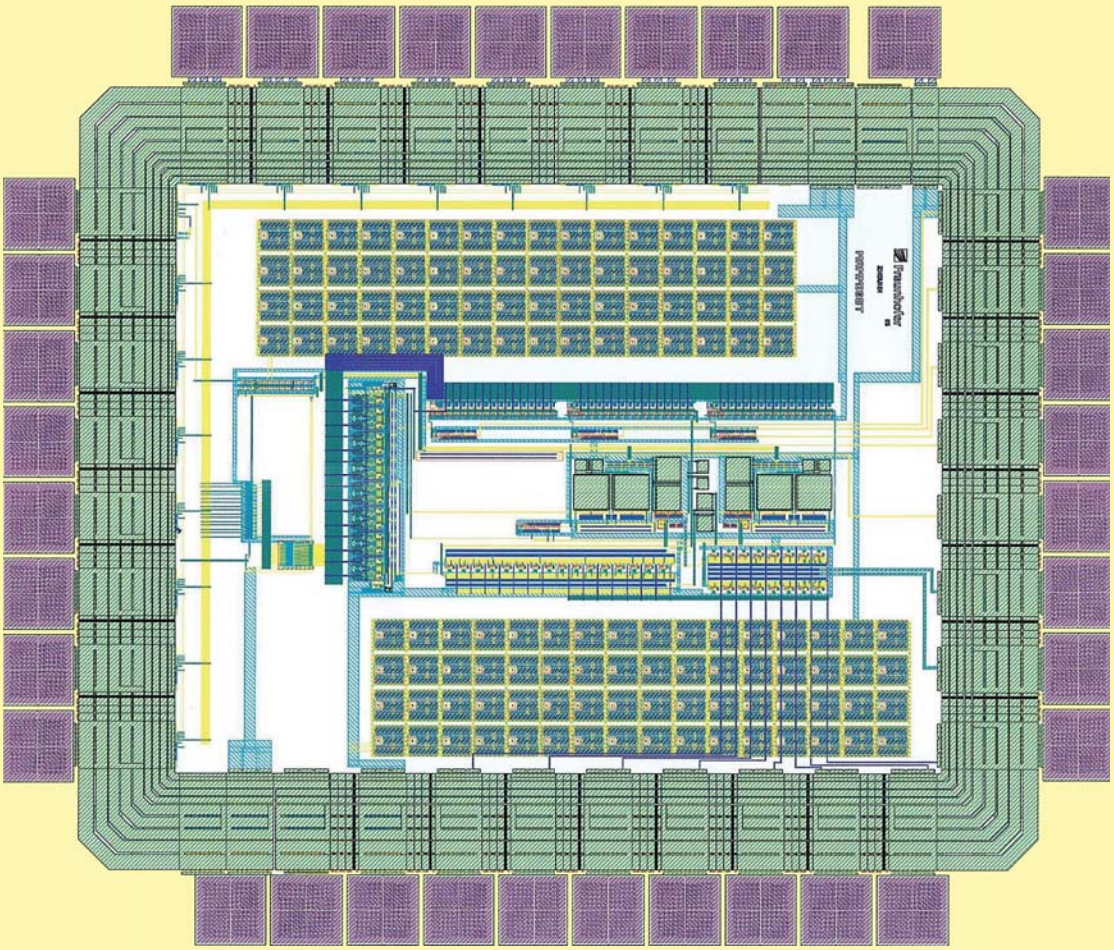
Fraunhofer
IIS

FRAUNHOFER-INSTITUT FÜR INTEGRIERTE SCHALTUNGEN IIS
INSTITUTSTEIL ENTWURFSAUTOMATISIERUNG EAS

DASS 2010
Dresdner Arbeitstagung Schaltungs- und Systementwurf

18. - 19. Mai 2010



FRAUNHOFER VERLAG

DASS 2010

Dresdner Arbeitstagung Schaltungs- und Systementwurf

18. – 19. Mai 2010

Tagungsband

Günter Elst (Hrsg.)

TECHNISCHE UNIVERSITÄT DRESDEN

Technische Universität Dresden
Fakultät Elektrotechnik und Informationstechnik
Fakultät Informatik

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS
Institutsteil Entwurfsautomatisierung EAS
Dresden

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

Layout/Bilder: Fraunhofer IIS

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: 978-3-8396-0126-6

Druck und Weiterverarbeitung:

IRB Mediendienstleistungen
Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart

Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

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
verlag.fraunhofer.de

Vorwort

Sehr geehrte Damen und Herren,

die in diesem Frühjahr stattfindende Dresdner Arbeitstagung Schaltungs- und Systementwurf bietet wie immer ein Forum zur Präsentation und Diskussion aktueller Arbeiten und Ergebnisse zum Entwurf sowie zur Entwurfsautomatisierung.
Darüber hinaus erfolgen Aussagen über die Entwicklung zukünftiger elektronischer und heterogener Systeme und die damit verbundenen Anforderungen an die Entwurfsunterstützung.

Die Arbeitstagung hat dieses Jahr das Schwerpunktthema

Technologiegerechter Entwurf:

Herausforderungen an den Entwurf für die Fertigung mit innovativen Technologien

gewählt. Ein Teil der Beiträge folgt diesem Thema und betrachtet die Abhängigkeiten des elektrischen Verhaltens von technologischen und strukturellen Einflüssen sowie Ansätze und Verfahren für einen Entwurf, der eine gewisse Robustheit gegenüber solchen Einflüssen aufweist.

Das Tagungsprogramm bietet den Teilnehmern wieder interessante Beiträge über neue Lösungen zum Entwurf komplexer Schaltungen und Systeme. Neben konstruktiven Verfahren und Verifikation werden Methoden des technologiegerechten Entwurfs unter den Bedingungen Robustheit, funktionelle Zuverlässigkeit, Rekonfigurierbarkeit, Selbstreparatur sowie spezielle Architekturen vorgestellt.

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

René Schüffny
Rainer Spallek
Ronald Tetzlaff
Günter Elst

Dresden, Mai 2010

Inhaltsverzeichnis

Seite

Vortragsreihe A

Requirements on Parameterized Layout Macrocells (Pcells) for a High Voltage Process and a Method for Realisation 7
André Lerch, Wolfgang Flohrer, Wolfgang Grimm, Jörg Doblaski - X-FAB Semiconductor Foundries AG Erfurt

Robustness Characterization of Standard Cell Libraries 13
André Lange, Lutz Muche, Joachim Haase - Fraunhofer IIS, Institutsteil EAS Dresden; Hendrik T. Mau - Globalfoundries Dresden

Ermitteln der Häufigkeitsverteilungen von Verzögerungszeiten digitaler Grundgatter infolge von Parameterschwankungen 19
Fabian Hopsch, Bernd Straube, Wolfgang Vermeiren - Fraunhofer IIS, Institutsteil EAS Dresden

Analog IP Porting: A Puzzle of Topology Conversion, Optimization and Verification 25
Udo Sobe, Enno Böhme, Dietmar Mörtl, Achim Graupner - ZMD AG Dresden; Michael Pronath - MunEDA GmbH München

Messverfahren zur Fehlerquellenanalyse im Bereich elektrisch leitfähiger Textilien 31
Ulf Wetzker - Fraunhofer IIS, Institutsteil EAS Dresden

Herausforderungen bei der Automatisierung des Layoutentwurfs bei dreidimensionalen heterogenen Systemen 37
Robert Fischbach, Jens Lienig, Tilo Meister - TU Dresden, Institut für Feinwerktechnik und Elektronik-Design

