zugehörige Simulink-Bibliothek mit Treiberblöcken für typische Mikrocontrollerperipherie vorgestellt. Hierbei wurde gezeigt, dass sich durch die Abstraktion der in Simulink bereitgestellten Treiberblöcke von der konkreten Implementierung auf einem Mikrocontroller sowie der Verwendung eines einheitlichen API's für die mikrocontrollerspezifischen C-Treiber zahlreiche Vorteile ergeben. Hierzu zählt sowohl die Minimierung des Aufwands bei der Unterstützung unterschiedlicher Mikrocontrollertypen als auch die leichtere Portierbarkeit und Wiederverwendung von (Teil-) Modellen in zukünftigen Applikationen.

Die Anwendbarkeit der modellbasierten Entwurftechniken auf Basis des EAS-Target wurde exemplarisch an zwei Beispielen dargestellt. Es konnte gezeigt werden, dass die Effizienz des aus Simulink generierten C-Codes genügt, um selbst Applikationen mit komplexerem steuer- und regelungstechnischen Anspruch auf kleinen 8-bit-Mikrocontrollern umzusetzen. Die Vorteile des modellbasierten Ansatzes liegen hierbei in der abstrakten, zunächst hardwareunabhängigen funktionalen Beschreibung der Modelle, die innerhalb der Entwurfsumgebung simulativ untersucht, getestet und optimiert werden können. Durch die parallele Entwicklung von Hard- und Software sowie die automatische Codegenerierung konnten bei den gezeigten Anwendungsbeispielen frühzeitig Tests durchgeführt werden. Dies resultierte einerseits in einer deutlichen Reduzierung des Zeitraumes bis zur Fertigstellung der Systeme. Andererseits konnten durch den modellbasierten Ansatz simulative Untersuchungen durchgeführt werden, welche an der gezeigten Beispielapplikation, des auf einer PEM-Brennstoffzelle basierenden Beatmungsgerätes, überhaupt nicht oder nur mit stark erhöhtem Aufwand umsetzbar gewesen wären.

Literatur

[1] Turevskiy, A.; Gage, S.; Buhr, C.: Model-Based Design of a New Light-weight Aircraft, The MathWorks, Natick, MA, 01760

[2] Langenwalter, J.; Erkkinen, T.: Embedded Steer-by-Wire System Development, The MathWorks, Embedded World, 17-19 Feb. 2004, Nuremberg, Germany

[3] The MathWorks: Real-Time Workshop Embedded Coder Developing EmbeddedTargets, Sept. 2004.

[4] The MathWorks: Writing S-Functions, Aug. 2005

[5] Beckhaus, P.; Schoemaker, M.; Notthoff, Th., Souzani, S.; Heinzel, A., Frahm, L.: Nutzung des Kathodenabgases aus Brennstoffzellen für medizinische und sportliche Anwendungen. Brennstoffzelle - Forschung - Demonstration - Anwendung, VDI-Berichte 2036, VDI-Verlag, Düsseldorf, ISBN 978-3-18-092036-8, 2008

[6] Priwitzer, H.; Enge-Rosenblatt, O.; Schneider, P.; Wielens, S.; Goessling, S.: Model-based Control Design Techniques for Small Systems Using Low-power Microcontrollers. Embedded World 2009 – Exhibition & Conference, Nuremberg, Germany, March 3-5, 2009, Proceedings.

Packet Based Communication Network for a Neuromorphic VLSI System

Vasileios Thanasoulis, Stephan Hartmann, Matthias Ehrlich, Johannes Partzsch, Christian Mayr, René Schüffny
{vasileios.thanasoulis, stephan.hartmann, matthias.ehrlich, johannes.partzsch, christian.mayr}@tu-dresden.de, schueffn@iee.et.tu-dresden.de
Technische Universität Dresden
Stiftungsprofessur für Hochparallele VLSI-Systeme und Neuromikroelektronik,
01062 Dresden, Germany

Abstract

One of the main challenges in neuromorphic VLSI systems is the design of the communication infrastructure. A high-speed communication network is required for the transmission of spike events between the neuron circuits on the neuromorphic ICs. We present the architecture and the implementation of such a network, centered around a custom-designed communication board. Besides the pulse stimulation, this board also allows for the complete configuration and control of the connected neuromorphic circuits by a host PC. The corresponding host-FPGA interface was optimized for throughput, while implementing error correction and frame ordering. With the two employed Gbit-Ethernet links, an effective throughput of almost 2×1 Gbit/s can be achieved, allowing for fast system configuration and monitoring.

1 Introduction

The last years have seen a steady increase in the size of neuromorphic systems in order to handle progressively more advanced computational tasks. One of the main challenges in neuromorphic VLSI systems is the design of the communication infrastructure. A high-speed communication network is required for transmission of so-called spike events between the neuronal circuitry on neuromorphic chips. Requirements on integration and communication bandwidth increase significantly when moving to a large-scale hardware system as currently developed in the *BrainScaleS* project, a successor of the recently finalized *FACETS* [1], [2].

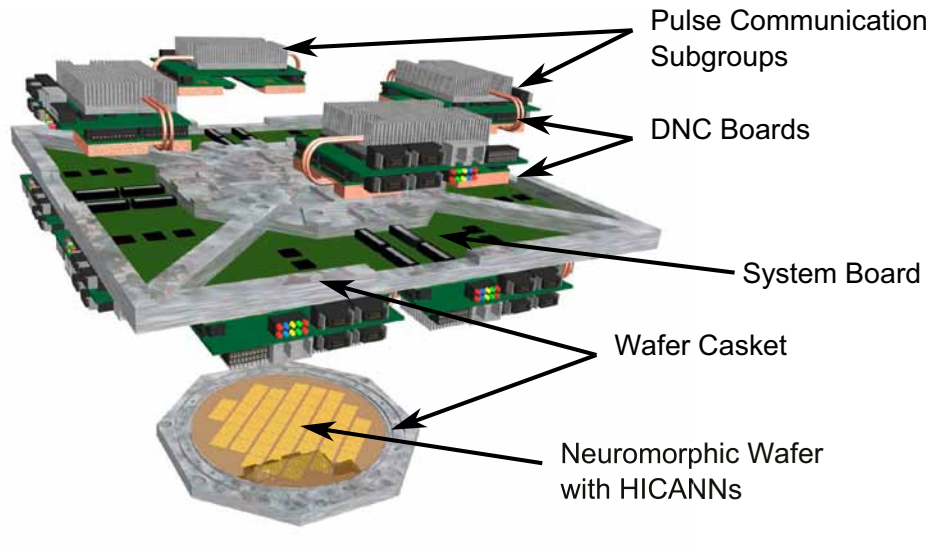


Figure 1: Concept of one wafer-scale module of the *BrainScaleS* system, taken with permission from [3]

1.1 A Large-Scale Neuromorphic Hardware System

The neuromorphic hardware system depicted in Figure 1 employs wafer-scale integration technology to gain a neuromorphic device with high connection density. Each wafer, which is shown on the bottom, will contain up to 2×10^5 neurons and 4×10^7 synapses [2]. The system is designed

for operating at a nominal speed-up factor of 10^4 with respect to biological real-time and shall provide a unique common research platform to study the dynamics of large-scale biologically inspired neural network models with an equivalent resolution in the order of microseconds over a period of years in biological real-time.

The neuromorphic wafers feature Adaptive Exponential Integrate&Fire neuron models [4] as well as various plasticity mechanisms. Pulse communication among the single dies of a wafer is achieved via a high density routing grid, named *Layer1* network which is directly implemented on the wafer. On each wafer, individual synapse-and-neuron blocks, called *High Input Count Analog Neural Networks (HICANNs)* [5] are connected directly via this *Layer1* network. External configuration and stimulation of the system as well as pulse communication is carried out through a dedicated high-speed packet-based communication network, named *Layer2*, which is described in this contribution. This network is composed of full-custom digital communication ASICs, called *Digital Network Chips (DNCs)*, and *FPGAs*. One FPGA and four DNCs form one so-called *Pulse Communication Subgroup (PCS)*, which is described in the following. The ASIC as well as the boards were designed at our chair [6], [7].

1.2 Pulse Communication Subgroup

The PCS shown in Figure 2 provides 2×1 Gbit/s *Ethernet* adapters for its connection with the *control host* and 4×10 Gbit/s sockets for connections to its counterparts on other wafers. It can furthermore connect to the *DNCs* via 4×16 Gbit/s links. Each of these FPGA-DNC links bundles 16 *Low Voltage Differential Signaling (LVDS)* lanes with *Double Data Rate (DDR)*, resulting in a raw data rate of 16 Gbit/s per link [6]. The PCS houses two *DDR2 SDRAMs* of different sizes: A 256 MByte frame storage and routing memory directly on-board and a 2 GByte *SODIMM* for storing playback and recording data.

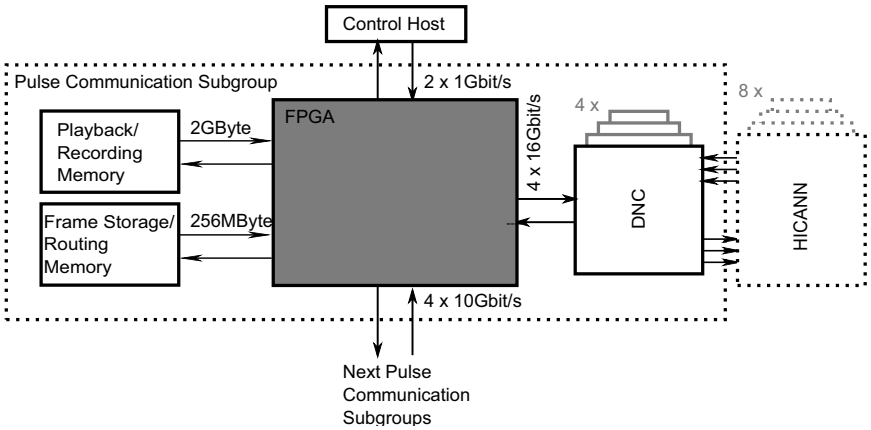


Figure 2: Schematic of the PCS connectivity and memory configuration

1.3 Packet Based Network

Figure 2 renders the logical structure of the packet based communication network from an FPGA-centric view. The design that is presented in this contribution implements a reliable communication network for configuration packets and pulse events that are transmitted to and from the wafer-scale neuromorphic system as well as between its single units. Data are routed from the control host via Ethernet to the PCS. A customized FPGA implementation of the *UDP/IP*[1] stack described herein ensures the synchronization with the control host side, the control of the dataflow and the reliability of the transmission. The packets are then forwarded to 4 DNCs connected to the PCS and then to the HICANNs of which 8 are connected to a DNC. Configuration data and pulse events can in turn be received at the control host following the same route but originating at the HICANNs.

The 256 MByte on-board memory is used as *Frame Storage* for buffering the host interface frames and for storing *Routing Tables* for pulse event redirection. The access and the storage

[1] User Datagram Protocol/Internet Protocol

of the frames in the memory are controlled by the FPGA design. The stored data can be routed from there to the DNCs, which forward them to the connected HICANNs, to a memory for pulse stimulation/storage or to a connected PCS of an adjacent wafer unit. From the same memory, upstream data can be sent back to the control host via the UDP/IP hardware stack.

The 2 GByte SODIMM memory is used for several supporting tasks concerning the experimental procedures. It is employed as *Playback* memory for stimulation data, such as input pulse events for the simulated neural network. At the same time, it operates as *Recording* buffer as traces of neural pulse events are also transferred to this memory. The stimuli packets are routed from the playback memory to the HICANNs and upstream pulse packets are copied to the recording memory constantly during an ongoing experiment.

2 Packet Format Definition

For the description of the UDP/IP stack we apply the abstraction layers as specified in [8]. Following the 64 bit UDP [9] header, a 32 bit *Synchronization* header is employed as depicted in Figure 3.

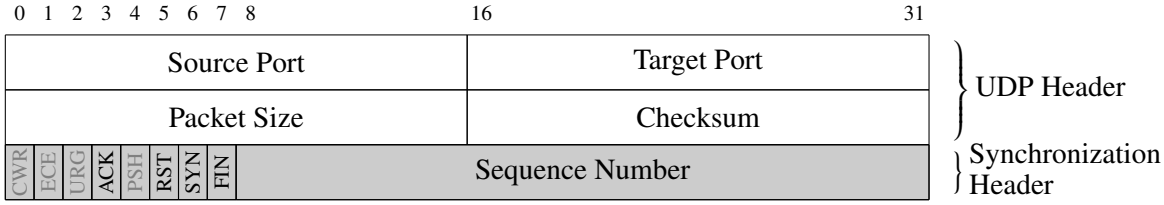


Figure 3: UDP and synchronization header

The synchronization header consists of 8 bits of flags which were adapted from the TCP[2] [10] and a 24 bit sequence number. Only the highlighted flags have been used to enable an initial handshake procedure and other basic control and synchronization functionalities.

The `ACK` and the `SYN` flags are used by both the control host and FPGA sites during the establishment of the communication and the synchronization of their connection. For this, a three-way handshake procedure was implemented, adapted from TCP. The `ACK` flag is further used to notify the host unit that the frame with the respective sequence number that is contained in the synchronization header has been stored successfully to the memory. The `RST` flag can be used for a synchronous reset of selected blocks of the FPGA design. The `FIN` flag finally informs that there is no more data to be expected from the sender.

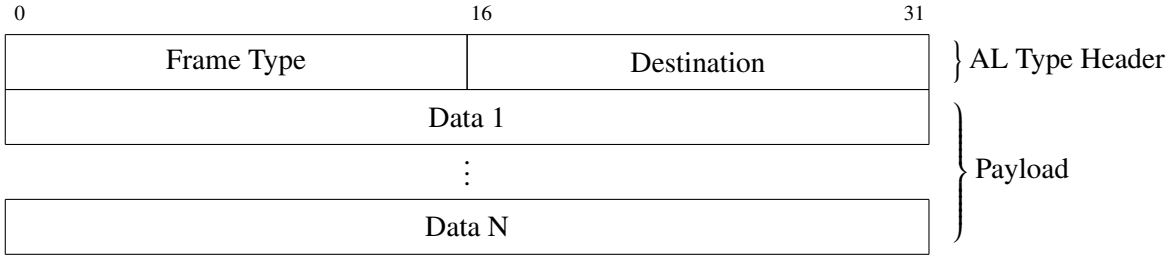


Figure 4: Application layer frame

On the *Application Layer (AL)*, we implemented the remaining frame as shown in Figure 4 with a maximum payload of 1500 Bytes. Each frame contains a 32 bit header that defines the type of the data and its destination. The remaining data is aligned to 32 bit.

3 FPGA Implementation

The FPGA design for the interface to the host consists of the UDP/IP protocol hardware stack, the AL controllers and the memory, as shown in Figure 5. The implemented UDP/IP communication stack consists of the lower three layers of the *IP suite* [8], i.e. the *Link-*, the *Internet-*

[2]Transmission Control Protocol

and the *Transport Layer*. The utilized communication scheme is optimized in terms of the network's throughput performance, i.e. Ethernet frames carry the maximum payload of 1500 Bytes wherever this is applicable.

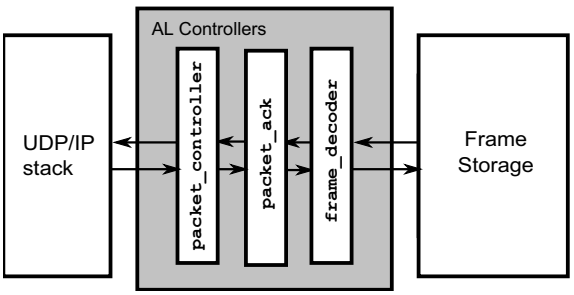


Figure 5: Simplified block structure of the data control and frame buffering architecture

A series of controllers, as illustrated in Figure 5 intervenes between the bidirectional transmissions from the UDP/IP hardware stack to the memory where frames are buffered for further processing. These AL controllers handle the initialization of the communication, the control of the bidirectional dataflow and the reliability of the communication between the control host and the target destinations. In particular, they implement:

- the acknowledgment back to the control host upon successful transmission,
- error correction by re-requesting dropped frames,
- the ordered storage of the configuration and pulse packets and
- the decoding and routing of the buffered data to other units of the neuromorphic system.

The `packet_controller` is responsible for the initialization and synchronization of the communication between the host side and the FPGA. Furthermore, it ensures the ordered storage of the configuration and pulse packets to the frame storage unit. This is performed by examining for each frame the sequence number, which is included in the synchronization header. The sequence of this number for the received frames from the host must be successive. Otherwise the storage of the received frames in the memory is paused and the frame with the correct sequence number is requested for resending from the host.

The `packet_ack` is responsible for the acknowledgments to the control host and storage of UDP frames to the memory. It also transfers frames, e.g. configuration data, back to the UDP/IP stack in case that the control host requested data from the FPGA or the connected HICANNs. The error correction in the communication is introduced by both the `packet_controller` and `packet_ack` modules by re-requesting dropped frames. The routing of the buffered data to the target destinations is performed by the `frame_decoder`. This is realized with the decoding of the frame type of the AL frame as shown in Figure 4 that includes the type of the packet and the destination.

4 Experimental setup

An experimental setup for the PCS is shown in Figure 6. It is utilized to directly measure the characteristics of the network connections that are realized by the control-host-FPGA, FPGA-FPGA and FPGA-DNC-HICANN connections. The latter is the main network connection to a single wafer for the routing of configuration data and pulse events. The setup is operated from the control host, allowing to fully characterize the packet based network and monitoring its traffic. From the monitored received and sent packets, the measurement is analyzed in the control host, calculating bandwidths, pulse loss and pulse delays.

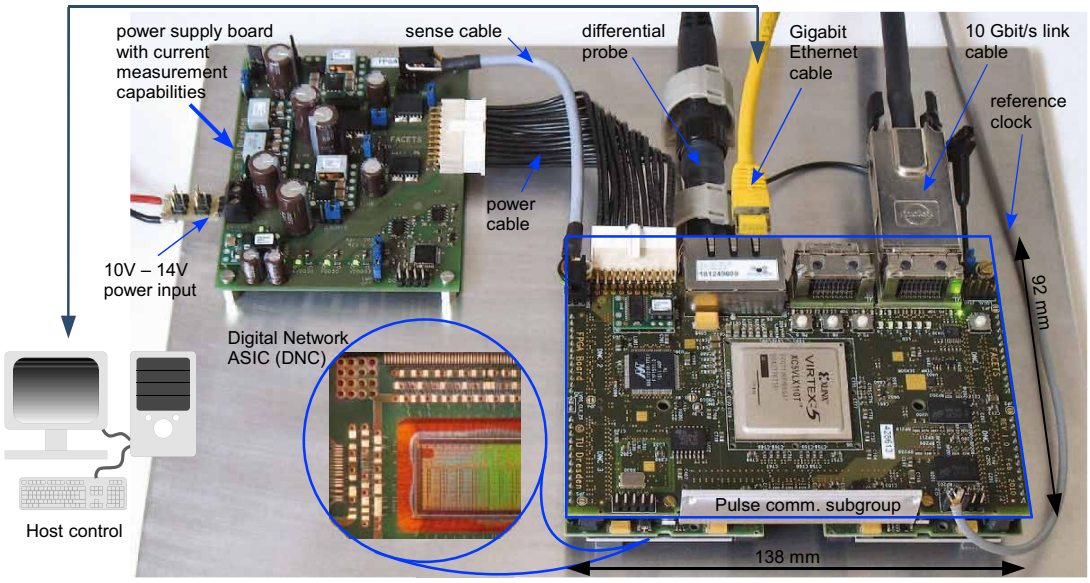


Figure 6: Experimental setup of the pulse communication subgroup

Reference	FPGA	reliable conn.	Slices	f_{clock}[MHz]	Throughput[Mbit/s]
[11]	Virtex2-XCV8000	full TCP	8318	37.5	350
[12]	Spartan3e-XC3S500e	no	1321	122.8	980[1]
[13]	Spartan3-XC3S200	TCP embedding	1584	105.6	957[1]
our design	Virtex5-LX110T	custom	1440	125	959[2]

Table 1: Comparison of the proposed design with other UDP/IP FPGA implementations.
[1] calculated bandwidth including headers
[2] calculated net payload bandwidth per channel

5 Results

The UDP/IP stack and AL controllers were developed for the *Xilinx Virtex5 LX110T* FPGA. The implementation was verified through simulation and by packet transmission/reception via a *JTAG* interface, with simultaneous monitoring of the network traffic.

Table 1 presents the results of the proposed design, concerning hardware and operational features, in comparison with other works that offer similar network services. Compared to the full TCP/IP stack implementation by Dollas et al. [11], our implementation is significantly faster and occupies almost a factor of 6 less area. This is especially achieved by avoiding the extensive functionality of the TCP protocol, replacing it by a dedicated, TCP-like protocol for error correction. Herrmann et al. [12] implement a UDP/IP stack whose performance values are comparable to our design, but which offers no mechanisms for ensuring a reliable connection. Lofgren et al. [13] present three hardware UDP/IP stack implementations, from which we only chose the ”Advanced” design for comparison, because it enables a reliable connection via TCP frame embedding. Again, this implementation is comparable to ours, but offers the error correction functionality only indirectly and with an unspecified lower throughput [13].