Ein Ansatz zur Modellierung CNT-basierter thermischer Vias für den effektiven Wärmetransport in elektronischen Schaltkreisen 43
Jörg Hertwig, Holger Neubert, Jens Lienig - TU Dresden, Institut für Feinwerktechnik und Elektronik-Design

Technologiegerechte Produktentwicklung von MEMS 49
Kai Hahn, Thilo Schmidt, Matthias Mielke, Rainer Brück - Universität Siegen, Institut für Mikrosystemtechnik; Dirk Ortloff - Process Relations GmbH Dortmund

Vortragsreihe B

Modellierung und Simulation zeitschlitzbasierter Netzwerke mit TrueTime 55
Christian Roßberg, André Froß, Daniel Froß, Ulrich Heinkel - TU Chemnitz, Professur Schaltkreis- und Systementwurf

EDADB - eine Infrastruktur zur Dokumentation und Wiederverwendung von Schaltungstopologien 61
Volker Boos - Institut für Mikroelektronik und Mechatronik Systeme GmbH Ilmenau

Modulares Framework für die synchronisierte Erfassung, Verarbeitung und Aufbereitung heterogener Sensornetzdaten 67
Matthias Vodel, René Bergelt, Matthias Glockner, Wolfram Hardt - TU Chemnitz, Professur Technische Informatik

Seite

Eingebaute Selbstreparatur zur Kompensation von Produktions- und Alterungsfehlern 73
Tobias Koal, Heinrich Theodor Vierhaus - BTU Cottbus, Lehrstuhl Technische Informatik

Möglichkeiten und Grenzen der Software-basierten Selbstreparatur in statisch geplanten superskalaren Prozessorarchitekturen 79
Mario Schölzel - BTU Cottbus, Lehrstuhl Technische Informatik

Eine Konzeption zu Energieeffizienz- und Performanceanalysen mittels Debug-Werkzeugen für eingebettete System-on-a-Chip 85
Andreas Gajda, Steffen Köhler - TU Dresden, Institut für Technische Informatik

Fehleranalyse mittels Trace-basierter Rekonstruktion von Prozessorzuständen 91
Andreas Gajda, Steffen Köhler, Rainer G. Spallek - TU Dresden, Institut für Technische Informatik

Vortragsreihe C

VLSI Implementation of a Heap Sorting Algorithm for Time Based Event Communication 97
Stefan Scholze, Stephan Henker, Johannes Partzsch, Christian Mayr, Rene Schüffny - TU Dresden, Institut für Grundlagen der Elektrotechnik und Elektronik

Sophisticated nvSRAM based Transponder Architecture provides novel Features for Field Programming and Safety 103
Andreas Scade, Stefan Günther - Anvo-Systems GmbH Dresden;
Hagen Grätz, Hans-Jürgen Holland - Fraunhofer IPMS Dresden

Hardware-Implementierung eines Partikelfilters zur Positionsbestimmung 111
Daniel Froß, Jan Langer, André Froß, Ulrich Heinkel - TU Chemnitz, Professur Schaltkreis- und Systementwurf

A Multibit Continuous-Time Delta-Sigma ADC for Direct Conversion of CdZnTe Detector Arrays 117
Matthias Völker, Haiyan Zhou, Johann Hauer - Fraunhofer IIS Erlangen

Rapid Prototyping mit FAUmachine am Beispiel einer VHDL-PCI-Soundkarte 123
Stefan Potyra, Matthias Sand, Volkmar Sieh, Dietmar Fey - Friedrich-Alexander-Universität Erlangen-Nürnberg, Lehrstuhl für Rechnerarchitektur

Ein retargierbarer, hochperformanter ISA-Simulator in Java 129
Marco Kaufmann, Thomas B. Preußer, Rainer G. Spallek - TU Dresden, Institut für Technische Informatik

Autorenverzeichnis 135

Requirements on Parameterized Layout Macrocells (Pcells) for a High Voltage Process and a Method for Realisation

André Lerch, Wolfgang Flohrer, Wolfgang Grimm, Jörg Doblaski
[andre.lerch|wolfgang.flohrer|wolfgang.grimm|joerg.doblaski]@xfab.com
X-FAB Semiconductor Foundries AG, Erfurt

Abstract

The continuously shrinking technology nodes in the semiconductor industry also lead to a rising effect of parasitics in the high voltage area. In addition, the development engineer is pushed to reduce the time to market and he has to estimate all parasitic effects as early as possible.

This article gives a short overview of the high voltage requirements of a modern high voltage process and shows a method to support the development engineer during the schematic entry and the layout process of the components. The extended functionality is based on an extended set of parameters. In order to guarantee that all combinations of parameters are design rule conform, a method of verification for this kind of cells is explained.

Introduction

The voltage limiting element of a process is mostly a parasitic bipolar transistor [1]. To reduce the influence of these devices in a high voltage process, additional biased rings and wells are used [2]. Additionally, it may be required to define voltage dependent design rules to achieve a higher voltage tolerance for the devices [3]. During the process of drawing the cells for production, the layout process, the layout engineer uses parameterized cells to generate very quickly a device with all layers according the process specification and design rules [4]. These parameterized cells, called pcells, are implemented in the wafer fabs (foundry) process development kit (PDK).

The starting point for the work was a 0.18µm process development kit of a typical technology. This technology should be extended to a high voltage process. To support the development engineer, the pcells in the PDK also had to be extended (see Fig. 1).

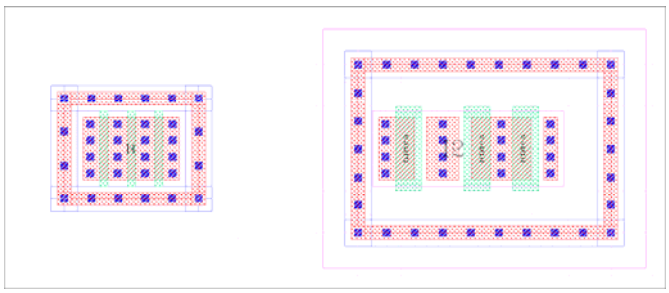


Figure 1: Layout Comparison of Typical MOSFET (left) and HV-MOSFET (right)

Each original parameterized cell was written as a closed element with its own code and with its own parameter setup. In detail the whole code was written for one cell in one file and the whole code was compiled into the final layout cell. The benefit of this method is, that each cell can be copied and renamed to any position and to any name. It is also very easy to extend the functionality of one cell. On the other hand the biggest disadvantage is the impossibility of handling the whole set of cells at once. Most functions are the same, or at least similar. To implement a function into a cell of the same category, e.g. MOSFET, Capacitor and Resistor, the code must be added by copy and paste. If a bug was found, all cells had to be corrected by hand. In addition, the internal standard of writing cells did not allow the use of subroutines. To fulfil the requirements of more complex analogue pcells a method with subroutines is needed.

Another weak point was the verification of the cell functionality itself. To detect a bug, the accepted method at this time was to create an example component and place a set of cells with

its most often used parameter sets by hand. This method was more a random test than systematic verification. The pcells were then delivered to the customer and showed in some cases runtime errors and design rule violation in the customer's environment. This kind of bugs could only be fixed with a long turn around time after creating a new process development kit (PDK) and deliver it to the customer.

At the end of the development a PDK with extensively verified pcells must exist. The task includes the creation of a widely reusable package of functions with a very easy configuration. Furthermore, the configuration should be easy transferred to other development platforms. And at the end, all cells must be verified with a method which prevents a failure during usage by the customer.