In conclusion, our implementation offers a reliable connection via UDP/IP at virtually no performance penalty concerning speed and area compared to other UDP/IP implementations that offer only indirect error correction handling. Thus, it is perfectly matched to the high throughput requirements that arise from the application in an accelerated neuromorphic hardware system.

6 Conclusion

A packet based communication platform for application in a wafer-scale neuromorphic hardware system was presented. The proposed network provides pulse event packet and configuration data transmission to all the individual synapse-and-neuron chips of the wafer via high speed communication links. The high-speed serial communication covers the entire chain from the

control host to the FPGA, then to DNCs and down to the neuromorphic chips and backwards.

Our implementation allows the configuration of the complete wafer-scale neuromorphic system, i.e. HICANNs, DNCs and FPGAs via the Ethernet interface. Compared to other communication platforms that offer similar network services, our implementation achieves one of the highest throughputs while offering a small area utilization (cf. Table 1). This is accomplished by implementing only these protocol features that are required by the neuromorphic application.

Packet transmission errors have been compensated with the introduction of a frame acknowledge mechanism, ensuring transmission reliability and correct ordering. Specifically for the operation in the neuromorphic system, our design contains a large recording and playback memory, which enables stand-alone operation for minimally 0.25 s in technical time, corresponding to approx. 40 min biological real-time at the employed speed-up factor of 10^4. However, in practice, much larger time spans may be recorded, because the number of traced spike sources is usually smaller than the allowed maximum. This long-term tracing is required e.g. for simulating learning behaviour [14]. Also, the high speed communication links can support closed loop experiments and recurrent network simulations, which demand relatively low latency over the network [15].

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement no 237955 (FACETS-ITN) as well as under grant agreement no 269921 (BrainScaleS)

References

[1] M. Ehrlich, C. Mayr, H. Eisenreich, S. Henker, A. Srowig, A. Grübl, J. Schemmel, and R. Schüffny. Wafer-scale VLSI implementations of pulse coupled neural networks. In *Proceedings of the 4. IEEE International Conference on Sensors, Circuits and Instrumentation Systems*, 2007.

[2] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *ISCAS 2010*, pages 1947–1950, 2010.

[3] BrainScaleS Homepage. `http://www.brainscales.org`.

[4] Romain Brette and Wulfram Gerstner. Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity. *Journal of Neurophysiology*, 94:3637–3642, 2005.

[5] Sebastian Millner, Andreas Grübl, Karlheinz Meier, Johannes Schemmel, and Marc-Olivier Schwartz. A VLSI Implementation of the Adaptive Exponential Integrate-and-Fire Neuron Model. *Advances in Neural Information Processing Systems*, 23:1642–1650, 2010.

[6] S. Hartmann, S. Schiefer, S. Scholze, J. Partzsch, C. Mayr, S. Henker, and R. Schüffny. Highly integrated packet-based AER communication infrastructure with 3Gevent/s throughput. In *17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), 2010*, 2010.

[7] S. Scholze, S. Henker, J. Partzsch, C. Mayr, and R. Schüffny. Optimized queue based communication in VLSI using a weakly ordered binary heap. In *IEEE International Conference on Mixed Design of Integrated Circuits and Systems MIXDES 2010*, pages 316–320, 2010.

[8] R. Braden (Editor). *RFC1122: Requirements for Internet Hosts – Communication Layers*, 1989.

[9] J. Postel. *RFC768: User Datagram Protocol*. ISI, 1980.

[10] DARPA Internet Program. *RFC793: Transmission Control Protocol*. DARPA INTERNET PROGRAM, 1981.

[11] A. Dollas, I. Ermis, I. Koidis, I. Zisis, and C. Kachris. An open TCP/IP core for reconfigurable logic. In *IEEE FCCM*, pages 297–298, 2005.

[12] F.L. Herrmann, G. Perin, J.P. Freitas, R. Bertagnolli, and J.B. Martins. An UDP/IP network stack in FPGA. In *Student Forum on Microelectronics*, 2009.

[13] A. Löfgren, L. Lodesten, and S. Sjöholm. An analysis of FPGA-based UDP/IP stack parallelism for embedded Ethernet connectivity. In *23rd. NORCHIP Conference*, pages 94–97, 2005.

[14] J.-P. Pfister and W. Gerstner. Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of Neuroscience*, 26(38):9673–9682, 2006.

[15] S. Hill and G. Tononi. Modeling sleep and wakefulness in the thalamocortical system. *Journal of Neurophysiology*, 93:1671–1698, 2005.

Modellgestützter und automatisierter Entwurf von Zustandsüberwachungssystemen

Christian Bayer, Olaf Enge-Rosenblatt
{Christian.Bayer; Olaf.Enge}@eas.iis.fraunhofer.de
Fraunhofer-Institut für Integrierte Schaltungen IIS/EAS, Dresden

Kurzfassung

Zur Beurteilung des qualitativen Zustandes von Maschinen und Anlagen werden zunehmend Zustandsüberwachungssysteme (CMS) eingesetzt. CMS erkennen frühzeitig eventuelle Schäden und ermöglichen eine bedarfsgerechte Wartungsstrategie. In diesem Beitrag stellen wir den modellgestützten Entwurf von CMS vor. Modelle vergrößern die zum Entwurf nötige Datenbasis und führen zu einem besseren Verständnis des Gesamtsystemverhaltens. Als Demonstrator dient eine Axialkolben-Pumpe, deren Modell vorgestellt und zur Generierung von Signalen genutzt wird. Auf Basis der Signale werden Merkmale für den Klassifikator abgeleitet. Für eine statistisch sichere Klassifikation der Zustände des Demonstrators wird ein automatisches Verfahren zur Bestimmung von Merkmalen mit optimalen Trenneigenschaften genutzt.

Einleitung

Maschinen und Anlagen unterliegen im Betrieb einer mit der Zeit zunehmenden Abnutzung, was deren Funktion negativ beeinflussen kann. Zur Beurteilung dieses qualitativen Zustandes gewinnen automatisierte Überwachungssysteme (Condition Monitoring Systems, CMS) in vielen Anwendungsbereichen zunehmend an Bedeutung [1, 2]. Die frühe Erkennung von Schäden oder Fehlern begünstigt eine angepasste Strategie der Wartung, trägt zur Kostensenkung bei und steigert auch die Qualität von Produkten.

Eine wesentliche Grundlage für ein zielgerichtet einsetzbares CMS ist eine möglichst genaue Kenntnis der betreffenden Anlage. Verschiedene Sensoren liefern Signale, die den Zuständen zugeordnet werden müssen. Es sind deshalb im Vorfeld zahlreiche Messungen erforderlich, um ein CMS für den späteren Einsatz richtig zu parametrisieren. Dieser Vorgang ist aufwändig und in der Regel können nicht alle Verschleißzustände der Maschine für eine Messung bereitgestellt werden. Es bietet sich daher an, ein Modell der Anlage zu entwerfen, das die betreffenden Zustände nachbilden kann und künstliche Sensorsignale liefert [3]. In diesem Beitrag demonstrieren wir diese Vorgehensweise anhand einer Axialkolben-Schrägscheibenpumpe. Fehler und Verschleiß lassen sich dann durch Modifikation entsprechend zugehöriger Modellparameter nachbilden.

Der Entwurf eines CMS erfordert einen großen Datensatz. Mit Hilfe des Modells und Messungen am realen Objekt lässt sich eine hinreichend große Menge Daten generieren. Es handelt sich hier in der Regel um statische oder zeitabhängige Größen. Letztere werden mit geeigneten Signalverarbeitungsalgorithmen für die Merkmalsfindung aufbereitet. Merkmale sind in Zahlen beschreibbare Eigenschaften des Systems. Statistische Parameter wie die Standardabweichung gehören dazu. Das CMS besitzt einen Klassifikator, der Merkmale bestimmten Zuständen zuordnen kann. Das Auffinden geeigneter Merkmale ist daher von großer Bedeutung. In den meisten Fällen sind diese nicht offensichtlich, weshalb wir in diesem Beitrag eine automatisierte Suche vorstellen. Das Ergebnis liefert sowohl die für eine Klassifikation optimalen Merkmale als auch die Untermenge der Daten aus denen sie generiert werden müssen.

Der automatisierte CMS-Entwurf birgt erhebliches Potential, da ein optimales Verhalten mit vergleichsweise geringem Aufwand erreicht werden kann. Die Modellierung der Anlage

verbreitert zudem die verfügbare Datenbasis und erlaubt somit Ergebnisse von höherer statistischer Sicherheit.

Modellgestützte Zustandsüberwachungssysteme

Zustandsüberwachungssysteme

Ein Zustandsüberwachungssystem soll in der Lage sein, graduelle Veränderungen eines Systems zu erkennen. Eine wesentliche Ursache für Veränderungen kann Verschleiß sein, so dass mit Hilfe des CMS eine bedarfsgerechte Wartung durchgeführt werden kann. Die Verlässlichkeit von Maschinen kann sich dadurch erhöhen bei gleichzeitiger Senkung der Kosten.

Das CMS benötigt zahlreiche Informationen über das betreffende System, die in der Regel von Sensoren geliefert werden. Abb. 1 zeigt schematisch wesentliche Komponenten eines CMS.

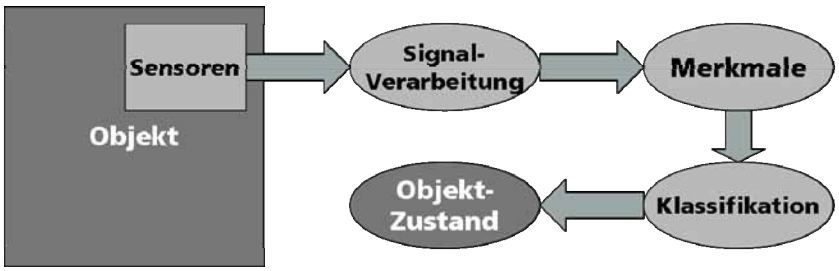


Abb. 1: Prinzip eines CMS

Die Sensorsignale werden durch geeignete Signalverarbeitungsalgorithmen aufbereitet und daraus Merkmale generiert. Die Wahl dieser Merkmale ist von entscheidender Bedeutung für die sich anschließende Klassifikation. Hier wird anhand der Merkmale entschieden, in welchem Zustand sich das System befindet. Klassifikatoren können beispielsweise Neuronale Netze sein, wie in diesem Beitrag noch gezeigt wird.

Modellunterstützung

Für die Bewertung von Merkmalen bedarf es eines tieferen Verständnisses des betreffenden Systems. Modelle spielen bei der Untersuchung des Verhaltens von Systemen eine zunehmend größere Rolle. Ursachen und Wirkung von Verschleiß können somit unter nahezu beliebigen Bedingungen getestet werden, was zu besseren Ansätzen für das individuelle CMS-Design führt. Durch Simulationen wird die zur Verfügung stehende Datenbasis deutlich verbreitert, wie in Abb. 2 dargestellt ist.

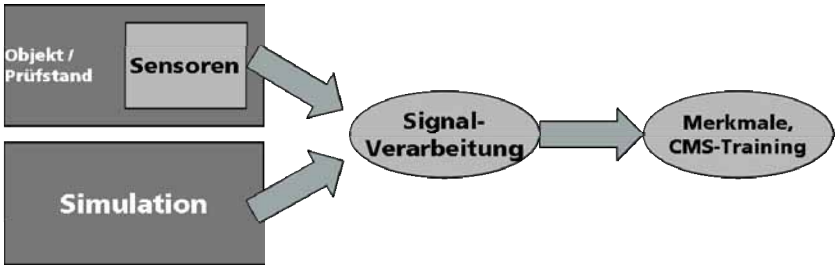


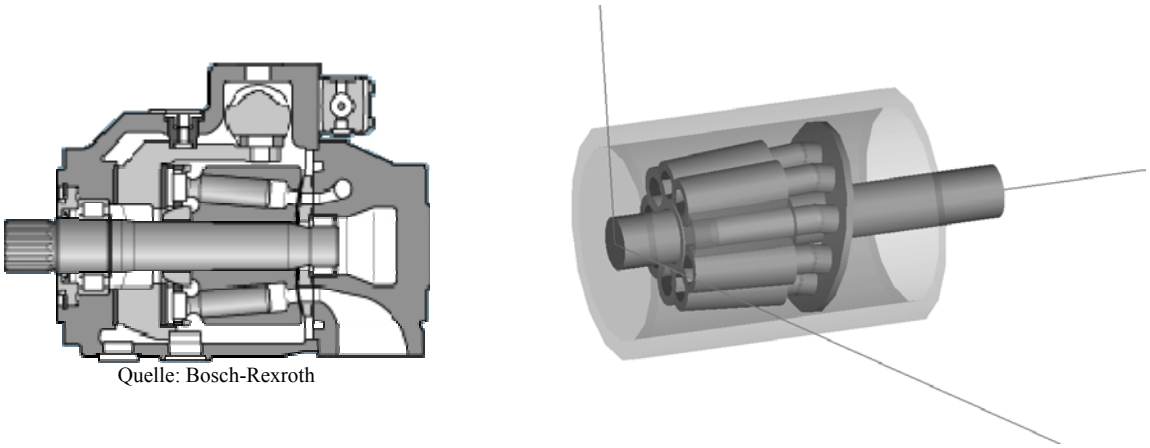
Abb. 2: Erweiterung des CMS um ein Modell

Neben dem Experiment können jetzt zusätzlich simulierte Daten in die Signalverarbeitung einfließen. Potenziell geeignete Merkmale für eine Klassifikation lassen sich somit leichter finden und auch das Training des CMS wird sich mit einer Erhöhung der Datenmenge verbessern lassen.

Modell der Axialkolben-Pumpe

Modell und Parameteroptimierung

Als Demonstrator für den CMS-Entwurf wurde eine Axialkolben-Pumpe gewählt. Sie findet Verwendung bei der Förderung von Öl in hydraulischen Hochdrucksystemen und ist trotz ihres langen und bewährten Einsatzes noch Gegenstand aktueller Forschung hinsichtlich der Modellierung [4, 5]. Die Pumpe unterliegt Verschleißerscheinungen, die ihre Ursachen in der starken mechanischen Belastung der Baugruppen und im Materialabtrag durch Kavitation haben.



Quelle: Bosch-Rexroth

Abb. 3: Querschnitt einer Axialkolben-Pumpe (links) und 3D-Modell (rechts).

Abb. 3 zeigt schematisch den Aufbau einer solchen Pumpe. Im Wesentlichen besteht sie aus einer rotatitionssymmetrischen Aufreihung von 9 Kolben und Zylindern, die ihrerseits über eine Ventil-Steuerplatte mit Zu- und Abfluss für die Hydraulikflüssigkeit verbunden sind. Der Kolbenhub wird über das Führen der Kolbenköpfe entlang einer feststehenden, geneigten Scheibe realisiert, wie es im Modell in Abb. 3 dargestellt ist.

Das Modell wurde mit Hilfe der Modellierungssprache Modelica [6] erstellt. Es enthält die für das CMS wesentlichen Komponenten. Als Messgröße dient bei dem hier vorgestellten CMS die axiale Beschleunigung, also die Vibrationen in Achsrichtung. Das Modell wurde daher auf eine Dimension reduziert, so dass nur noch Kräfte und Bewegungen entlang der Rotationsachse eine Rolle spielen. Abb. 4 links zeigt ein Schema des reduzierten Modells.

Das Modell enthält zahlreiche Parameter, die zum Teil unbekannt sind, da sie einer Messung nicht direkt oder nur mit größtem Aufwand zugänglich sind. Mit Hilfe von Referenzmessungen an der Pumpe und der Verwendung von Optimierungsverfahren können solche Parameter jedoch sehr gut approximiert werden. Das Ergebnis dieser Vorgehensweise zeigt Abb. 4 rechts. Dargestellt sind zwei Beschleunigungssignale für Messung und Simulation in Achsrichtung für genau eine Umdrehung der Pumpe. Die Übereinstimmung der Signale ist relativ gut, so dass das Modell jetzt für die Untersuchung verschiedener Zustände geeignet ist.

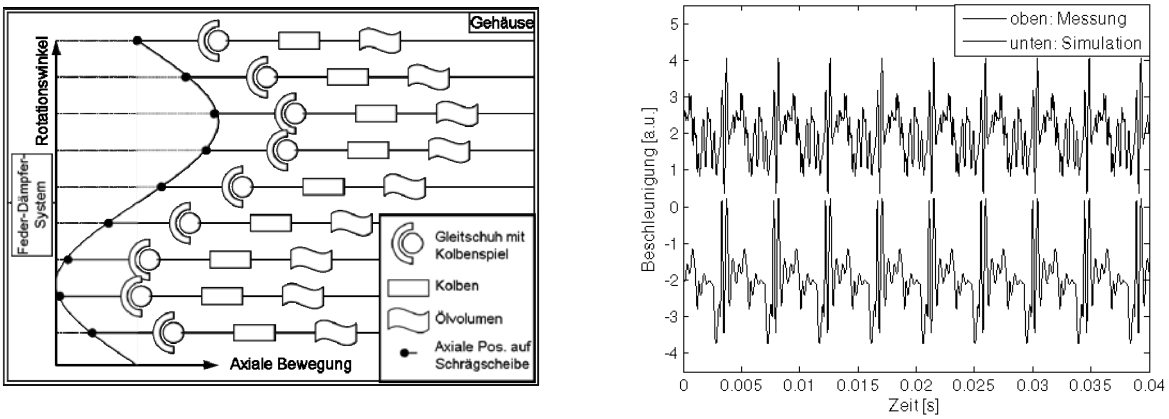


Abb. 4: Reduziertes, eindimensionales Modell (links) und Vergleich von Mess- und Modellsignalen der Beschleunigung (rechts).

Die Pumpe rotiert im Versuch mit konstanter Drehzahl, so dass alle Zeitsignale streng periodisch sind. In diesem Fall bietet sich die Betrachtung der Frequenzspektren an (siehe Abb. 5).

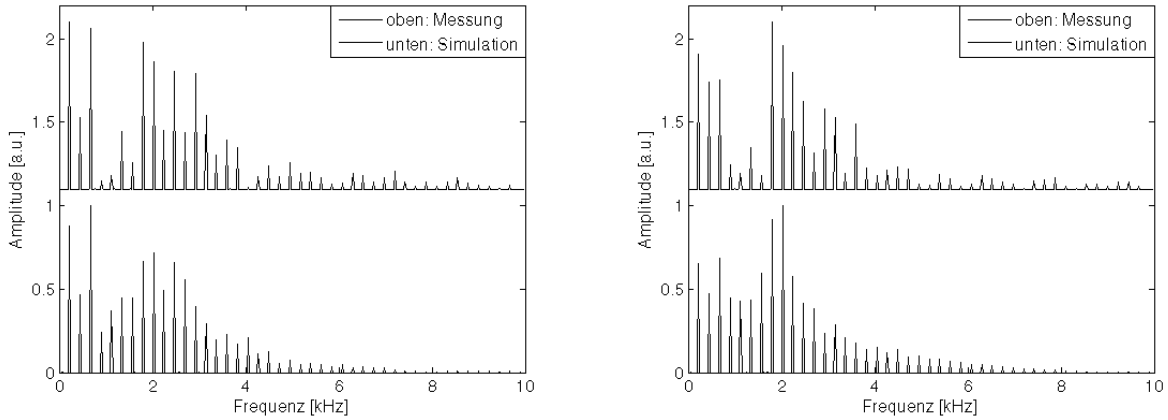


Abb. 5: Vergleich der Spektren von Mess- und Modellsignal einer ungeschädigten Pumpe (links) und einer Pumpe mit erhöhtem Axialspiel der Kolben (rechts)

Das CMS wird in dem hier präsentierten Beispiel mit Merkmalen versorgt, die ausschließlich den Spektren entnommen werden. Mit zunehmender Schädigung der Pumpe verändert sich die Verteilung der Harmonischen. Das CMS muss in der Lage sein, diese Veränderung erkennen und bewerten zu können.

Modellierung von Verschleiß

Nachdem ein Modell mit gültigen physikalischen Parametern erstellt wurde, können diese nun entsprechend der Verschleißerscheinungen variiert werden. Im vorliegenden Fall ist das Einbringen eines erhöhten Kolbenspiels vorgesehen. Abb. 5 (rechts) zeigt einen Vergleich von gemessenem und simuliertem Spektrum für eine Pumpe mit diesem Fehler. Die relativen Veränderungen im Spektrum gegenüber den Werten einer schadensfreien Pumpe sind für Messung und Simulation ähnlich. Das Modell kann also das Verhalten einer Pumpe mit Axialspiel im Wesentlichen abbilden.

Automatische Generierung von optimalen Merkmalen

Die hier verwendeten Merkmale sind statistische Parameter der Verteilung der Harmonischen. Beispiele sind Schiefe oder Standardabweichung. Insgesamt wurden 12 mögliche statistische Parameter und damit Merkmale definiert. Nun ist es nicht unbedingt sinnvoll, diese Parameter für das gesamte Spektrum zu errechnen. Am Beispiel aus Abb. 5 ist zu erkennen, dass sich Änderungen hauptsächlich innerhalb zweier Gruppen von Harmonischen ergeben. Besondere Beachtung würden hier die Gruppe der ersten drei Harmonischen und die Gruppe um 2 kHz finden. Verallgemeinert man diesen Ansatz, so kann man alle möglichen 2^n Gruppierungen zulassen, wobei n die Anzahl der betrachteten Harmonischen ist. Auf jeder dieser Gruppen lassen sich nun die statistischen Merkmale bestimmen, wodurch sich je nach Zustand der Pumpe Matrizen analog Abb. 6 ergeben.