Requirements to the Pcells

In order to highlight pcell requirements it is useful to study the typical method of developing a component. One development engineer creates a schematic and simulates the behaviour with simulators. After the development of the schematic is finished, a layout is created by another development engineer. After finishing the layout all parasitics are extracted and the whole component will be re-simulated. Depending on the result the layout is often changed, re-extracted and simulated again until the behaviour fits to the requirements. The reason for this workflow is, the developer for the schematic might not care about physical limitations of the process. E.g. he could use a MOSFETs which is too wide, or resistor which is too long or too small during the first round of simulations. This could cause re-arrangement and splitting of components by the layout developer, and this again changes the parasitics. To reduce the turn around cycle and avoid crass misdimensioning of devices the developed pcells have implemented physical limitations. With a well known method, called callbacks, the parameter entry of each cell is supervised for its limits. The parasitics will be also re-calculated during this procedure. Together with the models, which must be able to handle these calculated parasitic parameters, the simulation can handle the parasitics as they will later be detected during layout extraction process.

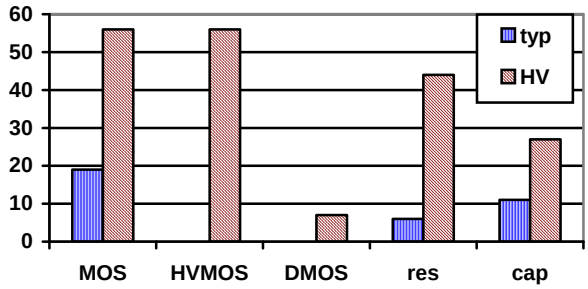


Figure 2: Comparison of Parameter Count of Cells for a Typical Process and for a High Voltage Process.

To fulfil the requirements of more complex pcells, even more input parameters are required. The number of parameters rose from below 20 in a simple MOSFET to above 50 for a more complex analogue pcell (see Fig. 2). According to the experience of the customer this amount of parameters can not be handled. Depending on the device application different parameters are in focus of the layouter's interest. In case of High Voltage applications the focus is on the distances of the interconnections, to achieve the required voltage stability. On the other hand, for power integration applications with high currents, the width of the metallisation and position of the contacts are more important than a reduction of parasitic capacitors.

To best support the layout engineer during parameter setup of the cells, the set of parameters which are visible to the layouter can be reduced by an application switch. This simple switch allows handling of all the basic options of a pcell like device length, width, splitting and simple inter device routing and all the parameters which are relevant for the intended application while hiding the rest. E.g. in the 'power' view, the dimensions of the metallisation can be changed to

achieve the requirements of high voltage or high current requirements. The layouter will be able to change parameters of the metallisation routing, e.g. space and width of the metal path.

The split of a component into many sub components requires a more detailed routing. Currently there are two methods of creating a layout. The first and traditional way is place and route the analogue cells by hand. For this method the routing up to the third metallisation layer was implemented. A well experienced layout engineer will be supported with device abutment, flight lines and selectable metallisation. In the second way, the automatic generation of an analogue cell layout, the pcells supports the tools with contact definitions and connectivity permutation.

Method of Code Development

In an earlier development an extended set of functions from an EDA software developer were used [5]. The disadvantage of the package was the closed source with a difficult setup. The new method had to provide an easy setup and a maximum of flexibility to the pcell programmer. This was achieved by using the following method.

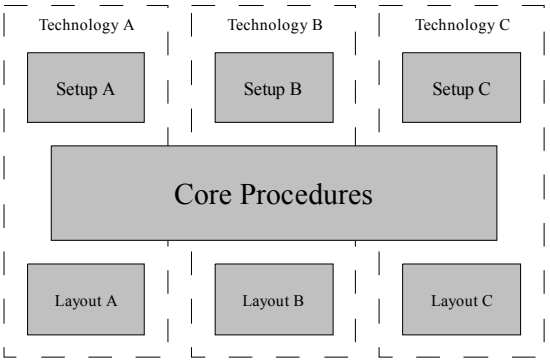


Figure 3: Abstract View of the Interactions

The code is split into three parts. These parts are the core, the setup and the layout implementation (see Fig. 3). Inside the core all functions are described with neither a relation to a technology nor a device. The core contains only information about the placement of the device components, e.g. Source-, Drain or Gate areas, resistor strips and the connecting metallisation (see Fig. 4). The design rule dependent functions for a given technology are imported from the user configuration in the setup. For the layout implementation it is required to map the components to a specific device using a description of the layout. The user has to generate all needed layers inside a function which is called from the core. This allows implementing new code into the cell and does not limit the functionality of the core to use predefined structures. A maximum of flexibility is given to the definition of the regions of a cell. With this method the programming of more or less complex devices is limited only by the experience of the programmer. Looking at the implementation in detail, all specific device regions, like drain, gate and source in case of a MOSFET, are generated by function calls from the core with the rectangle definition of their area. In addition to the area, information about the element count, the position inside the whole device as well as the information about a simple connectivity is provided by the function to the programmer. Together with the parameters in the setup, the core will place all elements in the right count with the right distance during the call of the pcell inside the customer's layout.

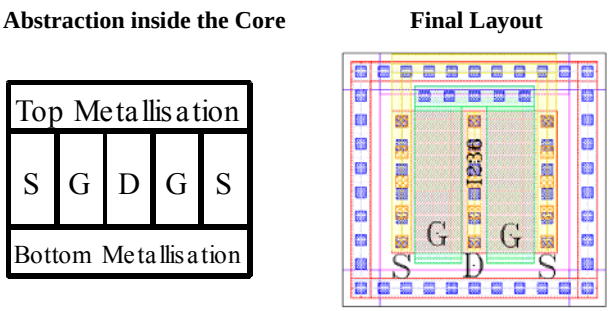


Figure 4: Display of Internal Components and Realisation in a Simple MOSFET

Apart from the aforementioned customer requirements, there are additional internal requirements to a PDK. The most important one is, to have simple pcells already available during the process development and device characterisation. The benefit of this is to characterise

the devices in the exact layout as they will be finally used. There will be no differences between the final PDK and the layouts used for characterisation purposes. An adaptation of parameters and dimension can be done for all required devices during three to five days if the process is sufficiently fixed and specified. The short time of conversion is only possible for processes with few differences to another already finished process with an existing PDK (e.g. a mostly similar process at a different technology node). The most important part is the contact system itself. A current state-of-the-art process allows using a contact system with fewer requirements in contact to via positioning. Much older processes without chemical-mechanical-polishing allow only congruent stacked contacts, or forbid stacked contacts generally. This causes an increase of programming effort to support this functionality to the pcells. Instead of implementing the full functionality of three level metals routing, the functionality of routing can at first be reduced to only one level metallisation.

Pcell Verification

The risen implemented functionality can only be handled with an extended method of verification. Using a visual and manual verification of the core creates lots of undetected problems during the phase of implementing other technologies. The typical extended MOSFET pcell has about 45 to 60 Parameters. Most of the parameters allow continuously ranged values. All other parameters have at least four selectable values. The test of all possible combination exceeds the possibilities of every commercial compute system. On the other hand, a manual, visual verification can not cover all required combinations of parameters. The chosen way is using a semiautomatic verification. The user defines a set of parameter values and the verification process generates from this setup all possible combinations (see also Fig. 5). Tests in the field shows a set of 110.000 combinations per cell reduce the verification time to an acceptable amount of time. A technology with approximately 50 different pcells can be verified by a cluster of 32 nodes during 24 hours.

In fact, this method does not cover all possible combination, but with the knowledge of the technology limitations and the knowledge of the integrated functionality inside the pcell core, the verification is reduced to an acceptable limit of compute time and human resources. The PDKs, which are verified with this method, did not show any failure of pcells at the customer's environment during extensive use for a large number of product designs over more than three years.

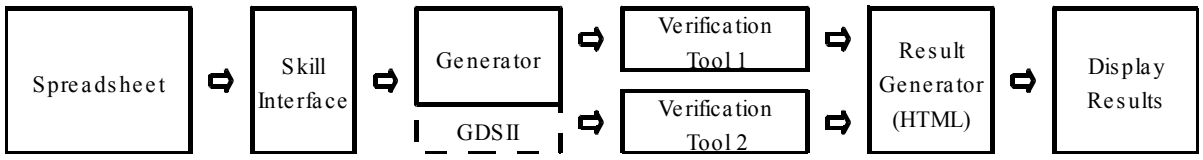


Figure 5: Verification workflow

How to configure and provide a human interface for this verification? How is it possible to check about 500 log files? To achieve a very easy handling for the person who runs the verification spreadsheets are set up for the different device groups. The user uses his or her experience to set up the spreadsheet with special definitions for parameter ranges. A converter generates from these special spreadsheets the Skill code which can be read into the verification environment. The user can now load the code into the verification software and can start to evaluate the generated devices with his preferred verification tool. During this verification process he is able to control the speed and license usage interactively. In case of license limitations he can reduce the amount of verification licenses used for the DRC/LVS process to not disturb other development processes inside the company. At the end of the verification all

log files are scanned for errors, warning and verification violations. The result can be displayed in a typical HTML browser. To use the already existing spreadsheet and browser software from the system reduces the programming effort of displaying software to a minimum. It was only required to program the interfaces from and to the verification engine. Additionally, the handling of the spreadsheet and displaying software is described in their respective documentations. It is not required to support documentation for using this kind of software.

Conclusion and Outlook

In this article, the requirements to state-of-the-art high voltage parameterized pcells are described. This includes the amount of required parameters as well as the method of generating a set of cells in a library. To verify such a complex set of pcells a method for verification on a network cluster was also introduced.

Another section, which requires similar complex functionality and consideration of parasitic elements, is the RF design. With an extension of the core it will be possible to support this kind of devices too. In opposite to the high voltage development, the parasitic capacitive devices are more interesting than diode leakage current.

The benefit of the quickly editable and convertible pcell library enables pcell programmers to provide libraries to other company departments early in the process development. It may be possible in the future to support the process characterisation with libraries for generating test fields. At this point the circle will be closed. The base for model characterisation will be done with the final pcells and this will create a very high correlation between measurement and simulation by using the identical elements for model characterisation and the layout.

Literature

[1] „The Art of Analog Layout", Prentice-Hall Inc., New Jersey 2001

[2] "Design Rule Specification XH18 – 0.18 µm Modular Mixed Signal HV CMOS", X-FAB Semiconductor Foundries AG, Erfurt, 2009

[3] "Design Rule Specification XDH10 - 1.0 µm Modular DIMOS 650V", X-FAB Semiconductor Foundries AG, Erfurt, 2009

[4] „Virtuoso XL Layout Editor User Guide", Cadence Inc., 2008

[5] „VCAD Productivity IP: Extensible Pcell Methodology (EPM)", Peter Herth and Olaf Zinke, Cadence, 2007

[6] "Interoperable PDK: Accelerating Analog EDA Innovation", Jingwen Yuan, Synopsys Inc. 2009, http://www.iplnow.com/tech_papers.php

[7] "Q&A: A Cadence View of 'Interoperable' PCells", Cadence, 2010, http://www.cadence.com/Community/blogs/ii/archive/2010/01/13/q-amp-a-a-cadence-view-of-interoperable-pcells.aspx

[8] "Interoperable PCell Libraries Or Bust!", Michael Ma, Chip Design Mag, http://chipdesignmag.com/display.php?articleId=924

Robustness Characterization of Standard Cell Libraries

André Lange, Lutz Muche, Joachim Haase
(andre.lange, lutz.muche, joachim.haase)@eas.iis.fraunhofer.de
Fraunhofer Institut für Integrierte Schaltungen, Institutsteil Entwurfsautomatisierung, Dresden

Hendrik T. Mau
hendrik.mau@globalfoundries.com
Globalfoundries, Dresden

Abstract

As feature sizes of integrated circuits shrink, fluctuations in electrical performance parameters drastically gain influence. They are especially important for standard cells as the basic components of digital designs, because they are instantiated numerously. In this paper, we present a methodology to statistically analyze performance parameter fluctuations and to abstract circuit robustness. As a result, we are able to determine the most critical cells in terms of performance parameter variability. These cells should either be optimized or rarely used in digital designs.

1 Introduction

As a basis for today's applications such as high-performance servers and mobile electronic devices, complexity and functionality of integrated circuits have dramatically increased for decades. One important keyword to steadily follow Moore's Law is scaling. To reduce feature sizes by a factor of about $\sqrt{2}$ per technology node, new materials and fabrication processes have been introduced. Recent developments have brought the $45\ nm$ node into mass production [5].
But this trend has also given rise to disadvantages. Besides the tremendous costs, variability has become a major issue. *Random Dopant Fluctuation*, *Line Edge Roughness* [2] and variations in dielectric properties [1] are examples for effects resulting from limited resolution in lithography and discreteness of charge and matter. Because these effects do not scale as feature sizes shrink, their relative impact grows. They cause fluctuations in device and interconnect characteristics which may translate into variability of integrated circuit performance. This trend is expected to continue when introducing $32\ nm$ and $22\ nm$ technologies.
Standard cells are the basic building blocks in digital integrated circuits. Depending on the circuit type the gate count may be very high, for instance up to 32 million in portable SoCs [5], and extreme performance parameter values become likely. Since malfunctions of only one cell may already hurt circuit performance or even cause functional failures, variations in standard cell performance parameters have to be considered in the design process.
Statistical gate-level analyses estimate the distributions of path delays and circuit power dissipation using derating factors [9] or the sensitivities of standard cell performance characteristics to device or process parameters ([3], [10]). However, determining the required information is costly. Therefore, we look for an alternative approach to abstract performance parameter distributions and to outline critical cells in terms of performance variability.
The remainder of this paper is organized as follows. In section 2 we propose a method to abstract the variability of a particular performance parameter based on Monte Carlo simulations. Limitations according to theory are presented before we combine the figures of merit for multiple performance parameters to a measure for cell robustness. We apply our approach to an example set of standard cells in section 3. Section 4 concludes this paper.

This work has been supported by the German Federal Ministry of Education and Research (BMBF) within the project 'CoolComputing' (Project ID 13N1083). The content is the sole responsibility of the authors.

2 Standard Cell Robustness

2.1 Preliminary Remarks

Performance parameters from different physical domains are of interest for most parts of integrated circuits, including standard cells. For our robustness characterization we focus on performance measures from the time, voltage and power domain, as summarized in Tab. 1. To obtain comparable results, we need to define a unit-free robustness measure.

Tab. 1: Performance parameters and physical domains for standard cell robustness characterization.

Performance parameter	Identifier	Physical domain	Performance parameter	Identifier	Physical domain
Cell delay	delay	Time	Leakage power	leakage	Power
Output transition	transition	Time	Noise immunity	noise	Voltage
Active power	power	Power	Setup / hold time	setup / hold	Time

We do not assume that specification boundaries are available because it is industry practice to design standard cells which are as fast and as energy saving as possible. Hence, for instance the parametric yield is not feasible for our purposes here.
We use Monte Carlo simulations in this work. Applying alternative sampling approaches and adapting the presented equations may be subject to future research.