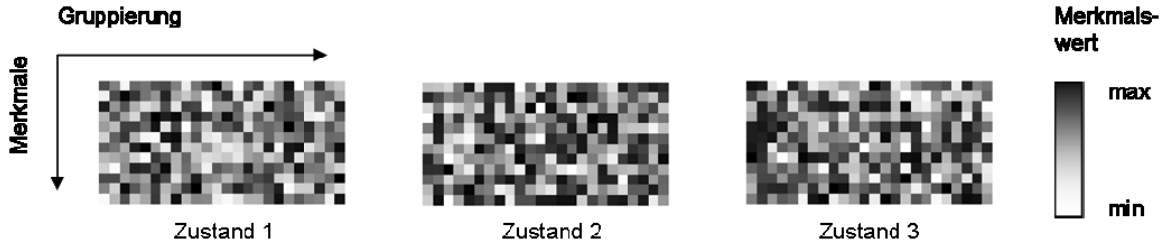


Abb. 6: Merkmals-Matrizen für drei verschiedene Zustände.

Vergleicht man die Matrizen für jeden Zustand elementweise, kann man je Merkmal diejenige Gruppe bestimmen, die optimale Trenneigenschaften bezüglich der Zustände verspricht. Dieser Vorgang wird automatisiert durchgeführt und kann auch auf andere Problemstellungen übertragen werden. Sind die in diesem Fall 12 optimalen Merkmal-Gruppe-Kombinationen gefunden, dienen diese als Eingangsgrößen für den Klassifikator. Da jedes Merkmal für sich bereits optimale Trenneigenschaften besitzt, wird das Klassifikationsergebnis statistisch deutlich sicherer.

Klassifikation

In den hier präsentierten Arbeiten wurde als Klassifikator ein Neuronales Netz (NN) genutzt. Insgesamt drei Zustände der Pumpe mussten vom NN erkannt werden. Neben dem axialen Spiel ist eine Pumpe mit Kavitationsschäden vermessen worden. Die Datenbasis für das Training umfasste Messungen der schadensfreien Pumpe und der Pumpe mit Kavitationsschaden. Das Modell lieferte zusätzliche Daten für Schadensfreiheit und außerdem die alleinigen Daten für Pumpen mit Axialspiel. Klassifiziert wurden jedoch ausschließlich Messdaten.

Abb. 7 zeigt zunächst den prinzipiellen Aufbau des NN. Die Ausgangsneuronen sind den Klassen zugeordnet und liefern im Idealfall 1 bei Klassenzugehörigkeit, 0 sonst. Rechts ist das Ergebnis der Zuordnung verzeichnet. Jedem geprüften Datensatz ist ein loser Datenpunkt ohne Linie zugeordnet. Linien mit Datenpunkten entsprechen den vorab bekannten Werten und dienen als Vergleich. Das NN ist in der Lage, die Klassen mit akzeptabler Sicherheit zuzuordnen.

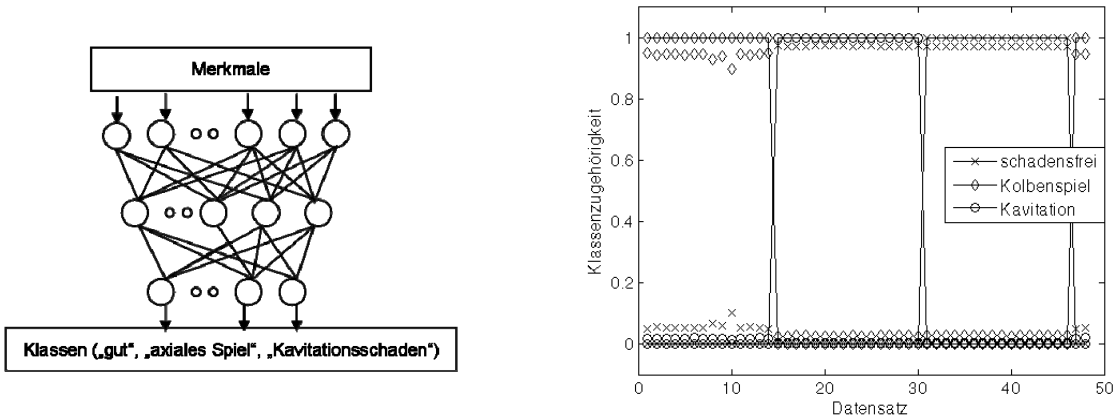


Abb. 7: Neuronales Netz als Klassifikator (links) und Klassifikationsergebnis (rechts).

Zusammenfassung

Der Entwurf komplexer, mechatronischer Systeme profitiert enorm von der Modellbildung und Simulation. Anhand der Axialkolben-Pumpe konnten wir das Potenzial aufzeigen, das die Modellierung im Bereich des CMS-Entwurfs besitzt. Ein Modell vergrößert die verfügbare Datenmenge für das CMS-Training, wenn entsprechende Messungen schwer oder auch gar nicht möglich sind. Darüber hinaus verbessert sich im Allgemeinen das Verständnis für das Gesamtverhalten eines Systems, was wiederum Impulse für ein optimiertes CMS-Design hervor bringt. Von großer Bedeutung ist die Generierung von Merkmalen aus Sensorsignalen. Hier konnten wir einen Ansatz zur automatischen Bestimmung optimaler Merkmale aufzeigen. Diese wurden einem Neuronalen Netz als Klassifikator zugeführt. Modell- und Messdaten dienten dem Training des NN, das anschließend Fehler in der Baugruppe korrekt erkannte. Mit diesem Verfahren ist es möglich, modellbasiert und, falls erforderlich, auch ohne Referenzmessungen geschädigter Baugruppen ein CMS zu entwerfen.

Die vorliegenden Ergebnisse entstanden im Rahmen des Projektes MoSiKMA, gefördert durch das BMBF. Die beteiligten Partner sind die GEMAC GmbH, Chemnitz, das Fraunhofer Institut IIS/EAS, Dresden, sowie die Lenord+Bauer & Co. GmbH, Oberhausen.

Literatur

[1] VDMA-Informationsveranstaltung: *Intelligenter Produzieren*. Frankfurt, 7. Nov. 2008.

[2] Schlecht, B. (Hrsg): *Dresdner Maschinenelemente Kolloquium* – DMK 2009. Dresden, 24./25. November 2009, Proceedings, TUDpress, 2009.

[3] Enge-Rosenblatt, O.; Wolf, G.; Reichert, T.: *Modellbasierter Entwurf von Signalverarbeitungs- und Klassifikationsalgorithmen für die Fehlerdiagnose in Maschinen und Anlagen* In: Statustagung „KMU-innovativ" IKT. Darmstadt, 17./18. November 2008. Tagungsunterlagen.

[4] Ming Liu: *Dynamisches Verhalten hydrostatischer Axialkolbengetriebe*. PhD thesis, Institute Product and Service Engineering, Ruhr-Universität Bochum, 2001.

[5] Liang Chen: *Model-based fault diagnosis and fault-tolerant control for a nonlinear electro-hydraulic system*. PhD thesis, TU Kaiserslautern, 2010.

[6] Fritzson P.: *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. IEEE Press, 2004.

Entwicklung eines integrierten RSSI-basierten Lokalisierungssystems für drahtlose Sensornetzwerke

Elena Chervakova
elena.chervakova@imms.de
Institut für Mikroelektronik- und Mechatronik-Systeme gemeinnützige GmbH
Ehrenbergstraße 27, D-98693 Ilmenau

Kurzfassung

Dieser Beitrag befasst sich mit den praktischen Aspekten der Entwicklung einer RSSI-basierten Ortung und Identifikation von Personen für nutzerbezogene Datenerfassung - von Messdatenerfassung bis zur Visualisierung von Ergebnissen. Dabei werden als wichtige Aspekte vor allem Kosten, Integration in übergeordnete Systeme und geringer Installationsaufwand in Betracht gezogen. Hierfür wurden verschiedene Messungen zur Untersuchung von Signalausbreitungseigenschaften durchgeführt. Es wird gezeigt, wie der Einsatz von Nachbearbeitungsalgorithmen zu einer erheblichen Fehlerminimierung führt.

Einleitung

Eine Vielzahl von Wireless Sensor Network (WSN) Anwendungen erfordern für unterschiedliche Zwecke die Positionsbestimmung eines Teilnehmers. Das Tracking eines mobilen Objekts stellt eine der anschaulichsten, aber auch schwierigsten Anwendungen für drahtlose Sensornetzwerke dar. Personenlokalisierung und Monitoring innerhalb eines Gebäudes werden zunehmend unverzichtbar und finden Anwendung in vielen Bereichen, beispielweise im Gesundheitswesen oder im Maschinenbau. Im Moment existiert ein breites Spektrum von Lokalisierungsverfahren für Sensornetzwerke, die sich durch die zugrunde liegende Hardware, das eingesetzte Messverfahren und die anwendungsspezifischen Anforderungen unterscheiden. Man kann jedoch keine ultimative Lösung nennen. Die Vielfältigkeit von Anwendungsszenarien erfordert spezifische Lösungen für den konkreten Einsatzfall. Aus diesen lassen sich die Anforderungen an Lokalisierungssysteme ableiten [1]. Darüber hinaus sollen Lokalisierungslösungen für Sensornetzwerke auch typische WSN-Anforderungen, wie Energieeffizienz und Skalierbarkeit, nicht aus dem Auge verlieren.

Gegenwärtig werden in der Gebäudeautomation festinstallierte Sensoren eingesetzt, um Daten für die Steuerung des Klimas im Gebäude zu liefern [2][3]. Durch Einsatz spezieller Algorithmen für drahtlose Sensornetzwerke kann in einem solchem Anwendungsszenario eine Personenlokalisierung durchgeführt werden.

Das entworfene Lokalisierungssystem den „BASe-Loc“ wird in ein Sensornetzwerk für Gebäudeautomation integriert und nutzt herkömmlichen Netzwerktraffic zur Lokalisierung durch Distanzmessung zu festinstallierten Sensoren.

Anforderungen an das System

Das Sensornetzwerk für Gebäudeautomation BASe-Net [4] mit Cluster-Tree Netzwerktopologie (Abbildung 1) besteht aus sog. *Subnodes* – Endknoten, die Umgebungsvariablen wie Temperatur, Luftfeuchte u.ä. messen und an entsprechenden Router (Cluster-Head) weiterleiten – und *Routern* – routingfähigen Cluster-Heads, die Information von Subnodes sammeln und ggf. über mehrere Hops zu eine Sammelstelle (Gateway) weiterleiten.

Für eine nutzerbezogene Datenerfassung sollte das Netzwerk um die Fähigkeit der Ortung und Identifikation von Personen erweitert werden. Dabei sollte die Lokalisierung keinen zusätzlichen Kosten-, Installations- und Systemintegrationsaufwand mit sich bringen. Mit diesen Anforderungen (Abbildung 2) wurde die Signalstärkemessung als Basis für das Lokalisierungssystem ausgewählt.

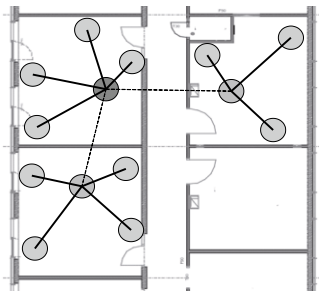


Abbildung 1 BASe-Net Netzwerktopologie

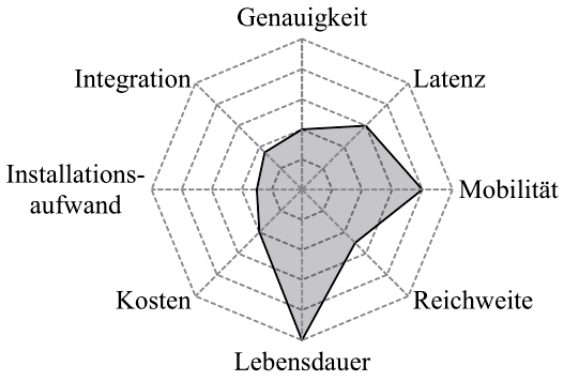


Abbildung 2 Anforderungen an das Lokalisierungssystem

Die Signalstärke (*RSSI*) wird beim Empfänger gemessen und in Beziehung mit der Distanz zum Sender gestellt. Aus mindestens drei Distanzen kann mittels Multilateration die Position des Empfängers berechnet werden. RSSI-Lokalisierung zeichnet sich durch leichte Implementierung und Integration, keine Notwendigkeit für zusätzliche Hardware und Verfügbarkeit auf den meisten Plattformen aus. Außerdem kann normaler Traffic des Sensornetzwerks für Signalstärkemessungen, sowie die Netzwerkinfrastruktur für das Weiterleiten der Daten verwendet werden.

Systemarchitektur

Ein typisches Lokalisierungssystem besteht aus drei Komponenten [5]:

- Erfassung von Eingangsparametern und Vorverarbeitung
- Berechnung der Position aus Eingangsparametern
- Anpassung der berechneter Position sowie des Gesamtbildes durch Einführung neuer Position

Es müssen Entscheidungen bezüglich der Auswahl von Messtechniken zur Ermittlung von Eingangsparametern, der Systemarchitektur, und der Algorithmen zur Abschätzung der Position getroffen werden [6].

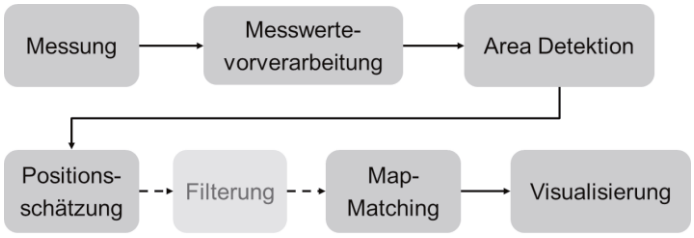


Abbildung 3 Komponenten des Lokalisierungssystems "BASe-Loc"

In dem BASe-Loc (Abbildung 3) soll ein Sensormodul, das eine Person bei sich trägt (ein *tag*), lokalisiert werden. Zunächst wird die Signalstärke entsprechend Alg. 1 gemessen. Nachfolgend werden auf der Serverseite die Messwertevorverarbeitung (siehe Alg. 2) und anschließend die Positionierung durch sog. *Area Detektion*, Positionsschätzung und Kartenanpassung (*Map Matching*) durchgeführt. Weiterhin können Lokalisierungsdaten visualisiert werden.

Algorithmus 1. Signalstärkemessungen

Eingabe: $M\{m_n\}$ Messreihe, RSSI Messwert von Beacon i $m_i := (b_i, rssi_{b_i})$, Anzahl der Messwerte in der Messreihe n, *sliding window* für Messung w_m.
Beginn:
$n = 0$
solange empfange Pakete von Beacons ∈ Netzwerk **tue**
    solange $n < 3$ **oder** $t \geq w_m$ **tue**
        Messe Signalstärke $RSSI_n$ von Beacons ∈ Netzwerk
        n=n+1
    ende
    wenn kein Router verfügbar **dann**
        solange kein Router gefunden **tue**
            suche Router
            melde sich als *Subnode* an
        ende
    sonst
        sende Paket mit Messwerte $M\{m_n\}$ an Router
    ende
ende

Algorithmus 2. Datenauswahl für Positionierung

Eingabe: Paket mit Messwerte $M\{m_n\}$, *sliding window* für Positionierung w_p, Menge der Messwerten in *sliding window* $S\{m_j\}$, minimale Anzahl der Messwerte für Lokalisierung k, Paketzeitstempel t_p.
Beginn:
extrahiere Daten $M\{m_n\}$ aus dem Paket mit dem Zeitstempel t_p
vereinige (M, w_p) in w_p
für alle $m_i \in w_p$ **tue**
    wenn existieren mehr als 1 $Beacon\ b$ mit ID i **tue**
        lösche m_i mit älterem Zeitstempel
    ende
ende
für alle $m_i \in w_p$ **tue**
    wenn Zeitstempel t_i des m_i älter als Zeitfenster w_p ist **tue**
        lösche m_i
    ende
ende
wenn $S\{m_j\} > k$ **tue**
    sende $S\{m_j\}$ an Positionierungsmodul
ende

RSSI Model

Zur Abbildung der Signalstärke auf die Distanz zwischen Sender und Empfänger wurden Charakteristika der Signalausbreitung sowie Einflussfaktoren auf die Messung untersucht. Signalstärkemessung wird von den meisten Funkmodulen unterstützt, unterscheidet sich aber je

nach Funkmodul in Genauigkeit und Auflösung (siehe Tabelle 1). Messergebnisse werden außerdem durch Antenneneigenschaften und Sensitivität des Empfängers beeinflusst.

Tabelle 1 RSSI bei unterschiedlichen Transceivern

Funkchip	Freq.Band	RSSI Auflösung
CC2420	2,4 GHz	1 dBm
CC1000	868 MHz	RSSI direkt als Spannung messbar
RF230	2,4 GHz	3 dBm
Si4432	868 MHz	0,5 dBm

Grundsätzlich hängt die Signalstärke bzw. Empfangsleistung P_E von der vom Sender ausgestrahlten Leistung P_S, Wellenlänge λ, Gewinn der Sendeantenne G_S, Gewinn der Empfangsantenne G_E und Distanz r ab [7]:

$$P_E = P_S \left(\frac{\lambda}{4\pi r}\right)^2 \cdot G_S \cdot G_E \qquad (1)$$

Theoretisch wird P_E mit steigender Distanz r monoton sinken. Jedoch haben im realen Fall Effekte wie Abschattung, Reflexion, Streuung und Beugung bei der Signalausbreitung v.a. im Indoor-Bereich erheblichen Einfluss auf die gemessene Signalstärke [8]. Auch Ausrichtung und Gehäuse des Funkmoduls können die Messung stark beeinflussen.

Zur Ermittlung der Kennlinie zwischen P_E und r wurden zahlreiche Messungen mit Zielhardware durchgeführt (Abbildung 4). Es hat sich gezeigt, dass die eindeutige Zuordnung von RSSI Werten zu der Distanz unmöglich und, bei sich schnell verändernden Umgebungen, nicht sinnvoll ist. Infolgedessen wurden jedem möglichen RSSI Wert ein Bereich bzw. minimale und maximale Distanz zugeordnet. Außerdem wurde jeder Messwert (RSSI) gewichtet, wobei das Gewicht der Wahrscheinlichkeit, dass der *tag* sich im zugeordneten Bereich befindet, entspricht.

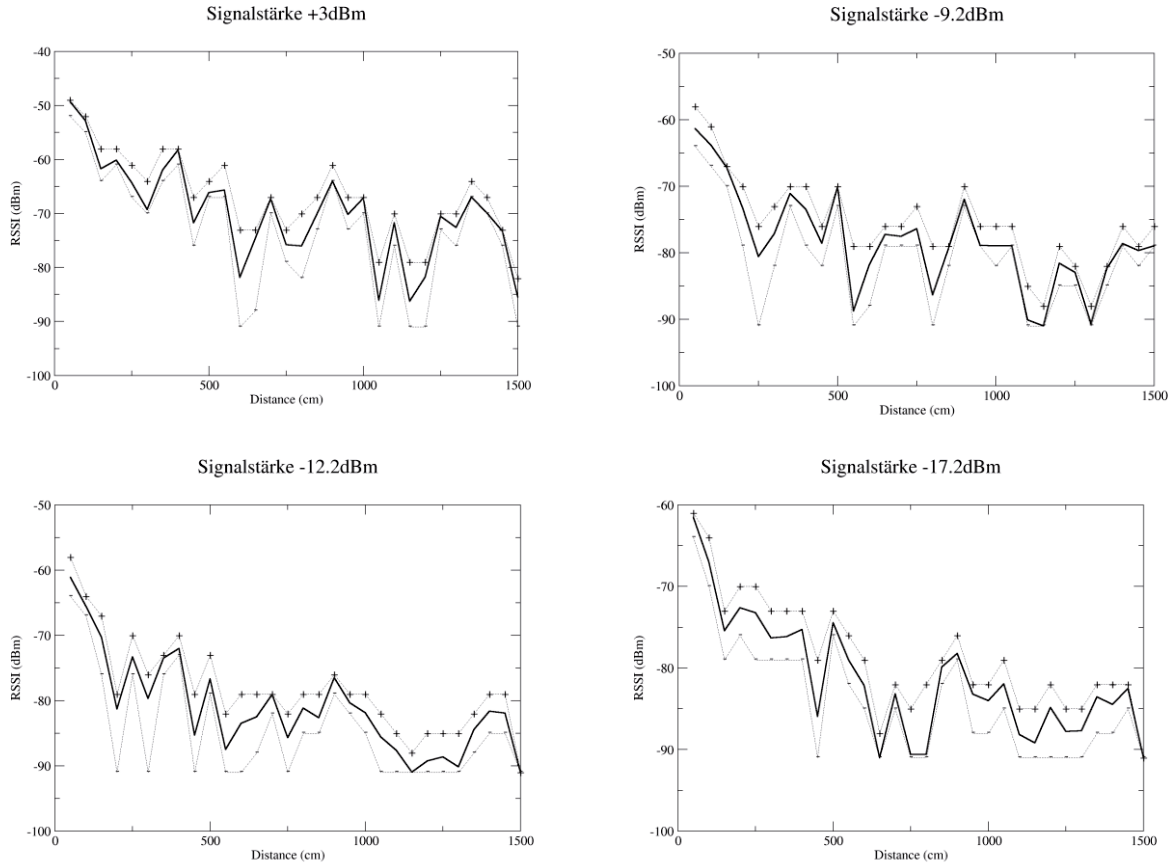


Abbildung 4 RSSI Messungen mit 2.4GHz IEEE 802.15.4 Transceiver AT86RF230 im Gebäude

Das RSSI-Modell wird im XML-Format beschrieben und beim Start der Lokalisierungssoftware geladen, es wird durch eine Menge von allen möglichen RSSI Messwerten $\{r_i\}$ beschrieben. Ein RSSI Messwert $r := (d_{min}, d_{max}, weight)$ ist ein Tupel mit minimalen und maximalen Distanzen d_{min} und d_{max}, und dem Gewicht $weight$, wobei $d_{min}, d_{max}, w \in \mathbb{R}^+$.

Positionsbestimmung

Im ersten Schritt wird der Bereich (sog. *Area*), in dem sich das *tag* befindet, detektiert. Das detektierte *Area* wird weiterhin für Positionsbestimmung und für Kartenanpassung (*Map Matching*) verwendet.

Der Datensatz der Messwerte $M\{m_i\}$ wird durch Tupel $m_i := (b_i, rssi_{b_i})$ mit Beacon-ID b_i und dazugehörigem RSSI-Wert $rssi_{b_i}$ gebildet. Zunächst werden alle verfügbaren Areas gewichtet. Für jeden m_i aus M

$$\forall b \in Area_j \; : \; w_{Area_j} = \sum weight_{rssi} \tag{2}$$

mit $weight_{rssi}$ entsprechend RSSI Modell. Die Auswahl des aktuellen Area $Area_{new}$ ist vom Gewicht w_{Area_j} und dem im letzten Schritt bestimmten Area abhängig.

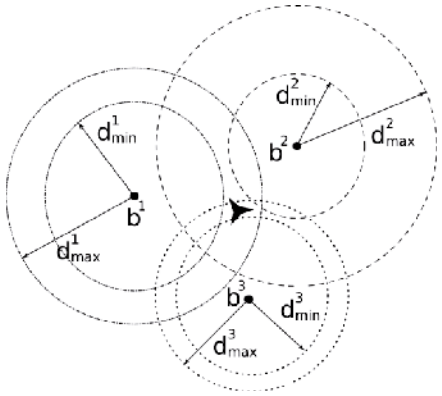


Abbildung 5 Positionsbestimmung durch Multilateration

Die Positionsschätzung kann nach RSSI-Modell entweder durch das Multilaterationsverfahren (Abbildung 5) oder durch den gewichteten Mittelpunkt *nach (3)* berechnet werden.

$$\forall b_i \in M\{m_i\} : pos(x,y) = pos\left(\frac{\sum x_{b_i} \cdot w_{b_i} \cdot w_{Area_i}}{\sum w_{b_i} \cdot w_{Area_i}}, \frac{\sum y_{b_i} \cdot w_{b_i} \cdot w_{Area_i}}{\sum w_{b_i} \cdot w_{Area_i}}\right) \tag{3}$$

Anschließend wird die Positionsanpassung gemäß der Koordinaten von $Area_{new}$ durchgeführt.

Ergebnisse

Die Testergebnisse des BASe-Loc Systems in einem Bürogebäude (Abbildung 6) haben gezeigt, dass eine eindeutige Raumidentifizierung möglich ist. Die Lokalisierungsgenauigkeit schwankt sehr stark je nach Position aufgrund der dynamischen Umgebung sowie durch starke Einflüsse der Mehrwegeausbreitung.

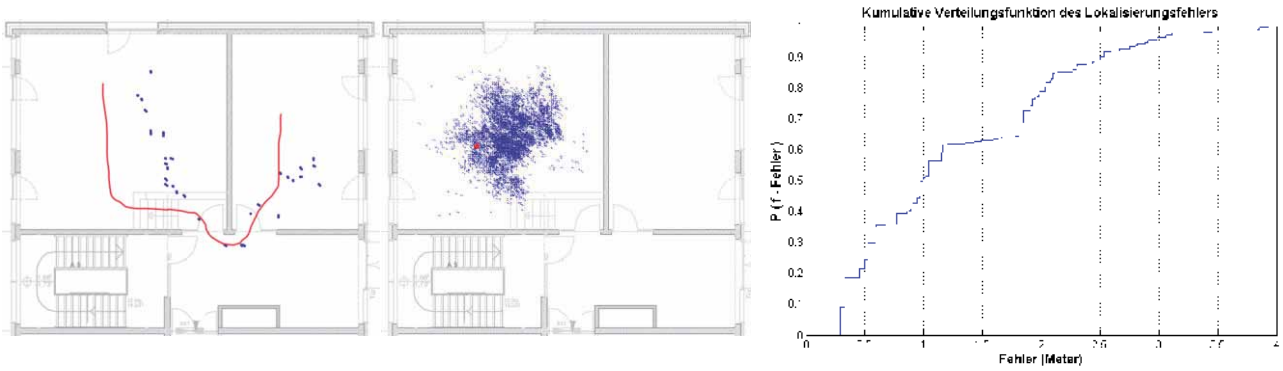


Abbildung 6 Ergebnisse von Traking und statischer Lokalisierung

Zusammenfassung und Ausblick

In dem Beitrag wurden Entwurf und Evaluierung eines Lokalisierungssystems vorgestellt. Das System zeichnet sich durch gute Integrierbarkeit, geringe Kosten und geringen Installationsaufwand aus. Der modulare Aufbau bietet viele Möglichkeiten zum Austausch oder zur Integration neuer Algorithmen für bspw. Positionsschätzung oder Kartenanpassung. Die Messtechnik kann ebenfalls ausgetauscht oder erweitert werden. Die aktuellen Arbeiten am BASe-Loc beschäftigen sich weiterführend mit der Einbindung zusätzlicher Sensorik in das System sowie mit Implementierung und Vergleich weiterer Positionierungsalgorithmen.

Literatur

[1] E. Chervakova, T. Rossbach, “Techniques and systems for indoor localization in wireless sensor networks,” in Proceedings of the 54th International Scientific Colloquium (IWK), Ilmenau, Germany, 2009.

[2] E. Chervakova, D. Klan, and T. Rossbach, “Energy-optimized sensor data processing,” in Proceedings of the 4th European Conference on Smart Sensing and Context (EuroSSC), 2009.

[3] S. Engelhardt, E. Chervakova, and T. Rossbach, “Wireless Sensornetzwerke mit Multisensorik in der Gebäudeautomation,” in Tagungsband 11. Wireless Technologies Kongress. Stuttgart, 29. – 30. September 2009.

[4] M. Götze, T. Rossbach, A. Schreiber, S. Nicolai, H. Rüttinger, “Distributed in-house metering via self-organizing wireless networks,” in Proceedings of the 55th International Scientific Colloquium (IWK), Ilmenau, Germany, 2010.

[5] N.Bulusu and S.Jha, “Wireless Sensor Networks: A Systems Perspective”, Artech House, 2005.

[6] K. W. Kolodziej and J. Hjelm, Local Positioning Systems: LBS Applications and Services. Taylor & Francis Group, 2006.

[7] Garg Vijay Kumar, Wireless communications and networking, Elsevier Morgan Kaufmann, 2007.

[8] Schiller J.: Mobilkommunikation, Pearson Studium, ISBN 10-3-8273-7060-4, 2003

Entwurf und Implementierung eines adaptiven Prozessors in einer funktionalen Hardwarebeschreibungssprache

Stefan Schulze
NEMONOS GmbH
Bernstorffstraße 99
22767 Hamburg

Sergei Sawitzki
FH Wedel
Feldstraße 143
22880 Wedel

Kurzfassung: Seit der Vorstellung von VHDL und Verilog-HDL vor ca. 25 Jahren ist ein kontinuierlicher Trend der Verlagerung von HDL-Entwurfsbeschreibungen in immer höhere Abstraktionsebenen festzustellen. Auch Erweiterungen einiger funktionaler Programmiersprachen ermöglichen mittlerweile Beschreibungen von Hardware-Strukturen. Der vorliegende Beitrag fasst die Ergebnisse und Erfahrungen aus einem Projekt zur Umsetzung eines adaptiven Prozessors mit Hilfe der Hardwarebeschreibungssprache Lava HDL zusammen. Die für einen Altera Cyclon II FPGA ermittelten Synthese-Ergebnisse sind vergleichbar mit denen eines VHDL-Handentwurfs, wobei die Lava-Beschreibung wesentlich kompakter ist. Die vorgestellte Prozessor-Architektur wird durch einen ebenfalls in Lava beschriebenen Coprozessor erweitert, der auf frei verbleibenden FPGA-Ressourcen implementiert wird.

1 Einleitung und Motivation

Überwiegende Mehrheit digitaler Systeme wird heutzutage mit Hilfe von Hardwarebeschreibungssprachen (HDL) entworfen. Deren bekannteste Vertreter — Verilog HDL und VHDL — können bereits eine über 25-jährige Entwicklungsgeschichte vorweisen. In den letzten Jahren wurden dabei stets Bemühungen unternommen, das Abstraktionsniveau der Hardwarebeschreibungen zu erhöhen. Das ist zum einen durch die aktuellen Erweiterungen von bestehenden Sprachen (z.B. VHDL-2008, SystemVerilog, Bluespec SystemVerilog) aber auch durch die Einführung neuer Sprachen (z.B. SystemC) erkennbar. Dabei werden in erster Linie solche altbekannten Probleme der "traditionellen" HDL wie mangelnde Parametrisierungs- und Verifikationsfähigkeiten adressiert. Ein weiterer innovativer Ansatz zur Lösung dieser Probleme ist aus dem Bereich der Softwaretechnik bekannt und kann grob unter dem Oberbegriff der "funktionalen Hardwarebeschreibungssprachen" zusammengefasst werden. Der grundlegende Unterschied zu den imperativen Sprachen, die eine Abfolge von Rechenschritten angeben ($\hat{=}$ "wie muss etwas berechnet werden") besteht lapidar formuliert darin, dass die Problemlösung in Form von (rekursiven) Funktionen angegeben wird ($\hat{=}$ "was muss berechnet werden"). Die ersten Versuche, Hardware mit Hilfe von funktionalen Sprachen zu beschreiben und zu synthetisieren (z.B. μFP [1], Ruby [2], Reflect [3]) erfolgten parallel mit der Etablierung von Verilog HDL und VHDL, stießen jedoch bestenfalls auf rein akademisches Interesse [4]. In den letzten Jahren wurden funktionale HDL jedoch zunehmend auch in komplexen Entwurfsprojekten sowohl im akademischen als auch im industriellen Bereich eingesetzt. Der vorliegende Beitrag fasst die Ergebnisse und Erfahrungen aus einem Projekt zur Umsetzung eines adaptiven Prozessors mit Hilfe der auf Haskell basierenden funktionalen Hardwarebeschreibungssprache Lava HDL zusammen.

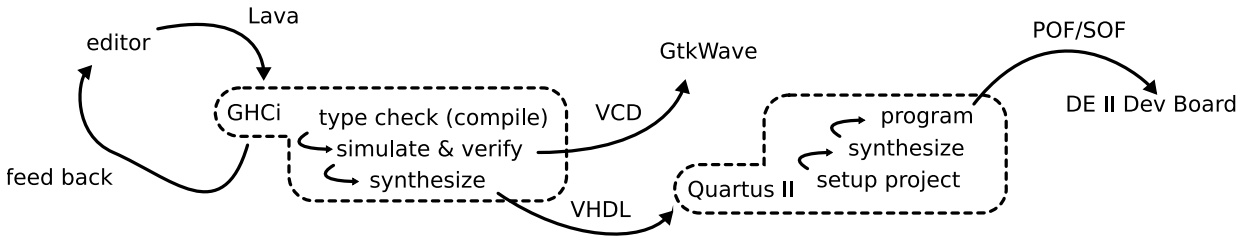


Abbildung 1: Entwurfsfluss

Einige bekannte Arbeiten befassen sich mit der Abbildung von Lava-Beschreibungen auf verschiedene Xilinx FPGA-Familien [5, 6]. Dabei wird Lava um Ausdrucksmittel zur Nutzung architekturspezifischer Eigenschaften von Xilinx (CLB Strukturen, Platzierungsvorgaben u.ä.) erweitert. Auch wenn die Ergebnisse viel versprechend erscheinen, so fällt dennoch auf, dass nahezu alle vorgestellten Entwürfe eine starke reguläre Struktur aufweisen. Diese Einschränkung wirft die Frage auf, ob Lava ebenfalls beim Entwurf komplexer Systeme mit signifikantem Anteil an irregulärer Logik vorteilhaft eingesetzt werden kann und dient somit als Hauptmotivation für die vorliegende Arbeit. Weiterhin wurden die Ergebnisse des Reduceron-Projektes ausgewertet [7], bei dem es sich um einen anwendungsspezifischen Prozessor für Graphreduktion handelt (der jedoch nicht als Universalprozessor einsetzbar ist).

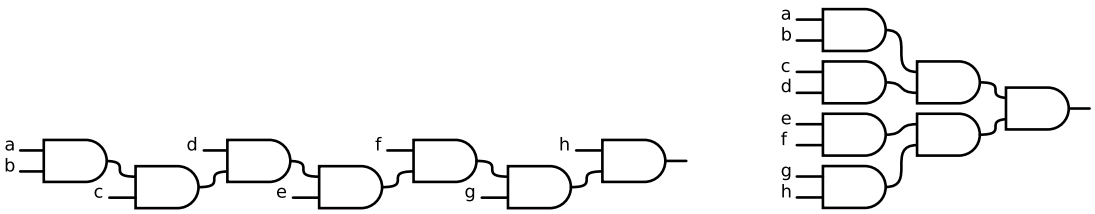
Im weiteren erfolgt eine kurze Einführung in die Grundlagen und Konzepte des funktionalen Hardware-Entwurfs mit Lava HDL, an die eine Vorstellung einer einfachen 32-Bit RISC-Architektur Pipo — des Versuchsobjekt dieser Studie — anknüpft. Anschließend wird deren komplette Umsetzung in Lava HDL inklusive der im Rahmen der vorliegenden Arbeit entwickelten, auf Vereinfachung der Prozessorbeschreibung ausgerichteten Erweiterungen diskutiert. Ein Teil des Beitrags ist der automatisierten Verifikation der Prozessorkomponenten mit Hilfe von entsprechenden Haskell-Bibliotheken sowie der Beschreibung des Entwurfsflusses gewidmet. Einige Ergebnisse müssen aus Platzgründen in stark zusammengefasster Form präsentiert werden. Implementierungsdetails können weiteren öffentlich verfügbaren Quellen entnommen werden [8, 9].

2 Hardwareentwicklung mit Lava-HDL

Lava ist genau betrachtet eine Bibliothek der funktionalen Programmiersprache Haskell und keine eigenständige Sprache. Dennoch hat sich in den Fachkreisen die Bezeichnung Lava-HDL durchgesetzt. Daher wird Lava auch in diesem Beitrag als höhere Hardwarebeschreibungssprache bezeichnet. Lava-HDL ist ein Beschreibungsmittel von rein struktureller Natur, d.h. sie besitzt keine “eingebauten” Ausdrucksmittel für Verhaltensbeschreibungen, wenngleich mehrere Erweiterungsbibliotheken eine Erstellung von Verhaltensbeschreibungen ermöglichen [10]. Die ursprüngliche Version von Lava wurde vor fast 15 Jahren entwickelt und erfuhr seitdem zahlreiche Erweiterungen [11, 12, 13].

Der wesentliche Teil der Entwicklungsumgebung ist der Haskell-Compiler bzw. Haskell-Interpreter. Dieser erzeugt bzw. interpretiert einen Code, von dem aus alle für den Entwurfsprozess notwendigen Schritte durch Aufruf von entsprechenden Funktionen gesteuert werden. Im Rahmen des Projektes wurden zum einen Impulsfolgediagramme im VCD-Format (als Ergebnis der funktionalen Simulation) und zum anderen synthesefähiger VHDL-Code erzeugt. Der letzte Schritt ist nur deswegen nötig, da momentan keine Entwicklungsumgebungen existieren, die eine direkte Hardware-Synthese von Lava-Code ermöglichen. Mit der generierten VHDL-Beschreibung können dagegen alle üblichen Synthese-Frontends genutzt werden. Im vorliegenden Projekt wurde Quartus II Software von Altera für eine prototypische Implementierung auf einem EP2C35 FPGA eingesetzt. Der Entwurfsfluss ist in Abbildung 1 skizzenhaft dargestellt.

Die Grundlage von Lava bilden die Definitionen von Basislogikzellen (Logikkonstanten, einfa-



(a) cascade `(-&-)` (b) tree `(-&-)`

Abbildung 2: Verschiedene Auswertungsfolgen des Ausdrucks $a \wedge b \wedge c \wedge d \wedge e \wedge f \wedge g$

che Gatter, Flipflops) sowie Beschreibungsmittel zu deren Verknüpfung:

```
low   :: Bit                -- Logic 0
high  :: Bit                -- Logic 1
inv   :: Bit -> Bit         -- Inverter
(<&>) :: Bit -> Bit -> Bit -- AND gate
(<|>) :: Bit -> Bit -> Bit -- OR gate
(<#>) :: Bit -> Bit -> Bit -- XOR gate
delay :: Bit -> Bit -> Bit -- D-flip-flop
```

Somit können beliebig komplexe Schaltnetze erzeugt werden, wobei Verbindungsmuster auch explizit spezifiziert werden können, wie die Abbildung 2 zeigt. Bei der Simulation von solchen Beschreibungen erfolgt eine rekursive Graphreduktion, bis ein Blatt-Knoten erreicht ist, dem ein konstanten Logik-Wert zugewiesen ist. Die Beschreibung von zustandsbehafteten Komponenten erfolgt mit Hilfe der Erweiterungsbibliothek `Recipe`. Diese ermöglicht eine Definition von speziellen "Zugriffpunkten" vom Typ `Reg`, die bei der Simulation und Erzeugung der Netzliste als Register fungieren. Die Erzeugung einer VHDL-Beschreibung erfolgt automatisch durch rekursiven Aufruf der Funktion `writeVHDL`, die lediglich den Namen der zu synthetisierenden Komponenten und ihrer Schnittstellen als Parameter übergeben bekommt.

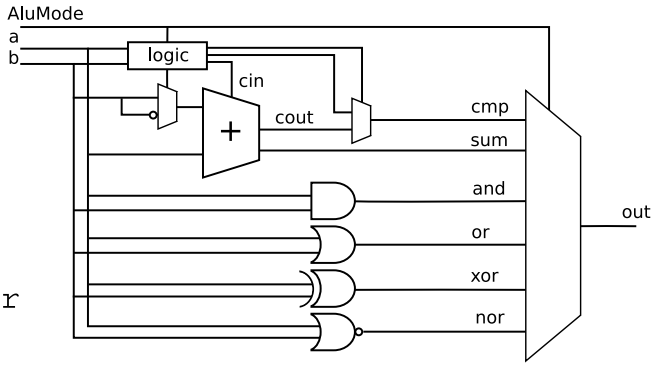
Ein großer Vorteil der Lava-Nutzung ist die Verfügbarkeit von zahlreichen Bibliotheken zur automatischen Verifikation, die in Form von *property based testing* erfolgt. Der Entwerfer spezifiziert die allgemeinen Einschränkungen sowie Sonderfälle, deren Einhaltung durch automatische (intelligente) Auswahl von Testfällen überprüft wird. Beispielsweise basiert die *QuickCheck*-Bibliothek auf Erzeugung von Testsätzen nach einem Zufallsmuster. Der Verifikationsvorgang bricht ab, wenn abweichendes Verhalten festgestellt oder eine spezifizierte Anzahl von Testsätzen generiert wurde. Bei *SmallCheck* erfolgt ein sukzessiver Ausbau von Testsätzen von ganz einfachen zu immer komplexeren. Dabei wird vorausgesetzt, dass eine fehlerhafte digitale Schaltung bereits für einfache Eingaben fehlerhafte Ausgaben produziert. Alle Funktionseinheiten des im nächsten Abschnitt vorgestellten Pipo-Prozessors wurden erfolgreich mit *SmallCheck* getestet.