2.2 Performance Parameter Robustness

In the following, let us assume we examine an arbitrary performance parameter y_j. Since it is subject to variation we describe it by the random variable Y_j with its probability density function (PDF) $f(y_j)$ and cumulative distribution function (CDF) $F(y_j)$. The sample size is denoted by N and $y_j^{(i)}$ represents the ith sample of the performance characteristic y_j.
To describe a performance parameter distribution, it is common to determine the mean value

$$\bar{y}_j = \frac{1}{N}\sum_{i=1}^{N} y_j^{(i)} \tag{1}$$

and the sample variance s_j^2 or its square root, the sample standard deviation s_j,

$$s_j^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(y_j^{(i)} - \bar{y}_j\right)^2. \tag{2}$$

In combination, both values yield the coefficient of variation [4]

$$CV_j = \left|\frac{s_j}{\bar{y}_j}\right|. \tag{3}$$

This unit-free measure may be interpreted as a relative variation. High values imply a large spread in the distribution which means a low robustness in our understanding.
But (1) to (3) do not pay attention to the shape of the underlying performance parameter distribution and the examples in Fig. 1 show that we need to include additional information. We use the principle of the hinge spread [4] from robust statistics but replace hinges by the inter-quartile range

$$IQR_j = Q_j(0.75) - Q_j(0.25). \tag{4}$$

This range, which contains the inner 50% of the performance parameter distribution, is defined using the quantile function, $Q_j = F_{Yj}^{(-1)}$. It can be applied to Monte Carlo analyses and alternative sampling techniques.

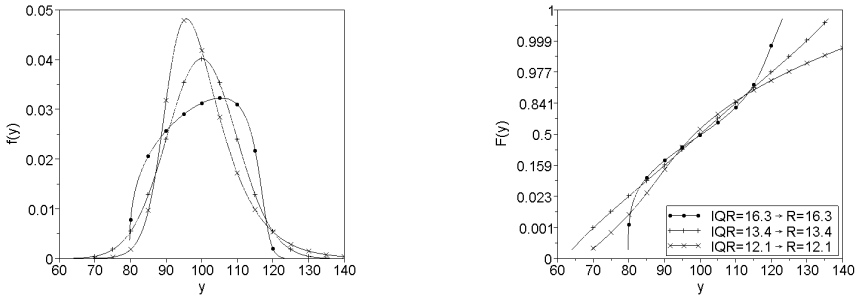


Fig. 1: Distribution functions with same mean value $\bar{y} = 100$ and standard deviation $s = 10$. Although the distributions have different shapes the coefficient of variation according to (3) cannot distinguish them. Our robustness approach in (5) punishes heavy-tailed distributions to enable distinctions.

A small value of the quotient IQR_j/s_j ($= 1.349$ for Gaussian distributions) indicates that the distribution of the performance characteristic y_j is heavy-tailed [4], which is underlined in Fig. 1. We interpret such a performance parameter as less robust. Combining the presented ideas, we propose the performance parameter robustness

$$R_j = \left| \frac{IQR_j \cdot \bar{y}_j}{s_j^2} \right|. \tag{5}$$

Since our approach punishes heavy-tailed distributions, a distinction is possible even in case of identical coefficients of variation CV_j, see Fig. 1.

Two problems arise from our definition in (5). On the one hand a performance parameter may be constant, that is $IQR_j = s_j = 0$. Then, the performance parameter y_j is not subject to variation which we define as infinite robustness, $R_j = \infty$. On the other hand, some performance parameter may have a zero mean, $\bar{y}_j = 0$. In this case, the performance parameter robustness is zero according to (5). We conclude that our robustness characterization approach is limited to performance parameter distributions with non-zero means. However, we believe this limitation is not of concern for standard cell performance parameters, as the ones defined in Tab. 1.

In addition to the internal cell configuration, logic states and external circuitry also define the performance characteristics in Tab. 1. Since these factors will not be known before using the gates in circuit design or operation, we apply a worst-case approach. For instance, leakage power robustness $R_{leakage}$ is the smallest robustness value for leakage in all possible logic states. In analogy, we determine the robustness for all remaining performance parameters in Tab. 1.

2.3 Cell Robustness

Since fluctuations in all performance parameters contribute to circuit variability, the single performance parameter robustness values from (5) are not sufficient to rate the robustness of a standard cell. We propose the cell robustness as their combination,

$$\frac{1}{R_{cell}} = \frac{1}{R_{delay}} + \frac{1}{R_{transition}} + \frac{1}{R_{power}} + \frac{1}{R_{leakage}} + \frac{1}{R_{noise}} + \frac{1}{R_{setup/hold}}, \tag{6}$$

that is dominated by the least robust performance parameters. If one of the performance parameters in Tab. 1 is missing for a specific cell, the respective performance parameter robustness will be set to ∞. For instance, $R_{setup/hold} = \infty$ will hold for all combinational cells.

Since there are various manufacturing processes, for example for low power mobile applications or high performance servers, the performance parameters may be of different importance. Introducing application specific weight factors into (6) may be a future enhancement.
By adding or adapting the performance parameters in Tab. 1 our robustness characterization proposed in (5) and (6) may be applied to further components of integrated circuits, for instance SRAM cells or analogue circuits, as long as the constraint of non-zero means is respected.

2.4 Confidence Interval Estimation

Due to our sampling based approach mean values, variances and inter-quartile ranges are only estimators for the exact quantities. Hence, the performance parameter robustness in (5) and the cell robustness in (6) are estimators as well. To increase the significance of our analysis results, we replace point estimation by interval estimation.
Methods to determine confidence intervals and the distributions for means, variances and inter-quartile ranges are available [4]. We use them for bootstrapping [7]. That is, we draw sets of means, variances and inter-quartile ranges and apply (5) and (6) to calculate sets of performance parameter robustness and cell robustness measures. We use them to determine the distributions and confidence intervals of our estimated robustness quantities.

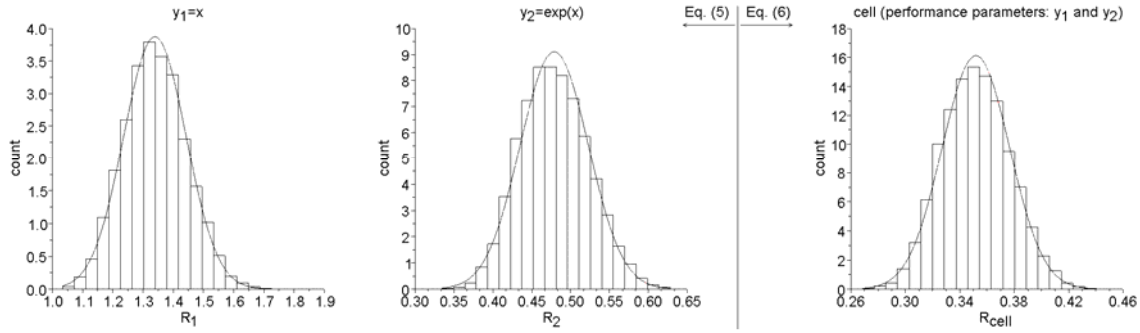


Fig. 2: Distributions of estimated robustness measures for a theoretical example with $X \sim N(1,1)$. Gaussian approximations (solid lines) are feasible.