3 Modellierung einer 32-Bit RISC Architektur

Bei der Wahl der zu implementierenden Architektur standen zwei Überlegungen im Mittelpunkt. Einerseits sollte der Prozessor wesentliche Merkmale von RISC-Architekturen (orthogonaler Befehlssatz, Pipeline, Universalregistersatz usw.) aufweisen, so dass nach Fertigstellung eine Testung komplexerer Anwendungen möglich ist. Andererseits dürfte die Komplexität des Entwurfs die (größtenteils aus dem akademischen Umfeld stammende) Werkzeuge nicht überfordern. Letztendlich konnte mit MIPS-Architektur ein guter Kompromiss gefunden werden, zumal für diese zu Referenzzwecken bereits eine öffentlich verfügbare VHDL-Implementierung *Plasma* existiert [14]. Die Beschreibung von wesentlichen Funktionseinheiten wie Befehlsdecoder, ALU, Registersatz usw. in Lava stellte dabei keine große Herausforderung dar, wie die Abbildung 3 zeigt.

```
alu :: Int -> ([Bit], ([Bit],[Bit])) -> [Bit]
alu n (f, (a,b)) = pick $
  [   isMode AluAdd
  -|- isMode AluSub --> sum
  ,   isMode AluLT
  -|- isMode AluLTU --> cmp
  ,   isMode AluOr  --> or
  ,   isMode AluAnd --> and
  ,   isMode AluXor --> xor
  ,   isMode AluNor --> neg or
  ]
  where
    isMode m = f#--m
    isUnsignedCmp = (msb a <--> msb b) -|- AluLTU
    cmp      = mux (isUnsignedCmp, (msb a, c))
             : zeroList (n-1)
    or       = a =|= b
    and      = a =&= b
    xor      = a =#= b
    doSub    = inv $ isMode AluAdd
    (sum, c) = adder (doSub, (a, b =#- doSub ))
```

(a) Lava-Code



(b) Synthetisierte Hardware

Abbildung 3: Arithmetisch-logische Einheit

Wesentlich komplexer ist dagegen die Zusammenfassung der Einheiten zur Verarbeitungspipeline, so dass ein resultierendes Modell korrekt das Verhalten des Prozessors bei der Abarbeitung der Befehle wiedergibt. Zu diesem Zweck wurde eine neue Lava-Bibliothek `PExpr` entwickelt, die eine Beschreibung einzelner Befehle entsprechend ihrer Wirkung auf zustandstragende Komponenten (Register, Befehlszähler) erlaubt. Zu diesem Zweck wurde eine Reihe von Bezeichnern zur Darstellung entsprechender Komponenten definiert. Der Code mit dem letztendlich die Befehle beschrieben werden ist in wesentlichen Teilen der in der MIPS-Spezifikation für einen gleichen Zweck verwendeten HDN *(hardware description notation)* ähnlich. So werden die Beschreibungen der Form

```
RD .=(RS :+: RT)
RD .=(RS :-: IMM)
```

entsprechend als ADD und SUBI Befehle korrekt umgesetzt. Auch eine Beschreibung von Makros wie z.B. das Kopieren von Registerinhalten in Form von `RD .= RS` ist möglich. Für Befehle mit Speicherzugriffen wird ein spezieller Bezeichner verwendet, der als Parameter die Adresse, auf die zugegriffen wird, sowie die Zugriffsbreite übergeben bekommt.

Nicht alle syntaktisch korrekten Beschreibungen stellen erlaubte Operationen dar, so darf ein *Immediate*-Wert z.B. nicht beschrieben sondern nur gelesen werden. Solche und ähnliche Einschränkungen werden automatisch bei der Erzeugung der Hardware-Struktur überprüft.

Durch den vorgestellten Ansatz der getrennten Modellierung von Funktionseinheiten und deren Verbindungsstruktur ist es relativ einfach, mit der selben Codebasis andere Prozessor-Instanzen vergleichbarer Architektur zu erzeugen. Die neuen Befehle müssen dabei lediglich mit Hilfe der `PExpr`-Bibliothek beschrieben werden, die Generierung von entsprechender Hardware erfolgt nahezu vollautomatisch. Zusätzlich (falls entsprechende Regeln vom Entwerfer vorgegeben sind) findet Konsistenzüberprüfung statt.

Tabelle 1: Zusammenfassung der Syntheseergebnisse

Vergleichskriterium	Pipo CPU	Plasma CPU	
Pipeline-Stufen	5	2	3
Logikzellen	2137	2453	2461
Registerzellen	381	452	568
Anzahl Codezeilen	1345	5341	
Speicher (bits)	2048	2048	
Eingebettete Multiplizierer	8	0	
Taktfrequenz [MHz]	60.26	19.49	38.68

Tabelle 2: Syntheseergebnisse des FFT-Coprozessors

Vergleichskriterium	Pipo mit radix-2 FFT	Pipo mit einzelner Butterfly	Pipo CPU ohne Coprozessor
Logikzellen	3325	2837	2137
Registerzellen	1396	794	381
Eingebettete Multiplizierer	40	16	8
Taktfrequenz [MHz]	57.12	56.33	60.26

4 Zusammenfassung der Ergebnisse

Bei einer prototypischen Umsetzung auf einem Cyclone II FPGA der Firma Altera konnte eine Taktfrequenz von über 60 MHz bei sehr moderatem Ressourcenverbrauch (unter 7% der Logikzellen eines EP2C35 FPGA) erreicht werden. Die Tabelle 1 fasst die Syntheseergebnisse im Vergleich zur bereits erwähnten Plasma-Implementierung zusammen. Die letztere ist komplett in VHDL beschrieben und besitzt keinen Hardwaremultiplizierer, dafür aber eine Reihe von zusätzlichen Komponenten, z.B. eine DMA-Einheit. Die Unterschiede bei der erreichbaren Taktfrequenz sind hauptsächlich auf die Pipeline-Tiefe zurück zu führen. Anwendungstests erfolgen mit Hilfe eines eigen entwickelten Assemblers, der zur Übersetzung einfacher Anwendungskernel aber auch komplexerer Programmstrukturen genutzt wird. Dabei wurden sowohl alle Befehle einzeln als auch möglichst viele verschiedene Kombinationen verschiedener Befehle getestet.

Zur gezielten Ausnutzung von durch den Prozessor nicht belegten FPGA-Ressourcen wurde eine Coprozessor-Schnittstelle definiert und umgesetzt, über die Teile der Anwendung direkt auf Hardware abgebildet werden können. Diese ist speziell darauf ausgerichtet, die aus einem externen Speicher kommenden Daten für eine parallele Bearbeitung vorzuhalten. Die Kommunikation mit dem Hauptprozessor erfolgt mittels Standardmechanismen aus der MIPS-Spezifikation.

Als Anwendungsbeispiel wurde die schnelle Fourier-Transformation (FFT) gewählt. Zum Vergleich wurden zwei verschiedene Coprozessor-Instanzen implementiert. Die erste Variante (Radix-2) berechnet eine Transformation für 4 komplexe 16-bit Eingabewerte, während die zweite eine einfache Butterfly-Operation implementiert. Die entsprechende Lava-Beschreibung ist nur ca. 60 Codezeilen lang. Der Programmcode besteht aus einer Hauptschleife und einer Interrupt-Service-Routine. Die Syntheseergebnisse für den durch einen Coprozessor erweiterten Pipo-Prozessor sind in der Tabelle 2 zusammengefasst. Dabei ist die Taktfrequenz für den Gesamtentwurf angegeben, ein einzelner Coprozessor kann mit 180 MHz getaktet werden. Die Umsetzung der Prozessorerweiterung ist unabhängig von der darunterliegenden Hardware-Plattform und kann somit auch auf einem anderen FPGA bzw. auch als ASIC erfolgen.

Der wesentliche Beitrag der vorliegenden Arbeit kann wie folgt zusammengefasst werden:

- Eine (nach Kenntnisstand der Autoren) erstmalige Umsetzung eines RISC-Prozessors in Lava HDL. Die Syntheseergebnisse bescheinigen dem Entwurf eine mit VHDL-Handentwurf vergleichbare Leistungsfähigkeit bei wesentlich kompakterem (ca. 1/4) Codeumfang.
- Vorstellung einer generischen Schnittstelle zur anwendungsspezifischen Prozessorerweiterung und deren praktischer Einsatz am Beispiel eines Coprozessors für schnelle Fourier-Transformation.
- Erweiterung des Sprachumfangs von Lava HDL um zusätzliche Bibliotheken zur vereinfachten Beschreibung des Befehlssatzes sowie um drei Back-Ends zur Generierung von VHDL-Code und Visualisierung der Simulationsergebnisse.

Das vorliegende Projekt zeigt, dass funktionaler Ansatz für die Beschreibung komplexer irregulärer Systeme geeignet ist und von einem erfahrenen Entwerfer vorteilhaft eingesetzt werden kann. Die gegenwärtigen Probleme beim praktischen Einsatz sind größtenteils auf mangelhafte Umsetzung einiger Aspekte von Lava HDL bzw. auf die Qualität der Entwurfswerkzeuge und nicht auf das Konzept des funktionalen Hardwareentwurfs als solches zurück zu führen.

Literatur

[1] M. Sheeran, *μFP, an algebraic VLSI Design Language*, PhD thesis, Oxford University, 1983

[2] G. Jones and M. Sheeran, *Circuit Design in Ruby*, Formal Methods for VLSI Design, Elsevier Science Publishers, 1994, pp. 13–70

[3] J. Grundy, T. Melham, and J. O'Leary, *A Reflective Functional Language for Hardware Design and Theorem Proving*, Journal of Functional Programming, Vol. 16(2), Cambridge University Press, 2006, pp. 157–196

[4] M. Sheeran, *Hardware Design and Functional Programming: a Perfect Match*, Journal of Universal Computer Science, Vol. 11(7), 2005: 1135–1158

[5] S. Singh and P. James-Roxby, *Lava and JBits: From HDL to Bitstream in Seconds*, in Proceedings of FCCM, 2001, pp. 91–100

[6] S. Singh, *Xilinx Lava Hardware Description Language*, 1. October 2009, URL: `http://www.raintown.org/lava`

[7] M. Naylor and C. Runcimann, *The Reduceron Research Project*, 2008–2010, URL: `http://www.cs.york.ac.uk/fp/reduceron/`

[8] S. Schulze, *Design, Implementation and Verification of a Processor with a Functional Hardware Description Language*, Diplomarbeit, FH Wedel, March 2010

[9] S. Schulze, S. Sawitzki, *Design, Implementation and Verification of an Adaptable Processor in Lava HDL*, Proceedings of 7th International Symposium on Applied Reconfigurable Computing, Belfast, UK, 23–25 March 2011

[10] M. Naylor, *Hardware-Assisted and Target-Directed Evaluation of Functional Programs*, PhD thesis, University of York, 2008

[11] P. Bjeese, K. Claessen, M. Sheeran, and S. Singh, *Lava: Hardware Design in Haskell*, in Proceedings of ICFP, ACM SIGPLAN, 1998

[12] K. Claessen and D. Sands, *Observable Sharing for Functional Circuit Description*, in Proceedings of Asian Computer Science Conference (ASIAN), Springer 1999

[13] K. Claessen, *An Embedded Language Approach to Hardware Description and Verification*, Lic. thesis, Chalmers University of Technology, August 2000

[14] S. Rhoads, *Plasma — most MIPS I(TM) opcodes*, URL: `http://plasmacpu.no-ip.org`

Analyse des Trace-Speicherbedarfs in eingebetteten Prozessorkernen

Steffen Köhler, Robert Ramm, Rainer G. Spallek

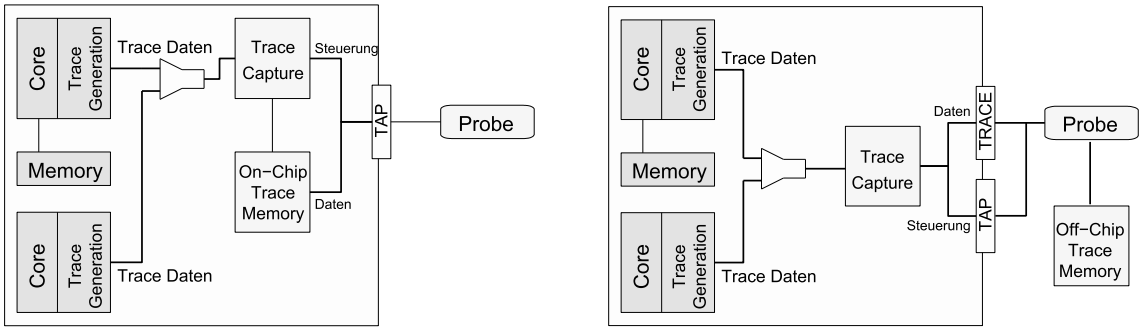
Institut für Technische Informatik
Technische Universität Dresden
D-01062 Dresden
{stk,rgs}@ite.inf.tu-dresden.de

Kurzfassung

Im Rahmen der Hardware- und Softwareanalyse in eingebetteten Prozessorkernen gewinnen On-Chip-Trace-Einheiten zunehmend an Bedeutung, insbesondere wenn eine geringe Rückwirkung der Analyseverfahren auf das Laufzeitverhalten von zeitkritischen Anwendungen absolut notwendig ist (z.B. Automotive Control). Es existieren verschiedene Ansätze, die mittels komplexer, programmierbarer Trigger-Blöcke die hardwaregestützte Verifikation spezieller Systemeigenschaften gestatten. Ziel dieser On-Chip-Trace-Einheiten ist dabei vor allem eine Reduktion des Trace-Datenstromes, da Hochgeschwindigkeitsschnittstellen zu dessen Ausgabe nicht ohne erhebliche Chip-Zusatzkosten realisierbar sind. Aus diesem Grund werden meist Triggereinheiten mit Trace-Pufferspeichern und Schnittstellen kombiniert, die mit einem vertretbaren Kostenaufwand für die anstehenden Verifikationsaufgaben realisierbar sind. Während die Realisierung der Triggereinheiten im Wesentlichen von der Verifikationsaufgabe selbst und die Schnittstelle von der verwendeten Halbleitertechnologie vorbestimmt werden, ist die Größe des Trace-Speichers abhängig vom geforderten Informationsumfang der Trace-Aufzeichnung. Die Notwendigkeit von verlustfrei zusammenhängend gespeicherten Trace-Informationen resultiert aus der Forderung der Rekonstruierbarkeit des Programm- und Datenflusses, um ein mittels Trace beobachtetes Verhalten im Kontext von Hochsprachenkonstukten nachvollziehen zu können. Es besteht daher ein direkter Zusammenhang zwischen der Trace-Speichergröße und der Rekonstruierbarkeit des Programm- und Datenflusses, der Gegenstand dieses Beitrages ist. Die Rekonstruktion umfasst dabei die Ermittlung von Registerinhalten durch Befehlssatzsimulation, wobei letztere auf der Basis von Load-/Store-Trace und Branch-Befehlstrace durchgeführt wird. Ziel ist die vollständige Ermittlung aller Registerbelegungen innerhalb eines Intervalls, für das Trace-Daten vorliegen. Damit diese Vollständigkeit erreicht wird, muss das angestrebte Trace-Intervall und damit die Trace-Speichergröße aufgrund der vorliegenden Maximallänge der Registerlebensspannen eine applikationsspezifische Mindestgröße aufweisen. Die Ermittlung dieser Größe wird anhand einer vereinfachten Variante des EEMBC Industrial/Automotive Benchmarks demonstriert, der als eine repräsentative Applikationslast für den Bereich Automotive Control angesehen werden kann. Hauptziel der Evaluation der Trace-Speichergröße ist der applikationsspezifische Zuschnitt des Tracespeichers, um einerseits einen hohen Rekonstruktionsgrad zu erreichen, andererseits die Kosten für die Chip-Fläche des Trace-Speichers gering zu halten.

1 Einleitung

Die Ermittlung von Software-Laufzeitinformationen mittels Hardware-Trace ist für bestimmte Analyseaufgaben in eingebetteten Systemen unumgänglich, da nur auf diese Weise eine nicht beeinflussende Systembeobachtung möglich ist. Ein typisches Anwendungsszenario umfasst dabei die Spezifikation der Beobachtungsaufgabe und deren teilautomatisierte Umsetzung durch programmierbare On-Chip-Trigger-Einheiten, die Erfassung, Kompression, Aufzeichnung und Ausgabe des Befehls- und Datenflusses sowie die darauf basierende anschließende Rekonstruktion des Befehls- und Datenflusses der ursprünglichen Programmrepräsentation (Hochsprache).
Die Trace-Erfassung mittels Hardware erfolgt dabei in mehreren Stufen, die im Wesentlichen Ereignistrigger, Zusammenführung und Vorverarbeitung der Triggerereignisse sowie deren Zwischenspeicherung und Ausgabe umfassen. Während die Architektur der On-Chip Triggereinheiten und deren Zusammenführung



(a) On-Chip Trace-Speicher

(b) Off-Chip Trace-Speicher

Abbildung 1: Trace-Speicheranordnungen

zu einem Trace-Strom maßgeblich von der Analyseaufgabe selbst bestimmt werden, hängt die Größe des Trace-Zwischenspeichers vor allem vom zu erwartenden Trace-Datenaufkommen sowie der Geschwindigkeit der Trace-Ausgabeschnittstelle ab. Da die Realisierung leistungsfähiger Ausgabeschnittstellen oft problematisch ist (hohe Pin-Anzahl bei paralleler und High-Speed Transmitter bei serieller Realisierung), stellt die Chip- bzw. Probe-basierte Zwischenspeicherung eine interessante Alternative dar, um Trace-Daten über schmalbandige Schnittstellen auszugeben. Mögliche Systemanordnungen sind in Abbildung 1 dargestellt. Das größte Trace-Datenaufkommen ist genau dann gegeben, wenn ein vollständiger Trace generiert wird. Dieser umfasst sowohl Befehl- und Load-/Store-Trace als auch zusätzliche Zeit- und Statusinformationen. Die in diesem Beitrag durchgeführten Analysen orientieren sich an einem fast vollständigen Trace-Datenstrom ohne Zeitinformationen, da diese für die Rekonstruktion des Programm- und Datenflusses nicht erforderlich sind. Die zusätzlich erforderliche Trace-Speichergröße kann als gering eingeschätzt werden, da Zeitstempelinformationen aufgrund ihrer geringen Änderung von Befehl zu Befehl relativ gut komprimierbar sind. Obwohl in diesem Beitrag nicht weiter betrachtet, spielen Zeitstempelinformationen für die Echtzeitanalyse eine entscheidende Rolle.
Die Auslegung des Trace-Speichers hängt - sowohl On-Chip als auch Off-Chip - hauptsächlich vom Anwendungsprofil, also dem vorliegenden Befehls- und Datenfluss ab. Da die Größe insbesondere des On-Chip Trace-Speichers die Chip-Kosten direkt beeinflusst, ist die Wahl einer dem Anwendungsprofil entsprechenden Trace-Speichergröße aus ökonomischer Sicht notwendig. Dies setzt eine Analyse des Anwendungsprofils relevanter Embedded-Applikationen bzgl. Rekonstruierbarkeit des abgearbeiteten Befehls- und Datenflusses voraus, die Gegenstand dieses Beitrages ist. Als repräsentative Applikationslast werden zu diesem Zweck Teile des EEMBC Industrial/Automotive Benchmarks [2] hinsichtlich ihres Trace-Speicherbedarf untersucht.

2 Trace-Datenerfassung

Da der Trace-Datenstrom bei hohen Prozessortaktfrequenzen ein großes Datenvolumen annehmen kann, ist es notwendig, diesen zu komprimieren. Entsprechende Techniken lassen sich sowohl für Befehls- als auch für Load-/Store-Trace anwenden. Bei der Generierung von Befehlstraces lassen sich deren spezifische Eigenschaften nutzen, die sich in folgende Kategorien untergliedern lassen [7]:

Verzweigung mit konstanter Zieladresse (direkt): Das Auftreten ist deterministisch. Da die Zieladresse bekannt ist, muss sie nicht extra übertragen werden. Im Quelltext treten diese Verzweigungen durch Call- und If-Anweisungen auf.

Bedingte Verzweigung mit konstanter Adresse (direkt) : Die Zieladresse ist konstant und muss nicht übertragen werden. Einzig das Erfüllen der Bedingung und ob der Sprung überhaupt durchgeführt wird, muss aufgezeichnet werden. Im Quelltext treten solche Verzweigungen durch Loop- und If-Anweisungen auf.

direkte Sprünge	2 Byte
indirekte Sprünge	3 Byte
Nicht ausgeführte Instr.	3 Byte
Load/Store	7 Byte

Tabelle 1: Aufzeichnungslänge einzelner Trace-Ereignisse

Verzweigung zu einer zur Laufzeit bestimmten Zieladresse (indirekt): Da nicht statisch ermittelt werden kann, wohin der Sprung führt, muss die Zieladresse aufgezeichnet werden. Im Quelltext treten solche Verzweigungen durch Return und Switch-Anweisungen auf.

Bedingte Verzweigung zu einer zur Laufzeit bestimmten Zeiladresse (indirekt): Es muss aufgezeichnet werden, ob und wohin der Sprung im jeweiligen Fall erfolgte. Im Quelltext treten solche Verzweigungen durch Return- und Switch-Anweisungen auf.