Let us consider an example "cell" with two performance parameters y_1 and y_2. Fig. 2 shows the distributions of the estimated robustness values. Since they are approximately Gaussian, it is sufficient to describe them by their estimated mean values $\bar{R}_j$ and standard deviations $s_{\overline{R}j}$. With a probability $(1-\alpha)$, the performance parameter robustness R_j belongs to the interval defined by the subsequent equation

$$\bar{R}_j - z_{1-\alpha/2} \cdot s_{\overline{R}j} \leq R_j \leq \bar{R}_j + z_{1-\alpha/2} \cdot s_{\overline{R}j}, \tag{7}$$

where $z_{1-\alpha/2}$ is the $(1-\alpha/2)$-quantile of the standard Gaussian distribution [7]. Increasing the sample size N reduces the estimated standard deviation $s_{\overline{R}j}$ in (7).

2.5 Handling Extreme Values

In contrast to inter-quartile ranges IQR_j, mean values $\bar{y}_j$ and variances s_j^2 are not robust against extreme values and outliers [7]. Already one extreme simulation result may falsify our robustness calculation so that we need to handle extreme simulation results carefully. Leakage power is especially affected because of its approximately lognormal distribution whereas minimum and maximum values can differ by more than an order of magnitude [6].
The David-Hartley-Pearson test and Grubbs test are methods for identifying outliers [4]. But since they relate sample results $y_j^{(i)}$ to the non-robust mean values $\bar{y}_j$ and standard deviations s_j, we apply a different approach: Trimming maximum and minimum simulation results.

Let us consider the performance parameters in Fig. 2. We examine the agreement of 1000 robustness interval estimations at a significance level $\alpha = 0.05$ applying (7), each using $N = 1000$ samples, with a reference $R_{j,ref}$ determined with the sample size $N_{ref} = 1e6$. Thereby, we trim the extreme values, whereas p is the fraction of data discarded at each distribution tail.

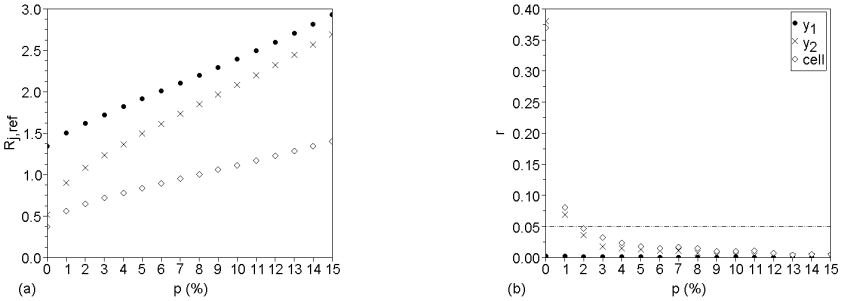


Fig. 3: Trimming extreme values when estimating performance parameter robustness. Dependence of the fraction p of trimmed data per tail on (a) reference parameter robustness $R_{j,ref}$ and (b) error rate r.

Since trimming extreme values reduces the variance, the reference robustness $R_{j,ref}$ rise with the fraction p in Fig. 3 (a). This especially effects the non-Gaussian performance parameter y_2. Its robustness $R_{2,ref}$ approximately doubles (triples) when rising p from 0 to 0.03 (0.05). The difference between the performance parameter robustness values decreases with the fraction p, reducing the discriminatory power of our approach in (5). Cutting data means an overhead in simulation costs which is a further reason for keeping the fraction p as small as possible.
Fig. 3 (b) shows the error rate r in dependence of the fraction p. It expresses the portion of intervals determined using (7) which do not include the reference robustness value. There may be errors due to our statistical approach but the error rate r should be at most the significance level α, $r \leq \alpha = 0.05$. In Fig. 3 (b), the smallest fraction fulfilling this constraint is $p = 2\%$. It is the best trade-off and hence we apply it in the practical application.

3 Practical Application

We implemented Perl scripts to parse the simulation data written by an industry standard circuit simulator. The robustness calculations are carried out in Scilab [8].

Tab. 2: Set of standard cells for robustness characterization application example.

i	1	2	3
Cell name	bufx2	inx1	inx2
Remark	2x buffer	1x inverter	2x inverter

We apply our approach to the three standard cells in Tab. 2 which represent an excerpt of a standard cell library. The sample size is $N = 1000$.
Fig. 4 (a) shows the results of analyzing the buffer. They are representative of all cells in Tab. 2. We see that leakage power has the least robustness, $R_{leakage}(bufx2) = 2.4$, which means it is most critical in terms of variation and dominates the cell variability. The cell delay, $R_{delay}(bufx2) \approx 4 \cdot R_{leakage}(bufx2)$, and active power also contribute to circuit variability whereas the robustness of transition time and noise immunity are more than one order of magnitude larger.
We compare the cells from Tab. 2 in Fig. 4 (b). The 2x buffer is most robust which fits our expectations because variations may average out when the number of devices grows. The inverters cannot be ranked because the robustness confidence intervals overlap. Further simulations are required to reduce the width of the robustness confidence intervals.

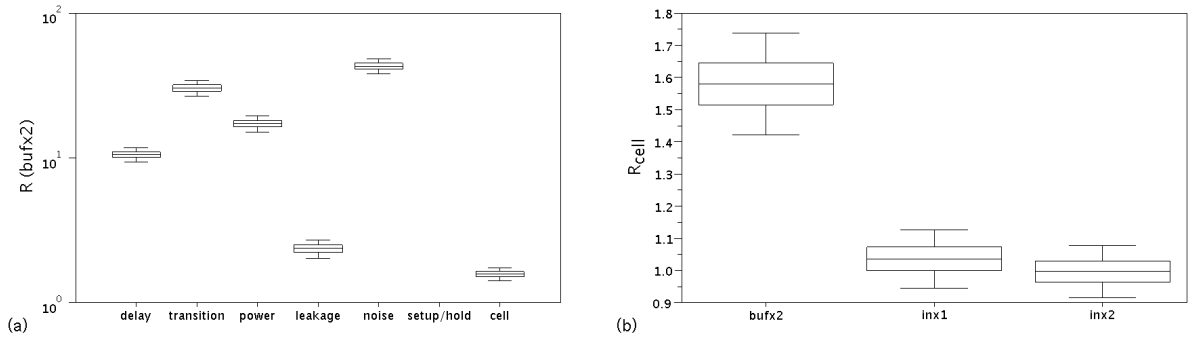


Fig. 4: Robustness characterization results. (a) 2x buffer bufx2. Note the logarithmic scale for robustness measures. $R_{setup/hold} = \infty$, since setup and hold times are not of concern in combinational logic. (b) Comparison of overall cell robustness.

4 Summary and Outlook

In our work we have presented an approach to analyze statistical circuit simulation data to abstract both performance parameter robustness and overall circuit robustness.

Although we have shown the principle applicability, there are still some open issues. First, the sample size $N = 1000$, that we have applied in our example, may be too costly to completely analyze a standard cell library. Second, we are currently not able to separate intra-die and inter-die effects because we do not link the performance parameter distributions to the device or process parameters causing fluctuations. Third, up to now, our robustness measures cannot be applied in gate-level analyses.

Nevertheless, our approach enables circuit comparison to outline critical standard cells in terms of variability. The interim results, mean values and standard deviations, may be used to derive timing derates during technology development. Apart from the constraint of non-zero means, our approach can be easily adapted to further types of circuitry including RAM and analogue blocks.

References

[1] A.R. Brown, J.R. Watling, A. Asenov: Intrinsic parameter fluctuations due to random grain orientations in high-k gate stacks. Journal of Computational Electronics, Volume 5, Number 4, Springer Netherlands, Dec. 2006.