Zusätzlich zu diesen Prinzipien können Adressinformation durch Ausnutzung des Lokalitätspinzips in komprimierter Form aufgezeichnet werden. Die örtliche Lokalität ist hoch, da Sprünge meist kurz und direkt sind. So bietet sich zum Beispiel statt der vollständigen Adresse die Übertragung der Differenz [5] oder der niederwertigen Bits [1] an. Neben dem Befehlstrace können auch Load-/Store-Aufzeichnungen von der Lokalität von Datenzugriffen profitieren, wobei nur deren Adressinformationen komprimierbar sind. Eine Kompression der gelesenen oder geschriebenen Speicherinhalte ist nicht möglich. Zugriffe auf Prozessorregister sind aufgrund ihres hohen Datenvolumens nicht zur sinnvollen Trace-Ausgabe geeignet. Sie lassen sich jedoch mit den in Abschnitt 3 beschriebenen Techniken mittels iterativer Befehlssatzsimulation rekonstruieren. Tabelle 1 zeigt die gewählte Kodierung für die Aufzeichnung der Trace-Ereignisse. Da einige Prozessorarchitekturen (z.B. ARM) jeden Befehl bedingt ausführen können, müssen nicht ausgeführte Befehle separat im Trace-Strom kodiert werden. Die angenommene Kodierung der Trace-Ereignisse orientiert sich an der Kodierung in realen Trace-Einheiten [1] [5] [3], sie ist jedoch zu keiner praktischen Realisierung identisch. Letzendlich wurden die in diesem Beitrag verwendeten Trace-Ereignisse durch den Simulator Jahris [4] für ein ARMv4 Prozessormodell generiert, da dieses Modell kurzfristig verfügbar war. Weitere Experimente mit anderen Prozessormodellen - insbesondere PowerPC - sind geplant.

3 Trace-Rekonstruktion

Die aufgrund der begrenzten Kapazität der Trace-Hardware im Bezug auf die Menge der Prozessorzustandsinformationen unvollständige Erfassung erfordert die Anwendung von Verfahren, die die fehlenden Trace-Daten ergänzen und komplettieren. Da wie bereits im Abschnitt 2 erläutert nur Load-/Store- sowie Branch-Befehle aufgezeichnet werden, müssen insbesondere Registerinhalte nachträglich ermittelt werden. Zur Rekonstruktion von Registerinhalten wird zunächst der ausgeführte Programmfluss anhand des Branch-Befehlstraces nachvollzogen, auf dessen Basis daraufhin eine komplette Datenflussanalyse über alle Registertransfers erfolgen kann. Den wichtigsten Rekonstruktionsschritt stellt die darauffolgende iterative Befehlssatzsimulation dar, wobei zunächst alle Befehle mit durch den Load-/Store-Trace erfassten Quelloperanden ausgeführt werden können. Im weiteren Simulationsverlauf steigt dabei die Anzahl der bekannten Registerinhalte, bedingt durch die Tatsache, dass diese wiederum als Quelloperand für weitere Operationen fungieren. Die Simulation ist beendet, wenn keine neuen Registerbelegungen mehr berechnet werden können. Einen Sonderfall stellt dabei die inverse Befehlssimulation dar, bei der einige ausgewählte, einfache, umkehrbare arithmetisch/logische Operationen (Addition, Subtraktion, Multiplikation, XOR) invers ausgeführt werden, um von Zielregisterbelegungen auf die Inhalte von Quellregistern schließen zu können. Für detailierte Informationen zur Realisierung der hier angewandten Methoden zur Rekonstruktion von Registerinhalten sei auf [6] verwiesen. Im Rekonstruktionsverlauf werden dabei sowohl kurzlebige Registerinhalte für Zwischenergebnisse arithmetischer Berechnungen als auch langlebige Registerinhalte wie Adresszeiger (Stack-Pointer), Schleifenzähler und lokale Variablen ermittelt. Mit zunehmender Länge des Registerbelegungsintervalls ist auch mit einer Erhöhung des Trace-Speicherbedarfs zu rechnen, da bei

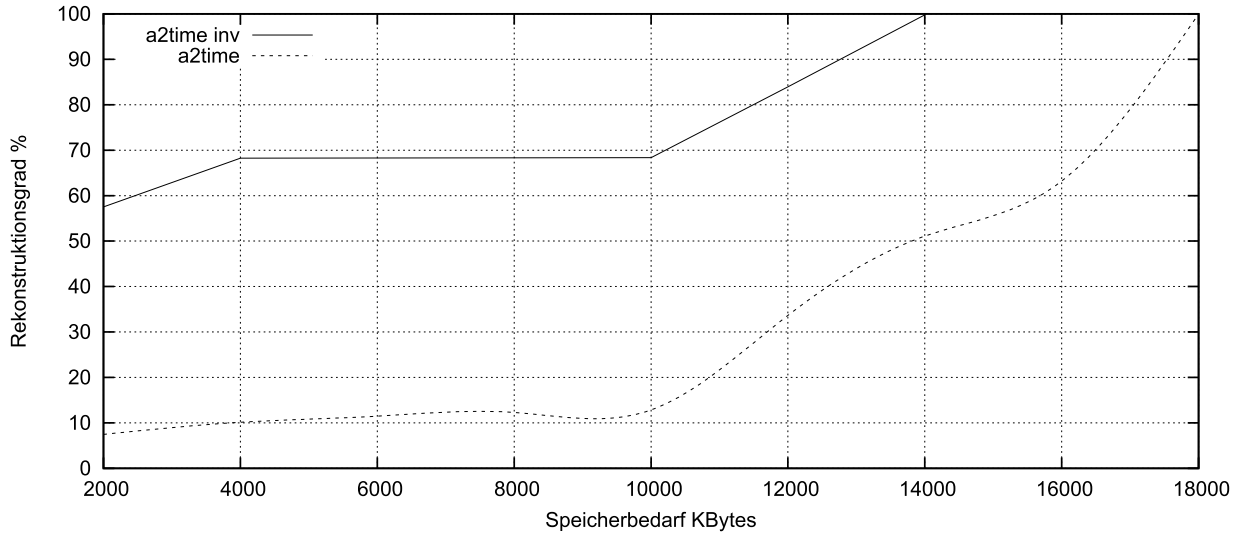


Abbildung 2: Trace-Speicherbedarf `a2time` Benchmark

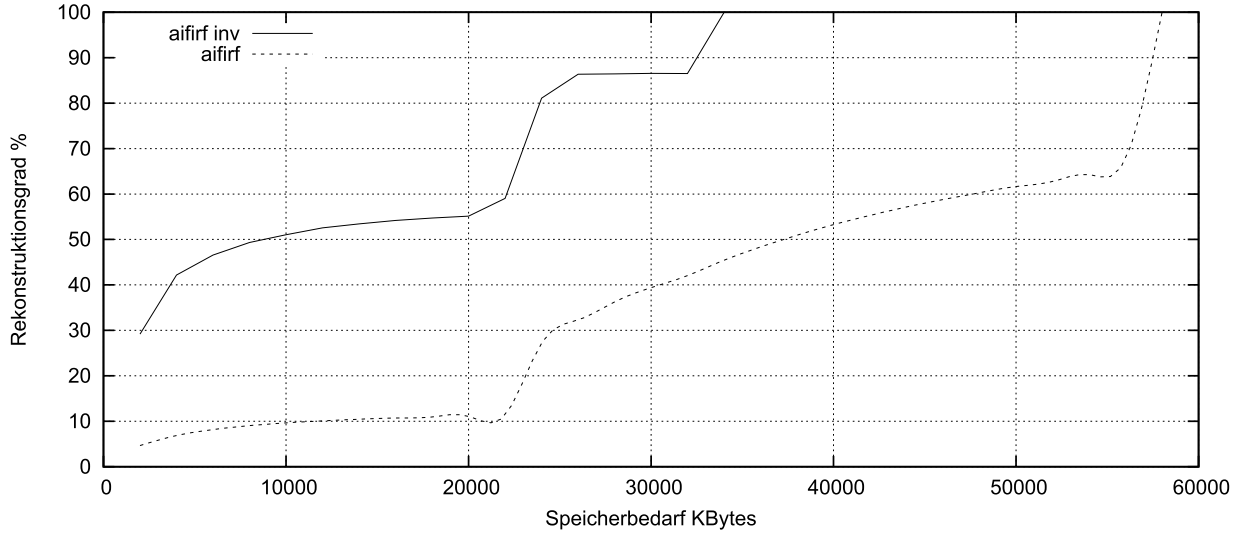


Abbildung 3: Trace-Speicherbedarf `aifir` Benchmark

unzureichender Speichergröße ein Trace-Ereignis zum Laden eines langlebigen Registers außerhalb des aufgezeichneten Intervalls liegen kann.

4 Experimentelle Ermittlung der Trace-Speichergröße

Zur Ermittlung der optimalen Trace-Speichergröße wurden vier Benchmarks aus der EEMBC Industrial/Automotive Benchmarksuite analysiert. Um den Speicherbedarf zur vollständigen Rekonstruktion aller - also insbesondere auch langlebiger - Registerinhalte unter Worst-Case-Bedingungen zu ermitteln, wird angenommen, dass die Trigger-initiierte Trace-Aufzeichnungen zu jedem beliebigen Zeitpunkt der Programmausführung gestartet werden kann. Die Aufzeichnung ist beendet, wenn der Trace-Speicher keine Trace-ereignisse mehr aufnehmen kann. In Schritten von 2KByte wird dabei die Trace-Speichergröße so lange erhöht, bis ein Rekonstruktionsgrad von 100%, bezogen auf alle Registerinhalte zu allen Trace-abgedeckten Ausführungszeitpunkten, erreicht ist.

Die Ergebnisse des Registerrekonstruktionsgrades in Abhängigkeit von der Trace-Speichergröße mit und ohne inverse Befehlssatzsimulation sind in den Abbildungen 2 bis 5 dargestellt. Die erforderliche Trace-Speichergröße fällt erwartungsgemäß unterschiedlich aus. Während die Benchmarks `a2time` (Trigonometrische Berechnungen) `aifirf` (Digitale Filter) und `tblook` (Table-Lookup) vor allem durch kurze Registerintervalle mit einem hohen Load-/Store-Anteil gekennzeichnet sind, werden Berechnungen beim

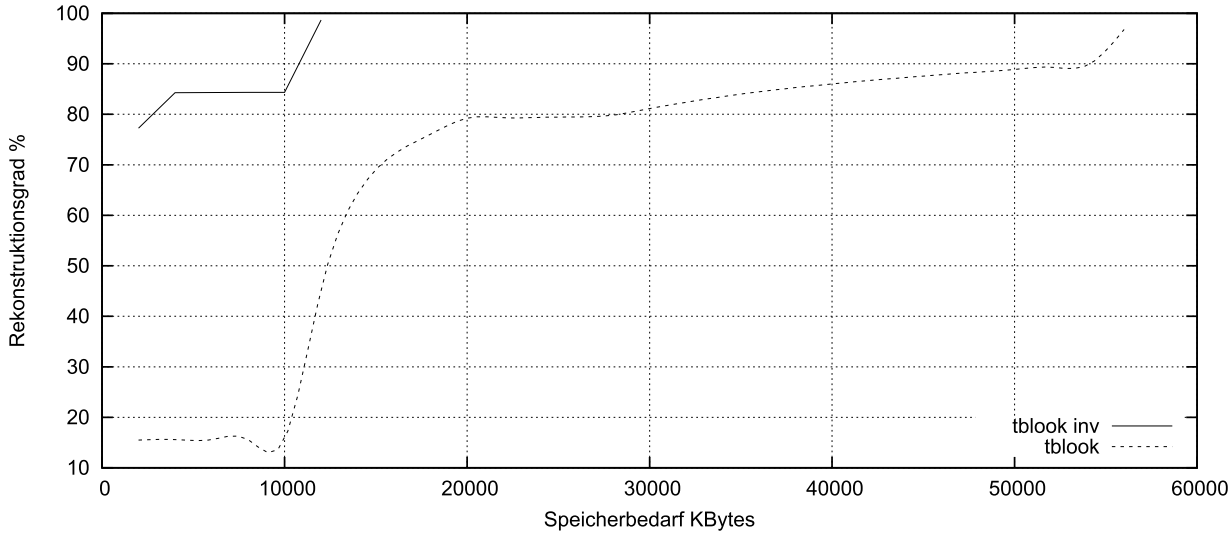


Abbildung 4: Trace-Speicherbedarf `tblook` Benchmark

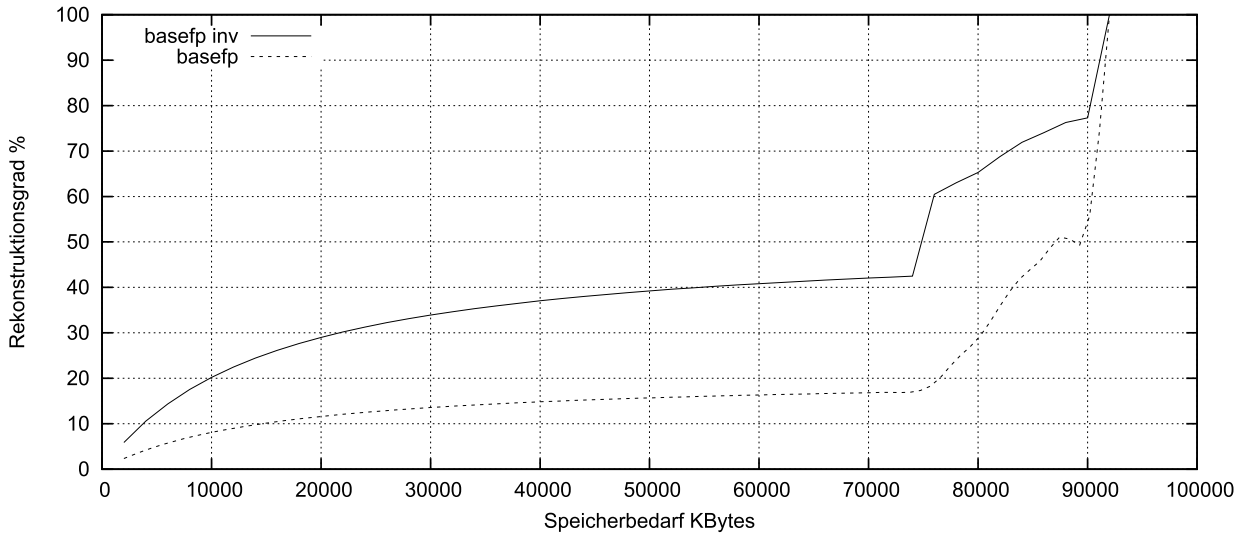


Abbildung 5: Trace-Speicherbedarf `basefp` Benchmark

`basefp` (Floating-Point Emulation) hauptsächlich in Registern ausgeführt (Worst-Case). In letzerem Fall muss das Laden der Quelldaten vollständig durch den Load-/Store-Trace abgedeckt werden. Das resultierende Trace-Profil wird ansonsten weitgehend durch die ausgeführten Branch-Befehle bestimmt. Aufgrund des sehr geringen Load-/Store-Tracevolumens ist der bei ausgeglichenen Traceprofilen beobachtbare deutliche Gewinn durch inverse Befehlssatzsimulation in diesem Fall geringer.

5 Zusammenfassung

Die applikationsspezifische Analsyse des Trace-Speicherbedarfs kann als ein geeignetes Verfahren angesehen werden, um eine sinnvolle On-Chip oder Off-Chip Trace-Speichergröße für eine konkrete Hardwarerealisierung zu ermitteln. Es wurde weiterhin gezeigt, dass die erforderliche Größe des Trace-Speichers durch begrenzte Anwendung einer inversen Befehlssatzsimulation reduziert werden kann.
Obwohl die herangezogenen Rekonstruktionsergebnisse ausschlißlich für eine ARMv4 Architektur ermittelt wurden, so sind diese aufgrund der Ähnlichkeit der Architeturmodelle und damit der Registerbelegung auch eingeschränkt repräsentativ für weitere eingebettete RISC Prozessoren mit Load-/Store-Architektur. Es ist geplant, dies anhand einer PowerPC Architektur unter Nutzung von realen NEXUS-basierten Trace-Daten [3] nachzuweisen.

Literatur

[1] ARM Ltd. *CoreSight on-chip trace and debug*, 2007.

[2] Embedded Microprocessor Benchmark Consortium. EEMBC Industrial/Automotive Benchmark. Technical teport, `http://www.eembc.org`, 2007.

[3] IEEE-Industry Standards and Technology Organization (IEEE-ISTO). *The Nexus 5001 Forum - Standard for a Global Embedded Processor - Debug Interfacen. 2.0*, 2003.

[4] Marco Kaufmann, Thomas B. Preußer, Rainer G. Spallek. Ein retargierbarer, hochperformanter ISA-Simulator in Java. In *Dresdner Arbeitstagung Schaltungs- und Systementwurf (DASS2010)*, 2010.

[5] MIPS Technologies Inc. *MIPS PDtrace Specification. Rev 6.11*, 2008.

[6] Steffen Köhler, Andreas Gajda, Rainer G. Spallek. Fehleranalyse mittels Trace-basierter Rekonstruktion von Prozessorzuständen. In *Dresdner Arbeitstagung Schaltungs- und Systementwurf*. Fraunhofer IIS, 2010.

[7] X. HU, S. CHEN. Applications of on-chip trace on debugging embedded processor. In *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*. IEEE Computer Society, 2007.

Entwurf eines FPGA-basierten heterogenen rekonfigurierbaren eingebetteten Systems

Daniel Kriesten, Volker Pankalla, Ulrich Heinkel
Technische Universität Chemnitz
Professur Schaltkreis- und Systementwurf
09107 Chemnitz
<krid, pav, heinkel>@hrz.tu-chemnitz.de

Zusammenfassung

Im Rahmen dieses Beitrages stellen wir die Fortführungen unserer Arbeiten auf dem Gebiet der Run-time Reconfiguration vor. Für Untersuchungen an realen, eingebetteten heterogenen rekonfigurierbaren Systemen wurde ein System-on-Module um eine rekonfigurierbare Einheit erweitert. Im Beitrag wird die gewählte Systemarchitektur mit möglichen Alternativen verglichen. Außerdem beschreiben wir die Umsetzung von dynamischer und dynamisch partieller Rekonfiguration, sowie die dafür genutzten Tools und Arbeitsschritte. Da Linux als Betriebssystem für das eingebettete System gewählt wurde, wird auch die Integration des FPGA in den Betriebssystemkontext betrachtet.

1. Einleitung

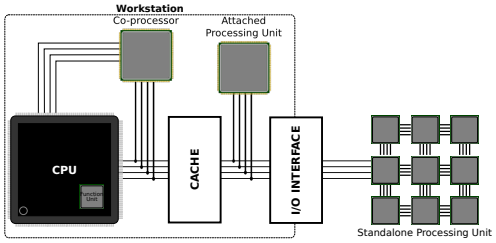
Galten *Field Programmable Gate Arrays (FPGAs)* noch vor wenigen Jahren als stromhungrige Abwärmeproduzenten, erfüllen die heutigen Produkte auch die Anforderungen des embedded Marktes nach geringer Energieaufnahme und Verlustleistung. Doch im Bereich eingebetteter Systeme, mit ihren konträren Anforderungen an Rechenleistung, Energieverbrauch und Verlustleistung, stellen sich völlig neue Herausforderungen. Eines der Kernthemen ist Steigerung der Effizienz durch die Veränderung der Konfiguration der verwendeten reprogrammierbaren Hardwareeinheit zur Laufzeit, im Folgenden mit dem englischen Begriff *Run-time Reconfiguration (RTR)* bezeichnet. Trotz der Vielfalt an rekonfigurierbaren Schaltkreisen ist auch bei eingebetteten Systemen häufig die Kombination von Prozessor und FPGA anzutreffen. Diese Klasse von Systemen wird oft als *heterogene Hardware-Systeme* oder *heterogene rekonfigurierbare Systeme (HRS)* bezeichnet [Koc02].
Das Konzept der HRS erfährt gerade einen neuen Boom. Auf der Embedded World, die im März 2011 in Nürnberg stattfand, stelle Marktführer Xilinx seine neue Produktreihe Zynq™ vor. Bei dieser Kombination aus FPGA und Leistungsfähigem Universalprozessor steht der Prozessor im Vordergrund. Xilinx bezeichnet diese Konzept als *processor-centric design* ([DeH10]). Mit der Stellarton-Serie führten Intel® und Altera Ende 2010 eine Kombination aus Leistungsfähigem x86-Prozessor und FPGA, vereint auf einem Die-Carrier, in den Markt ein. Durch die Nutzung von eingebetteten, rekonfigurierbaren Blöcken, wie sie beispielsweise in der MORPHEUS-Plattform [VRH09] des gleichnamigen EU-Projekts zum Einsatz kommen, oder der *stacked silicon interconnect technology*, zur Kombination mehrerer Blöcke auf einem Die, stehen weitere Architekturen vor dem Durchbruch.
Im Rahmen dieses Beitrages stellen wir die Fortführungen unserer Arbeiten [SKH09, KHS10] auf dem Gebiet der RTR vor. In Abschnitt 2 stellen wir verschiedene Hardwarearchitekturen für RTR sowie die von uns gewählte Architektur vor. Abschnitt 3 betrachtet verschiedene Ansätze für geeignete Software und Abschnitt 4 fasst die Ergebnisse der Arbeiten zusammen.

2. Hardwaresystem

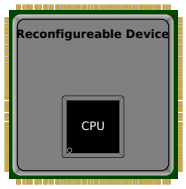
Ziel des Hardwareentwurfs war die Erweiterung eines vorhandenen *eingebettetes System (ES)* zum RTR, so dass die Untersuchung und Umsetzung von Strategien zur dynamischen und zur dynamisch partiellen Rekonfigration möglich sind. Daher werden in diesem Abschnitt die Systemlevel-Architekturen für das Design von HRS sowie die zur Konfiguration vorhandenen Schnittstellen vorgestellt und diskutiert. Abschließend wird das für die vorliegende Arbeit verwendete System erläutert.