[2] Cheng, B.; Roy, S.; Brown, A.R.; Millar, C.; Asenov, A: Evaluation of intrinsic parameter fluctuations on 45, 32 and 22nm technology node LP N-MOSFETs. 38th European Solid-State Device Research Conference, Sept. 2008.

[3] U. Eichler, J. Haase, R. Häußler, H. Kinzelbach: Gate-level digital power simulation with varying technology parameters. Semiconductor Conference Dresden, Apr. 2008.

[4] J. Hartung: Statistik – Lehr- und Handbuch der angewandten Statistik, 12. Auflage, R. Oldenbourg Verlag, München, 1999.

[5] International Roadmap for Semiconductors, 2009 edition.

[6] H. Kinzelbach: Timing- and Leakage-Variability: Analysis and Modeling Activities at Infineon. Vortrag auf dem Workshop über Simulation und Analyse statistischer Größen in Digitalschaltungen und SRAM, TU München, Apr. 2008.

[7] J.P. Marques de Sá: Applied Statistics Using SPSS, STATISTICA, MATLAB and R. 2nd edition, Springer, Berlin, 2007.

[8] http://www.scilab.org

[9] Synopsys: PrimeTime Advanced OCV Technology – Easy-to-Adopt, Variation-Aware Timing Analysis for 65-nm and below. http://www.synopsys.com, Apr. 2009.

[10] C. Visweswariah et. al.: First-Order Incremental Block-Based Statistical Timing Analysis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 25, Number 10, Oct. 2006.

Ermitteln der Häufigkeitsverteilungen von Verzögerungszeiten digitaler Grundgatter infolge von Parameterschwankungen

Fabian Hopsch, Bernd Straube, Wolfgang Vermeiren
{fabian.hopsch | bernd.straube | wolfgang.vermeiren}@eas.iis.fraunhofer.de
Fraunhofer IIS/EAS Dresden

Kurzfassung

In künftigen Technologien wird der Einfluss von Parameterschwankungen auf das Verhalten nanoelektronischer Schaltungen immer stärker. Um unter solchen Bedingungen eine hohe Produktqualität zu erreichen, werden statistische Methoden beim Entwurf und Test benötigt. In dieser Arbeit werden elektrische Fehlersimulationen unter Parameterschwankungen vorgestellt. Die auf der elektrischen Ebene erzielten Ergebnisse werden auf die Logikebene abstrahiert. Es wird eine Bibliothek für digitale Grundgatter aufgebaut. Von den Projektpartnern wird sie für Analysen und Simulationen auf der Logikebene verwendet, wie zum Beispiel für die statistische statische Timing-Analyse.

1 Einleitung

Das Auftreten von Defekten und unzulässigen Parametervariationen verursacht fehlerhaftes Verhalten nanoelektronischer Schaltungen. Dies verschlechtert sich für die immer kleiner werdenden physikalischen Strukturen künftiger CMOS-Technologien [1]. Einerseits gibt es defekt-orientiertes Testen, das die Nachteile des Einsatzes von klassischen Fehlermodellen zu bewältigen versucht [2]. Zum Anderen gibt es eine Entwicklung hin zu statistischen Methoden, um die Parameterschwankungen beim Entwurf zu berücksichtigen [3].

Der Einfluss von Parametervariationen auf das Verhalten fehlerbehafteter Schaltungen ist bereits in früheren Arbeiten analysiert worden. Das Ermitteln der Fehlerüberdeckung unter Annahme von Schwankungen von Bauelementeparametern für analoge [4][5][6][7][8][9][10] bzw. Mixed-Signal Schaltungen [11] steht dabei im Vordergrund. In den Arbeiten [12] und [13] werden statistische Methoden zur Analyse des Leckstromes und der Verzögerungszeit von Schaltungen vorgestellt.

In dieser Arbeit werden Fehlersimulationen auf der elektrischen Ebene für digitale Grundgatter aus einer Zellbibliothek vorgestellt. Dabei werden durch den Herstellungsprozess bedingte Parametervariationen berücksichtigt. Die Ergebnisse werden durch die Partner des Projektes RealTest für Simulationen und Analysen auf der Logikebene verwendet. Dafür werden die auf der elektrischen Ebene erzielten Resultate auf die Logikebene abstrahiert. In dieser Arbeit wird dabei die Verzögerungszeit der Gatter verwendet. Die Ergebnisse werden in einer Bibliothek zusammengefasst.

Die Ergebnisse können zum Beispiel als Histogramme oder Häufigkeitsverteilungen aufbereitet werden. Sie können damit für die statistische statische Timing-Analyse (SSTA) genutzt werden [14]. In kommerziellen EDA-Flows wird derzeit die statische Timing-Analyse (STA) eingesetzt, um kürzeste und längste Pfade in einer Schaltung zu finden und damit das Timing der Schaltung zu überprüfen. Dies ist allerdings ein sehr konservativer Ansatz, bei dem, vor allem durch die höhere Komplexität der Variation, viel Schaltungsperformanz verschenkt wird. Bei der SSTA werden anstelle von Verzögerungszeiten der Gatter Häufigkeitsverteilungen der Verzögerungszeit verwendet. Damit bietet die SSTA auch eine Möglichkeit zur Berechnung der Produktionsausbeute einer Schaltung. Die klassische SSTA berücksichtigt herstellungsbedingte und umgebungsbezogene Variationen von Parametern. Im Rahmen des Projektes RealTest sollen dabei auch die Auswirkungen von Defekten betrachtet werden.

Diese Arbeit wurde im Rahmen des von der Deutschen Forschungsgemeinschaft (DFG) geförderten Projektes RealTest − Test und Zuverlässigkeit nanoelektronischer Systeme (Fkz: Str412/1-2) durchgeführt; http://realtest.date.uni-paderborn.de.

Die Arbeit ist folgendermaßen gegliedert. Im nächsten Abschnitt wird die Fehlersimulation unter Parameterschwankungen auf der elektrischen Ebene dargestellt. Anschließend wird im Abschnitt 3 die Fehlersimulation am Beispiel eines NAND2-Gatters veranschaulicht. Die Ergebnisse werden in Abschnitt 4 vorgestellt. Abschnitt 5 fasst die Arbeit zusammen und gibt einen Ausblick auf weiterführende Arbeiten.

2 Elektrische Fehlersimulation unter Parameterschwankungen

Fehlersimulationen werden auf verschiedenen Abstraktionsebenen eingesetzt. In dieser Arbeit werden sie auf der elektrischen Ebene durchgeführt. Dabei werden in die elektrische Netzwerkbeschreibung einer Schaltung Fehler injiziert. Fehler sind Modelle, die die Auswirkungen von in der Schaltung auftretenden Defekten repräsentieren. Fehlersimulationen werden für vielfältige Aufgaben eingesetzt. Dazu gehören die Fehlererkennung und das Bestimmen der Fehlerüberdeckung eines gegebenen Tests. Weiterhin wird die Fehlersimulation bei der Testsignalentwicklung und bei der Diagnose eingesetzt.