2.1. System-Level Architektur

Allgemein besteht ein heterogenes rekonfigurierbares System aus einem oder mehreren Prozessoren, einer oder mehreren rekonfigurierbaren Einheiten sowie allen weiteren benötigten Komponenten. Oftmals werden die Systeme nach dem Grad der Kopplung zwischen rekonfigurierbarer Einheit und Prozessor unterschieden ([CH02, TCW+05]).



(a) Vier System-Level Architekturen von [CH02]



(b) Fünfte Architektur nach [TCW+05]

Abbildung 1: Fünf System-Level Architekturen

Abbildung 1 stellt fünf Architekturen dar, die jeweils einen anderen Grad der Kopplung aufweisen. Eine direkt in die CPU integrierte rekonfigurierbare Einheit oder aber ein in ein rekonfigurierbares Array eingebetteter Prozessor (als Soft-Core oder Hardmakro, siehe Abbildung 1b) stellen die engste Form der Kopplung dar. Während Co-Prozessoren und Multicore-ähnliche Systeme weniger fest gekoppelt sind, stellen über externe I/O-Schnittstellen angebundene rekonfigurierbare Arrays die loseste Form der Kopplung dar. Generell gilt, dass bei Systemen mit engerer Kopplung die rekonfigurierbare Einheit kleiner ist. Allerdings ermöglicht die engere Anbindung höhere Geschwindigkeiten bei der Kommunikation und der Rekonfiguration. Daher erfüllen lose gekoppelte Systeme meist eigenständige Aufgaben, die ein geringeres Maß an Kommunikation erfordern. Derzeit werden vor allem *Static Random Access Memory (SRAM)* basierte FPGA als rekonfigurierbare Einheiten in HRS eingesetzt.

2.2. Konfigurationsstrategien

SRAM basierte FPGA verlieren beim Trennen der Stromversorgung ihre Konfiguration. Daher müssen diese Schaltkreise mindestens einmal zur Einschaltzeit konfiguriert werden. Dabei gibt es zwei grundlegende Konzepte in HRS. Wenn die Konfiguration des FPGAs zur Laufzeit des Systems nicht geändert werden soll oder kann, wird die *statische Konfiguration* verwendet. Dabei ist der notwendige Bitstrom meist in einem *Programmable Read-Only Memory (PROM)* hinterlegt. Das zweite Konzept ist die Run-time Reconfiguration, also die Veränderung der Konfiguration des FPGAs zur Laufzeit des Systems. Alle aktuellen FPGAs wie auch die meisten älteren Serien unterstützen die dynamische Rekonfiguration, bei der das FPGA angehalten und mit einem anderen

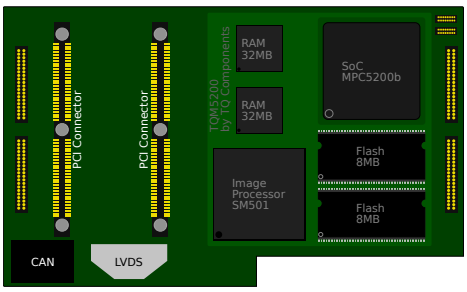
Bitstrom neu konfiguriert wird. Dieser Vorgang kann, wie in [KHS10] gezeigt, schnell mehrere 10 Sekunde benötigen. Um diesem Problem zu begegnen, unterstützen einige FPGAs dynamisch partielle Rekonfiguration. Dabei werden nur Teile des FPGAs neu konfiguriert, während das restliche System weiter arbeitet. Nachteile dieser Technologie sind die für die Konfiguration zusätzlich benötigte Hardware, der veränderte Toolflow und der höhere Preis der FPGAs.

2.3. Schnittstellen für die Konfiguration

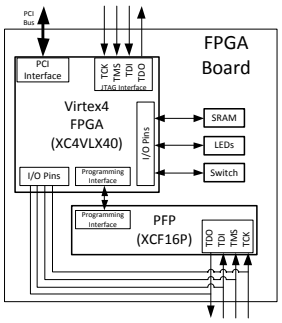
Die Konzepte und Schnittstellen für die Rekonfiguration von FPGAs der verschiedenen Hersteller sind einander ähnlich. Sowohl die FPGAs als auch die PROMs bieten eine *Joint Test Action Group (JTAG)*-konforme Schnittstelle zur Programmierung. Allerdings ist die Geschwindigkeit dieser Schnittstelle begrenzt, so dass die FPGAs zusätzlich serielle und parallele Schnittstellen bieten, die von den zugehörigen, herstellerspezifischen PROMs bedient werden können. Neuere Serien bieten oft weitere Schnittstellen wie *Serial Peripheral Interface (SPI)* zur Anbindung externer Flash-Speicher.

2.4. Basissystem und FPGA-Board

Zwei der in Abschnitt 2.1 vorgestellten Architekturen finden sich häufig in derzeit verfügbaren HRS. Dies ist zum einen die loseste Kopplung über einen externen Bus und zum anderen die engste Kopplung mit einem eingebetteten Prozessor.
In Abbildung 2a ist das zu erweiternde Basissystem dargestellt. Der Prozessor, ein Motorola MPC5200b *System-on-Chip (SoC)*, eine einfache *Graphics Processing Unit (GPU)* sowie einige Peripherieschaltkreise sind auf einem *System-on-Module (SoM)* untergebracht. Dieses SoM ist mit einem Basisboard verbunden, welches unter anderem die Schnittstellen des Systems nach außen sowie die Spannungsversorgung beherbergt.



(a) Architektur des Eingebetteten Systems



(b) Architektur des FPGA-Boards

Abbildung 2: Basissystem und FPGA-Board

Durch seine Modularität bietet dieses System die Möglichkeit, das verwendete SoM durch ein alternatives zu ersetzen. Würde man hier einen geeignetes FPGA einsetzen, wäre die Realisierung eines RTR mit eingebettetem Prozessorkern vorstellbar. Aus Kosten- und Effizienzgründen wurde jedoch eine andere Option gewählt. Über den auf dem Basisboard vorhandenen *Peripheral Component Interconnect (PCI)*-Konnektor wird das System um ein FPGA-Modul erweitert. Für das Modul wurde ein Xilinx Virtex4 FPGA gewählt. Dieses FPGA ist ausreichend dimensioniert, um eine Vielzahl von Experimenten zu ermöglichen. Auf dem FPGA-Board, welches in Abbildung 2b dargestellt ist befinden sich zusätzlich ein Xilinx „*Platform Flash PROM (PFP)*“ XCF16P zur Aufnahme eines Bitstroms, ein Cypress CY7C1062AV33 512Kx32 SRAM, vier LEDs und ein DIP-Schalter und je ein JTAG-Interface zum Virtex4 FPGA und zum PFP.

Um das System so kompakt wie möglich zu halten, wurde auf zusätzliche Bauteile verzichtet und das FPGA direkt mit dem PCI-Bus verbunden. Einige *Input und Output (I/O)*-Pins des FPGAs sind mit dem JTAG-Interface des PFPs verbunden, um so die dynamische Rekonfiguration zu realisieren. Die extern erreichbaren JTAG-Interfaces zum FPGA und zum PFP dienen lediglich Debugging-Zwecken.

3. Software für heterogene rekonfigurierbare Systeme

Das in Abschnitt 2.4 beschriebene System erlaubt sowohl die Umsetzung von dynamischer als auch dynamisch partieller Rekonfiguration. Beide Ansätze sowie die verwendeten Tools werden im folgenden näher betrachtet.

3.1. Betriebssystem und Firmware

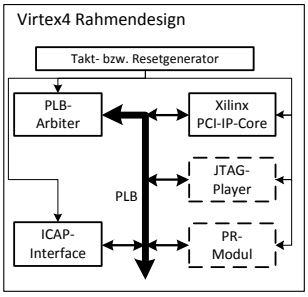
Derzeit erscheint das Linux-*Betriebssystem (BS)* die geeignete Wahl für eingebettete Systeme mit hinreichenden Ressourcen. Sowohl der BS-Kern als auch die meisten Anwendungsprogramme sind kostenfrei und im Quellcode erhältlich, was Erweiterungen auch auf Betriebssystemebene ermöglicht. In vielen Fällen wird zusätzlich ein Bootloader verwendet, der das System initialisiert und anschließend den Linux-Kernel startet. Oftmals kommt zu diesem Zweck die Open-Source Software *Das U-Boot* ([Zun10]) zum Einsatz.
Bei HRS muss die Möglichkeit bestehen, aus dem Betriebssystem oder der Firmware heraus das rekonfigurierbare Array zu programmieren. Untersuchungen hierzu wurden unter anderem in [KHS10] und [SHV+09] dargestellt. Für den Aufbau der Software, die den Rekonfigurationsvorgang steuert, gibt es verschiedene Implementierungsvarianten. Zur Designzeit muss eine Partitionierung der Software erfolgen, wobei Teile der Software im *Kernel-Space* und Teile im *User-Space* implementiert werden können. Wie beispielsweise in [KHS10] dargestellt, hat diese Entscheidung entscheidenden Einfluss unter anderem auf Performance und Wartbarkeit.

3.2. Toolflow für dynamische und dynamisch partielle Rekonfiguration

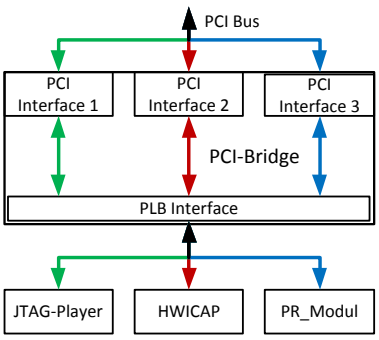
Der von uns verwendete Design- und Toolflow orientiert sich weitgehend am Standard Xilinx Designflow wie er in [Xil08] beschrieben ist, und umfasst dabei mindestens die Beschreibung und Synthese des Designs, die Implementierung des Designs und die Verifikation.
Bei der Umsetzung des Desigflow kommen die Tools aus der *Xilinx Integrated Software Environment (ISE) Design Suite 12 (ISE12)* zum Einsatz. Dabei wird das Design zunächst mit dem *Embedded Development Kit (EDK)* erstellt und synthetisiert. Die Implementierung für die dynamische Rekonfiguration erfolgt ebenfalls mit dem EDK, während für die dynamisch partielle Rekonfiguration zusätzlich das Tool *PlanAhead* genutzt wird.
Neben allgemeinen Optimierungen wurde der komplette Designflow für dynamisch partielle Rekonfiguration in der ISE12 neu gestaltet und entscheidend verbessert. So ist nun zum Beispiel die Übergabe des Designs vom EDK zu *PlanAhead* möglich, ohne dass die Projektdateien manuell bearbeitet werden müssen. Auch das Verwalten der Signalgrenzen zwischen dem dynamischen und dem statischen Teil des Designs geschieht nun automatisch und erfordert nicht länger das manuelle Hinzufügen so genannter Busmakros.

3.3. Umsetzung zweier Rekonfigurationsstrategien

Zur Umsetzung der beiden nachfolgend beschriebenen Rekonfigurationsstrategien wurde zunächst das in Abbildung 3a dargestellte Rahmendesign erstellt. Der *Xilinx PCI-IP-Core* wurde, wie in Abbildung 3b gezeigt, mit drei getrennten Adressräumen für den *JTAG-Player*, das *Internal Configuration Access Port (ICAP)-Interface* und das *PR-Modul* konfiguriert.



(a) Rahmendesign



(b) PCI Busanbindung

3.3.1. Dynamische Rekonfiguration des FPGAs

Zuerst wurde, basierend auf der *Xilinx Application Note (XAPP)* 975 [HP08], die *dynamische Rekonfiguration (dR)* umgesetzt. Dabei wird der neue Bitstrom für das FPGA vom SoM über den PCI-Bus durch das FPGA in den PFP übertragen. Ein als *JTAG-Player* bezeichnets Logik-Modul übernimmt die Ansteuerung der Programmierschnittstelle des PFP. Um sicherzustellen, dass bei der Übertragung des Bitstroms keine Fehler auftraten, wird der Bitstrom zusätzlich mit einer Prüfsumme versehen. Damit ist ein stabiler Systemzustand nach der Rekonfiguration des FPGAs grundlegend gewährleistet. Kommt es bei der Rekonfiguration zu einem Fehler, kann das System nur durch einen manuellen Eingriff wieder in Betrieb genommen werden.

3.3.2. Dynamisch partielle Rekonfiguration des FPGAs

Im Gegensatz zur Lösung aus Abschnitt 3.3.1 muss bei der *dynamisch partielle Rekonfiguration (dpR)* lediglich der partielle Bitstrom für das *PR-Modul* via PCI an den ICAP-Controller des FPGAs übertragen werden. Dieser übernimmt anschließend die partielle Rekonfiguration. Der statische Teil des Designs wird schon zur Produktionszeit des Systems im PFP gespeichert. Somit ist das System immer in einem funktionsfähigen Zustand. Lediglich die Integrität des *PR-Modul* muss nach der Rekonfiguration sicher gestellt werden, beispielsweise durch einen *Build-in Self Test (BIST)*.

3.4. Konfigurationssoftware

Der Rekonfigurationsvorgang wird in beiden Beispielen durch das auf dem SoM verwendete Linux-BS gesteuert. Die Implementierung erfolgte komplett im *User-Space*, was die Performance negativ beeinflussen könnte, aber den Implementierungs- und Wartungsaufwand verringert. Das entwickelte C-Programm hat über den PCI-Bus Zugriff auf den *JTAG-Player* und den ICAP-Controller im FPGA. Das Programm Übernimmt die Übertragung der Bitströme an das FPGA und gibt den Nutzer Feedback über den Verlauf der Rekonfiguration.

4. Zusammenfassung und Ausblick

In diesem Beitrag stellen wir die Erweiterung eines ESs um eine rekonfigurierbare Einheit vor, wobei wir uns bei der Umsetzung am Stand der Wissenschaft orientiert haben. Neben dem Hardwaredesign fanden auch aktuelle Erkenntnisse der Softwarearchitektur Beachtung, wobei hier noch weiterer Forschungsbedarf besteht.

Danksagung

Die hier vorgestellten Arbeiten entstanden im Rahmen des Projekts *Generalisierte Plattform zur Sensordatenverarbeitung* (Fkz.: 03IP505) und wurden durch das Bundesministerium für Bildung und Forschung (BMBF) innerhalb der Initiative „InnoProfile - Unternehmen Region“ gefördert . Weitere Informationen finden sich unter [GPS06].

Literatur

[CH02] Compton, Katherine und Scott Hauck: *Reconfigurable computing: a survey of systems and software*. ACM Computing Surveys (CSUR), 34(2):171, 2002, ISSN 0360-0300.

[DeH10] DeHaven, Keith: *Extensible Processing Platform - Ideal Solution for a Wide Range of Embedded Systems*, 2010. `http://www.xilinx.com/support/documentation/white_papers/wp369_Extensible_Processing_Platform_Overview.pdf`.

[GPS06] GPSV, InnoProfile Initiative: *Generalisierte Plattform zur Sensordatenverarbeitung*, März 2006. `http://www.gpsv.tu-chemnitz.de`.

[HP08] Hussein, Jameel und Rish Patel: *Xilinx XAPP975: Low-Profile In-System Programming Using XCF32P Platform Flash PROMs*. Xilinx Inc., v1.0.3 Auflage, May 2008. `http://www.xilinx.com/support/documentation/application_notes/xapp975.pdf`.

[KHS10] Kriesten, Daniel, Ulrich Heinkel und Jörg Schneider: *Integration von Konfigurationsmechanismen für Xilinx-FPGA in das Linux-OS*, 2010.

[Koc02] Koch, Andreas: *Architectures and Tools for Heterogeneous Reconfigurable Systems*. Workshop on Heteregeneous Reconfigurable Systems, Seiten 1–2, 2002.

[SHV+09] Strunk, Jochen, Andreas Heinig, Toni Volkmer, Wolfgang Rehm und Heiko Schikc: *Run-Time Reconfiguration for HyperTransport coupled FPGAs using ACCFS*, Seiten 54–63. WHTRA, 2009, ISBN 978-3-00-027249-3.

[SKH09] Schneider, Jörg, Daniel Kriesten und Ulrich Heinkel: *Fehlertolerante Firmware-Updates FPGA-basierter Funkprototypensysteme mit Linux-OS*. In: *9. Chemnitzer Fachtagung Mikrosystemtechnik*. Technische Universität Chemnitz, 2009, ISBN 978-3-00-029135-7.

[TCW+05] Todman, T J, G. A. Constantinides, S J E Wilton, O Mencer, W Luk und P Y K Cheung: *Reconfigurable computing: architectures and design methods*. IEE Proceedings - Computers and Digital Techniques, 152(2):193, 2005, ISSN 13502387.

[VRH09] Voros, Nikolaos S., Alberto Rosti und Michael Hübner: *Dynamic System Reconfiguration in Heterogeneous Platforms*, Band 40 der Reihe *Lecture Notes in Electrical Engineering*. Springer Netherlands, Dordrecht, 2009, ISBN 978-90-481-2426-8.

[Xil08] Xilinx Inc.: *Xilinx Development System Reference Guide*, Kapitel 2, Seiten 29–52. Xilinx Inc., 2008. `http://www.xilinx.com/itp/xilinx10/books/docs/dev/dev.pdf`.

[Zun10] Zundel, Detlev: *The DENX U-Boot and Linux Guide (DULG)*, 2010. `http://www.denx.de/wiki/DULG/WebHome`, besucht: 2010-11-03.

Automatische und teilautomatische Generierung anwendungsspezifischer Beschleunigungshardware aus der Softwarebeschreibung

Timm Bostelmann
FH Wedel
Feldstraße 143
22880 Wedel
bos@fh-wedel.de

Sergei Sawitzki
FH Wedel
Feldstraße 143
22880 Wedel
saw@fh-wedel.de

Kurzfassung

Durch die Erweiterung eines Universalprozessors mit anwendungsspezifischer Hardware lässt sich die Berechnung vieler Algorithmen erheblich beschleunigen. Dabei erfordert die Beschreibung der Hardware mit einer klassischen Hardwarebeschreibungssprache wie Verilog HDL oder VHDL ein hohes Maß an Erfahrung mit dem Entwurf digitaler Systeme. Der vorliegende Beitrag beschreibt die Möglichkeiten zur automatischen und teilautomatischen Generierung anwendungsspezifischer Beschleunigungshardware aus der Softwarebeschreibung und vergleicht diese bezüglich des Aufwands, der Geschwindigkeit und des Ressourcenbedarfs mit einer reinen Softwareimplementierung und dem manuellen Entwurf in VHDL.

1 Motivation

Die Verfügbarkeit leistungsfähiger und günstiger programmierbarer Logik ermöglicht seit einiger Zeit deren Einsatz an Stelle klassischer digitaler Signalprozessoren. Dabei ermöglicht die Anpassung der Hardware an das Problem eine deutliche Leistungssteigerung. Darüber hinaus kann bei hohen Stückzahlen die Umsetzung in eine anwendungsspezifische integrierte Schaltung und damit eine weitere Optimierung der Leistung, Kosten und des Energiebedarfs erfolgen. Die Implementierung des vollen Funktionsumfangs in anwendungsspezifischer Hardware kann jedoch sehr aufwändig sein. Häufig ist nur ein Bruchteil der Gesamtfunktion tatsächlich zeitkritisch. Entsprechend ist es in vielen Fällen sinnvoll ausschließlich die zeitkritischen Teile in Spezialhardware abzubilden und diese an einen Universalprozessor anzubinden. Die weniger kritischen oder schlecht parallelisierbaren Teile können in diesem Fall weiterhin als Software codiert und auf dem Prozessor ausgeführt werden. Dies betrifft in der Regel die Benutzerinteraktion und umfangreiche Protokolle, wie TCP/IP und USB. Der Implementierungsaufwand lässt sich auf diese Weise mit geringen Performance-Einbußen bereits deutlich senken. Die Implementierung der Spezialhardware erfolgt in der Regel durch die Beschreibung mit einer Hardwarebeschreibungssprache, wie Verilog HDL oder VHDL. Zum Erzielen guter Ergebnisse wird dabei nicht nur die Kenntnis der verwendeten Sprache, sondern auch ein hohes Maß an Erfahrung mit dem Entwurf digitaler Systeme benötigt.

Abhängig vom Entwurfsverlauf kann zum Zeitpunkt der Spezifikation der Beschleunigungshardware bereits eine vollständige oder nahezu vollständige Softwareimplementierung vorliegen. Dies ist insbesondere dann der Fall, wenn die zeitkritischen Teile der Software durch eine Analyse des Laufzeitverhaltens ermittelt werden. An diesem Punkt knüpfen die im Folgenden vorgestellten Entwicklungswerkzeuge an. Sie ermöglichen die Generierung von Beschleunigungshardware aus der vorhandenen oder einer an das Werkzeug angepassten Softwarebeschreibung. Im vorliegenden Beitrag werden der Nios II C-to-Hardware Acceleration Compiler (C2H Compiler) [1] von Altera und der Riverside Optimizing Compiler for Configurable Computing (ROCCC) [2] bezüglich des Aufwands, der Geschwindigkeit und des Ressourcenbedarfs mit einer reinen Softwareimplementierung und dem klassischen Entwurf in VHDL verglichen.

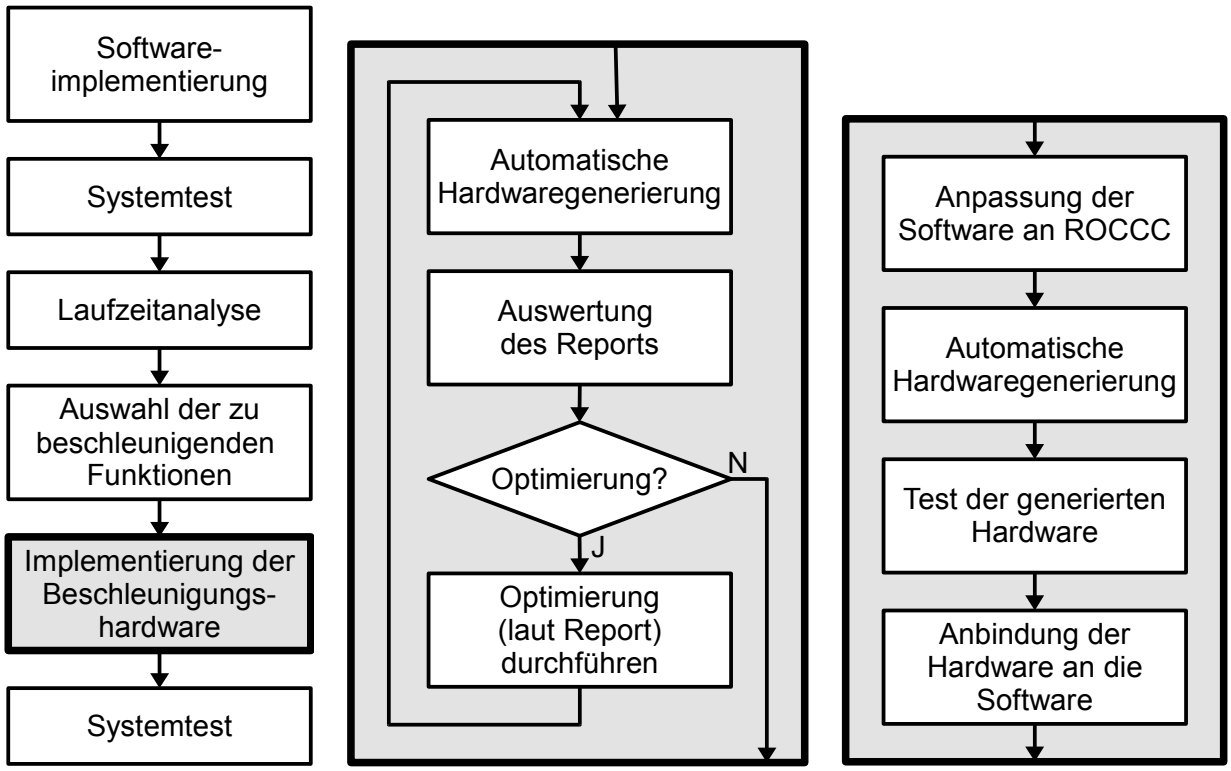
2 Versuchsumgebung und Entwurfsablauf

Der Vergleich erfolgt auf einem Altera DE2-Board. Der verbaute Cyclone II FPGA ist vom Typ EP2C35F672C6. Er enthält über 33000 Logikzellen und 420 Kibit dedizierten Speicher. Als Entwicklungssoftware wird Version 10.1 der Altera Werkzeuge verwendet, insbesondere Quartus II [3] für die Hardwarebeschreibung und Synthese, der SOPC Builder [4] zum Erstellen und Konfigurieren des Prozessors und die Nios II IDE [5] für die Entwicklung der Software. Letzte enthält eine komfortable Integration des C2H Compilers und wird deshalb an Stelle der Nios II Software Build Tools eingesetzt. Als Prozessor wird ein Nios II/e [6], also die kompakteste Ausführung, verwendet. Programm und Datenspeicher liegen im internen Speicher des FPGAs. Sowohl der Prozessor, als auch die Beschleunigungshardware arbeiten mit einem gemeinsamen Takt von 50 MHz. Für die Zeitmessung wird nicht die interne Funktion des Nios II verwendet, da diese auf Timerinterrupts basiert und somit eine Beeinflussung des Ergebnisses nicht ausgeschlossen werden kann. Stattdessen wird ein externer Taktzähler über eine eigene Hardwareabstraktionsschicht angesteuert. Der Zähler läuft taktsynchron und während der Messung vollständig unabhängig vom Prozessor. Die für die Kommunikation zwischen dem Zeitmesser und Prozessor benötigte Zeit wird während der Initialisierung ermittelt und bei der Ausgabe des Ergebnisses berücksichtigt.

Im vorliegenden Beitrag wird der Vergleich anhand eines Pseudozufallszahlengenerators geführt. Andere Algorithmen liefern (in absoluten Zahlenwerten ausgedrückt) andere Ergebnisse, die sich allerdings in den Tendenzen überwiegend deckungsgleich zeigen. Der Pseudozufallszahlengenerator basiert auf einem 128 Bit breiten, linear rückgekoppelten Schieberegister. Er erzeugt parametrisiert zwischen 1 Bit und 128 Bit pseudozufällige Daten. Diese können beispielsweise in der Verschlüsselungs- oder Nachrichtentechnik eingesetzt werden. Der Algorithmus lässt sich sehr effizient in Hardware abbilden, so dass ein gutes Syntheseergebnis erwartet werden kann.

Abbildung 1(a) zeigt den (vom Algorithmus unabhängigen) verwendeten Implementierungsablauf. Die Darstellung hebt den Bereich um die Generierung der Beschleunigungshardware hervor. Andere Details, wie zum Beispiel Rückführungen für die Fehlerkorrektur werden nicht dargestellt. Nach der vollständigen Implementierung der Software und erfolgreichem Systemtest wird eine Laufzeitanalyse durchgeführt, um die Funktionen zu ermitteln, die häufig aufgerufen werden oder viel Zeit für die Berechnung benötigen. Mit Hilfe der gewonnenen Informationen wird eine Auswahl der zu beschleunigenden Funktionen getroffen. Der weitere Ablauf für die Implementierung der Beschleunigungshardware ist abhängig vom verwendeten Verfahren.

C2H Compiler: Durch die Integration in die Entwicklungsumgebung ist die Verwendung des C2H Compilers sehr einfach. Der Ablauf wird in Abbildung 1(b) gezeigt. Die zu beschleunigenden Funktionen werden über die Entwicklungsumgebung ausgewählt. Die Übersetzung zu VHDL



(a) allgemein (b) C2H Compiler (c) ROCCC

Abbildung 1: Entwurfsabläufe

Code, sowie die Anbindung an den Prozessor über die Avalon Schnittstelle [7] und die transparente Integration in die Software erfolgen automatisch. Voraussetzung hierfür ist, dass die Beschreibung der zu beschleunigenden Funktionen in ANSI C unter Auslassung von für die Synthese kritischen Konstrukten vorliegt. Hierzu gehören zum Beispiel die Verwendung von Fließkommaarithmetik, Rekursion oder des Adressoperators (&). Eine Liste nicht unterstützter Konstrukte ist in Kapitel 7 des Nios II C2H Compiler User Guide [1] zu finden. Auf nur schlecht synthetisierbare Codeabschnitte wird nach der Übersetzung in einem Report hingewiesen. Dies ermöglicht eine Optimierung durch das Umschreiben des Codes oder die Verwendung von Pragmas. Auch nach der Generierung der Hardware können einzelne oder alle Beschleuniger zur weiteren Analyse in der Entwicklungsumgebung aktiviert und deaktiviert werden.

ROCCC: Abbildung 1(c) zeigt den Implementierungsablauf bei Verwendung des ROCCC. Da dieser herstellerunabhängig ist, fehlt die lückenlose Integration des C2H Compilers. Zudem ist nicht die Übersetzung nahezu beliebigen ANSI C Codes vorgesehen. Stattdessen muss eine Aufteilung der zu beschleunigenden Software in Module und Systeme erfolgen.

Module haben ausschließlich skalare Ein- und Ausgabeparameter. Die Skalare können Ganz- oder Fließkommazahlen sein, wobei für die Berechnung letzter externe Module eingebunden werden müssen. Gleiches gilt für Division und Rest bei Ganzzahlen. Die Beschreibung des Moduls erfolgt durch Implementierung einer C Funktion, deren Parameter den skalaren Ein- und Ausgabewerten entsprechen. Schleifen innerhalb vom Modulen werden vollständig ausgerollt. Dementsprechend entsteht bei der Synthese parallele Hardware. Unterfunktionen müssen ebenfalls als Module implementiert werden und können dann wie gewohnt aufgerufen werden.

Systeme können neben optionalen skalaren Ein- und Ausgabeparametern Datenströme empfangen und erzeugen. Diese werden innerhalb der C Funktion als Feld angesprochen. Die Operationen auf den Datenströmen erfolgen in einer Schleife. Diese Schleife wird nicht ausgerollt, wodurch die Berechnung sequenziell erfolgt. Eine teilweise oder vollständige Parallelisierung der Datenstromverarbeitung kann über eine Compileroption erreicht werden. Das Einbinden vorhandener Module ist wie zuvor beschrieben möglich.

Offensichtlich ist der ROCCC dafür ausgelegt, einfache Operationen wiederholt auf einem Datenstrom auszuführen. Damit ist er nicht für alle Probleme geeignet, jedoch für eine große Teilmenge der potenziell zeitkritischen. Der Test der generierten Hardware kann mit einer ebenfalls vom ROCCC erzeugten Testbench erfolgen. Hierfür muss lediglich ein Satz von Testdaten mit den erwarteten Ergebnissen eingegeben werden. Diverse Compileroptionen ermöglichen eine Einflussnahme auf das Syntheseergebnis. Eine Beschreibung ist unter den Punkten 3.8 und 4.8 des ROCCC 2.0 User's Manual [2] zu finden. Das Anbinden der Hardware an die Software erfolgt durch eine in C implementierte Hardwareabstraktionsschicht. Die für die physikalische Kommunikation benötigten Komponenten werden dem Prozessor mit Hilfe des SOPC Builders hinzugefügt.

Manuelle Implementierung: Die manuelle Implementierung der Beschleunigungshardware kann wie die bisher vorgestellten Verfahren in den Entwurfsablauf aus Abbildung 1(a) eingeordnet werden. Falls die Auswahl der zu beschleunigenden Funktionen auch ohne eine vorhergehende Laufzeitanalyse möglich ist, kann die Hardware auch parallel zur Software implementiert werden. Die Wahl der Schnittstelle zwischen Universalprozessor und Spezialhardware kann sich stark auf die resultierende Geschwindigkeit auswirken. Eine enge Kopplung kann zum Beispiel durch Einbinden als Befehlssatzerweiterung erreicht werden. Vorteil dieser Methode ist die geringe Verzögerungszeit für den Aufruf des Spezialbefehls. Damit ist sie vor Allem für kurze, jedoch häufig ausgeführte Befehle sinnvoll. Die Anbindung als Coprozessor ermöglicht eine lose Kopplung. Die Zeit für den Aufruf und den Transfer der benötigten Daten ist in der Regel höher als bei einer Befehlssatzerweiterung, allerdings ist der Universalprozessor während der Berechnung im Coprozessor nicht blockiert, was bei komplexen Befehlen von erheblichem Vorteil sein kann. Die Implementierung der Beschleunigungshardware erfolgt durch Beschreibung mit VHDL. Für die Simulation der Hardware wird ModelSim-Altera 6.6c eingesetzt. Die Anbindung an die vorhandene Software erfolgt wie für den ROCCC beschrieben.

3 Ergebnisse

Während der Durchführung der Versuchsreihe sind an verschiedenen Stellen Probleme aufgetreten. Die Übersetzung des Pseudozufallszahlengenerators mit dem C2H Compiler schlägt bei der Verwendung von Zeigern auf 64 Bit Integer reproduzierbar fehl. Eine Funktion mit der Signatur `uint32_t foo(uint32_t* a, uint32_t* b)` lässt sich problemlos übersetzen. Bei einer Funktion mit der Signatur `uint64_t bar(uint64_t* a, uint64_t* b)` entstehen Überschneidungen im Adressbereich des Prozessors, die Übersetzung bricht ohne Hinweis auf die tatsächliche Fehlerursache ab. Außerdem kam es bei der Beschleunigung mehrerer Funktionen, die auf einem globalen Speicherfeld arbeiten, teilweise zu einer Kollision im Namensraum des generierten VHDL Codes, die manuell aufgelöst werden musste.

Tabelle 1 zeigt für die angegebenen Verfahren jeweils die Dauer, die das Erzeugen eines Bits und das von 128 Bit benötigt, sowie den Faktor der Beschleunigung gegenüber der reinen Softwarelösung. Außerdem wird die Anzahl der Logikzellen, die von der Beschleunigungshardware benötigt werden, sowie der maximale Takt mit dem diese laut Timinganalyse arbeiten können angegeben.

Nr.	Verfahren	Dauer 1 bit	Dauer 128 bit	Faktor	Logikzellen	max. Takt	Aufwand
1	Software	19 µs	2127 µs	1	0	-	100%
2	ROCCC	9 µs	934 µs	2,3	751	179 MHz	125%
3	ROCCC 32	-	35 µs	60,8	10678	149 MHz	125%
4	C2H	5 µs	73 µs	29,1	1085	77 MHz	105%
5	C2H Optimiert	5 µs	10 µs	212,7	984	94 MHz	130%
6	Manuell	3 µs	5 µs	425,4	142	255 MHz	200%

Tabelle 1: Beschleunigung eines Pseudozufallsgenerators

Die letzte Spalte enthält eine subjektive Abschätzung des Aufwands ausgehend von der Annahme, dass die wesentlichen Probleme der Werkzeuge bekannt sind und von Anfang an umgangen werden. Der in dieser Versuchsreihe tatsächlich getriebene Aufwand ist aufgrund der Anfangs geringen Erfahrung mit den Werkzeugen nicht repräsentativ (tatsächlich benötigten alle Verfahren durch die auftretenden Probleme mehr Zeit als der manuelle Entwurf). Es sei in Erinnerung gerufen, dass in der gezeigten Messreihe die Spezialhardware taktsynchron zum Prozessor mit 50 MHz ausgeführt wird und nicht mit dem angegebenen, maximalen Takt.

Auffällig ist der erhebliche Geschwindigkeitsunterschied zwischen dem Ergebnis des ROCCC in Zeile zwei der Tabelle und den anderen Verfahren beim Schieben um 128 Bit. Grund hierfür ist, dass der Beschleuniger als Modul implementiert ist und entsprechend nicht mehrere Schiebevorgänge sequenziell ausführen kann. Die Iteration erfolgt in der Software. Eine Implementierung als System wurde nicht vorgenommen, da das Ergebnis des vorhergehenden Schiebevorgangs als Eingabe des folgenden benötigt wird. Damit ist eine Datenstromverarbeitung nicht sinnvoll möglich. Stattdessen wurde ein weiteres Modul implementiert, welches 32 Schiebeoperationen in einer Schleife ausführt. Das Ergebnis ist in Zeile drei zu finden. Die Verarbeitungsgeschwindigkeit steigt durch diese Maßnahme um den Faktor 27. Das automatische Ausrollen der Schleife durch den ROCCC führt allerdings zu einem um den Faktor 14 erhöhten Ressourcenbedarf.

Die mit dem C2H Compiler erzielten Ergebnisse sind in Zeile vier und fünf von Tabelle 1 zu finden. Die vorgenommene Optimierung besteht entsprechend dem Vorschlag des erzeugten Reports in der Einführung eines lokalen Speichers in der Beschleunigungshardware. Hierdurch wird der Datenaustausch mit dem Prozessor bei mehrfachem Schieben deutlich reduziert. Die Geschwindigkeit für die Erzeugung von 128 Bit konnte durch den manuellen Eingriff um den Faktor sieben gegenüber der vollständig automatisch generierten Hardware erhöht werden. Ebenfalls hat sich wider ersten Erwartens der Ressourcenbedarf um 101 Logikzellen verringert. Dies ist darauf zurückzuführen, dass die Komplexität der erzeugten Logik für den Datenaustausch mit dem Prozessor durch die Einführung des lokalen Speichers gesunken ist. Für die Erzeugung von 128 Bit benötigt der optimierte C2H Beschleuniger lediglich doppelt so lange, wie für ein Bit. Daraus lässt sich schließen, dass ein großer Teil der benötigten Zeit für den Aufruf des Beschleunigers über die Hardwareabstraktionsschicht, sowie den Datenaustausch benötigt wird und keine nennenswerte Beschleunigung durch weitere Optimierung zu erwarten ist. Das mit dem manuellen Entwurf erzielte und in Zeile sechs gezeigte Ergebnis bestätigt diese Vermutung.

In einem weiteren Test wurde das optimierte Ergebnis des C2H Compilers mit dem maximal möglichen Takt betrieben, also asynchron zum Prozessor. Die für die Synchronisierung benötigte Hardware wird ebenfalls automatisch generiert. Sie benötigt gemeinsam mit der PLL 366 zusätzliche Logikzellen. Die Generierung eines Bits benötigt mit 6 µs etwas länger als im synchronen Betrieb. Die Generierung von 128 Bit benötigt 8 µs und ist somit 20% schneller als im synchronen Betrieb.

Bezüglich des Ressourcenbedarfs ist ein deutlicher Unterschied zwischen dem manuellen Entwurf und der automatisch generierten Hardware zu erkennen. Der Bedarf des manuellen Entwurfs entspricht nahezu dem denkbaren Optimum. Das Schieberegister benötigt bereits 128 Flipflops. Der Zustandsautomat für die Steuerung und der Zähler für die sequenzielle Wiederholung des Schiebevorgangs benötigen zwei und sieben Register, womit insgesamt bereits 137 Zellen durch Register belegt sind. Die automatisch generierte Hardware benötigt ungefähr sieben mal so viel Platz. Bereits während der Implementierung der Software hat sich die mangelnde Ausdrucksstärke von C für die Beschreibung von Hardware gezeigt. Der größte auf der verwendeten Architektur verfügbare ganzzahlige Datentyp `unsigned long long` fasst 64 Bit. Ein größerer Datentyp wäre für einen 32 Bit Mikrocontroller auch kaum sinnvoll. Aufgrund der zuvor angesprochenen Probleme mit 64 Bit Integer Variablen bei Verwendung des C2H Compilers werden die Daten auf vier 32 Bit große Variablen verteilt gespeichert. Für das Schieben und Berechnen der Rückkopplung sind in C viele binäre Operationen und Maskierungen nötig. Diese werden bei der Übersetzung zu VHDL direkt übernommen, da der Compiler nicht die ursprüngliche Intention des Programmierers erkennt. Das Resultat ist eine deutlich kompliziertere Hardware als beim manuellen Entwurf.

4 Fazit

Abschließend kann gesagt werden, dass sich mit dem C2H Compiler und dem ROCCC (die nötige Erfahrung vorausgesetzt) bereits mit geringem Aufwand gute Ergebnisse erzielen lassen. Die vom ROCCC gezeigten Schwächen können auf den in diesem Beitrag verwendeten Algorithmus zurückgeführt werden. Bei der Implementierung eines FIR Filters als SIMD Befehl sind die Ergebnisse im Vergleich deutlich besser. Die Verantwortung für die vorausgehende Beurteilung der Eignung eines Algorithmus verbleibt beim Entwickler. Die Möglichkeiten bei nicht ausreichendem Ergebnis gezielt einzugreifen sind durch die teils mangelnde Ausdrucksstärke von C systembedingt beschränkt. Sicherlich könnte man fordern, dass die Compiler mehr und bessere Möglichkeiten bieten das Syntheseergebnis gezielt zu beeinflussen. Dies stünde allerdings im Widerspruch zur ursprünglichen Intention, die Beschleunigungshardware möglichst automatisch zu generieren. Wenn die volle Kontrolle gewünscht oder benötigt wird, zum Beispiel weil das Ergebnis der automatischen Synthese nicht ausreicht, ist ein manueller Entwurf die einzige vernünftige Alternative. Es bleibt zu wünschen, dass die Menge der nicht durch die Werkzeuge unterstützten Sprachkonstrukte weiter sinkt und die noch auftretenden Probleme, wie das Entstehen von Kollisionen im Namensraum des generierten Codes, behoben werden.

Literatur

[1] Altera, Inc.: Nios II C2H Compiler User Guide, November 2009, Version 1.6

[2] Jacquard Computing, Inc.: ROCCC 2.0 User's Manual, February 9 2011, Revision 0.6

[3] Altera, Inc.: Quartus II Handbook, December 2010, Version 10.1.0

[4] Altera, Inc.: SOPC Builder User Guide, December 2010, Version 1.0

[5] Altera, Inc.: Nios II Software Developer's Handbook, February 2011, Version 10.1.0

[6] Altera, Inc.: Nios II Processor Reference Handbook, December 2010, Version 10.1.0

[7] Altera, Inc.: Avalon Interface Specifications, August 2010, Version 1.3

Autorenverzeichnis

ISBN 978-3-8396-0259-1

www.eas.iis.fraunhofer.de