Für die Fehlersimulationen wird der gegenüber [15] erweiterte elektrische Fehlersimulator aFSIM eingesetzt. Dieser Fehlersimulator setzt auf unterschiedlichen Netzwerksimulatoren, wie z. B. SPICE, Spectre, Eldo oder TITAN (Infineon), auf. In Abbildung 1 ist der Fehlersimulator aFSIM schematisch dargestellt. Als Eingabedaten werden eine elektrische Beschreibung der Schaltung in Form einer Netzliste, eine Liste S von Eingangssignalen, eine Liste P mit Parameterwerten, eine Fehlerliste F und Auswertekriterien benötigt. In der Netzliste werden automatisch die Eingangssignale und die Parameterwerte gesetzt und die Fehler injiziert. Die $|S|*|P|*|F|$ modifizierten Netzlisten werden automatisch auf verteilte Server oder ein HPC-Cluster verteilt. Für die Simulation wird der entsprechende Netzwerksimulator verwendet. Die Resultate der Simulationen werden gemäß der Auswertekriterien ausgewertet und die Ergebnisse in einer Datenbank gespeichert.

In den Listen gibt es für die einzelnen Elemente nur Einschränkungen bezüglich der Syntax des verwendeten Simulators. In der Liste der Eingangssignale ist jede vom verwendeten Simulator unterstützte Signalquelle zulässig. Über die Parameterliste können alle in der Schaltung vorkommenden Parameter geändert werden. Dazu gehören sowohl Device- als auch Modellparameter. Die Variationen können global oder lokal sein. In der Fehlerliste wird beschrieben, welche Modifikationen der Netzliste vorgenommen werden, um die elektrischen Auswirkungen potentiell in der Schaltung auftretender Defekte zu beschreiben. Dabei besteht die Möglichkeit, in der Netzliste vorhandene Elemente zu ändern oder zu entfernen und neue Bauelemente hinzuzufügen.

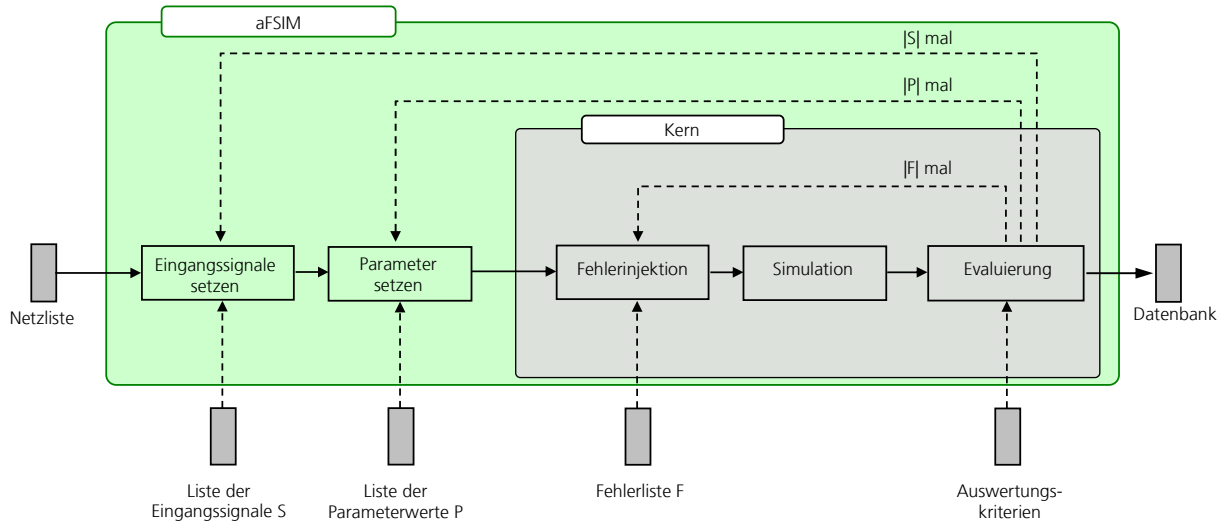


Abbildung 1: Schema des Fehlersimulators aFSIM; Simulationen werden automatisch verteilt

3 Fehlersimulation für ein NAND2-Gatter

Elektrische Fehlersimulationen unter Parameterschwankungen wurden für einige digitale Grundgatter aus der Nangate 45 nm Open Cell Library (OCL) [16] mittels des Fehlersimulators aFSIM durchgeführt. Untersuchungen erfolgten für die Grundgatter INV, NAND2, NOR2 und XOR2. Die Vorbereitung und Durchführung der Fehlersimulation unter Parameterschwankungen wird im Folgenden am Beispiel des NAND2-Gatters beschrieben.

Für die Fehlersimulation wird ein elektrisches Modell der Schaltung benötigt. In dieser Arbeit werden aus dem Layout extrahierte Netzlisten verwendet. Damit ist ein sehr realitätsnahes Modell der Schaltung vorhanden, dass gleichzeitig die Möglichkeit bietet, an potentiellen Defektorten Fehler zu setzen. Die extrahierte Netzliste des NAND2-Gatters besteht aus vier Transistoren, 26 Widerständen, die Leitungen repräsentieren, und 16 Kapazitäten zum Substrat. Das Gatter wird für die elektrischen Simulationen in eine Umgebung eingebettet. An den Eingängen *A1* und *A2* werden Treiber, bestehend aus zwei in Reihe geschalteten Invertern, eingefügt. Am Ausgang *Out* wird eine kapazitive Last C_L angeschlossen, die die mögliche Belastung durch nachfolgende Gatter repräsentiert. Für die ersten Untersuchungen wird ein Wert aus der OCL von C_L=0.4 fF verwendet. In Abbildung 2 ist das eingebettete NAND2-Gatter ohne die parasitären Elemente dargestellt.

In den hier vorgestellten Untersuchungen wird die Aufzeichnung der Verzögerungszeit als Auswertekriterium verwendet. Sie wird als die zeitliche Differenz von Ausgangssignal bei halber Versorgungsspannung und Eingangssignal bei halber Versorgungsspannung definiert [17]. Eine Verzögerungszeit kann somit nur berechnet werden, wenn beim Eingangssignal und am Ausgangssignal ein Signalwechsel stattfindet. Bei Testsequenzen der Länge zwei treten Signalwechsel auf. Für das NAND2-Gatter gibt es 16 mögliche Testsequenzen der Länge zwei. Sechs dieser Testsequenzen stellen einen Signalwechsel beim Eingang dar und erzeugen einen Signalwechsel am Ausgang. Sie sind in Tabelle 1 angegeben. Die Eingangssignale werden durch stückweise lineare Quellen beschrieben. Als Anstiegszeit wird ein Wert von 7.5 ps aus der OCL verwendet.

Die Liste der Parameterwerte wird aus Daten der typischen, langsamen und schnellen Technologieecken der OCL erzeugt. Die Ecken zeichnen sich durch entsprechende unterschiedliche Werte für die Bauelementeparameter Längenreduktion L_{INT}, Schwellspannung V_{TH0}, Bulkeffektkoeffizent K_1, Mobilität bei geringem Feld μ_0 und Eindringtiefe X_J aus. Die Verteilung der Parameterwerte wird als normalverteilt angenommen. Die Parameterwerte der typischen Technologieecke werden als Mittelwert und die Werte der langsamen und schnellen Technologieecke als 3σ-Werte der Normalverteilung verwendet. Zusätzlich wird noch anhand der Daten eines Industriepartners die Kanallänge *L* und die Oxiddicke *TOX* variiert. Die sieben angegebenen Parameter bestehen für n- und p-Kanal-Transistoren. Für die 14 Parameter werden jeweils 10 000 Parameterwerte entsprechend der Verteilungen ausgewählt. Damit besteht die Liste der Parameterwerte aus 10 000 Vektoren der Dimension 14.

In der Fehlerliste sind Einzelfehler der Kategorie Kurzschlüsse zwischen zwei Leitungen und Unterbrechungen von einzelnen Leitungen enthalten. Die Fehler werden nur im NAND2-Gatter

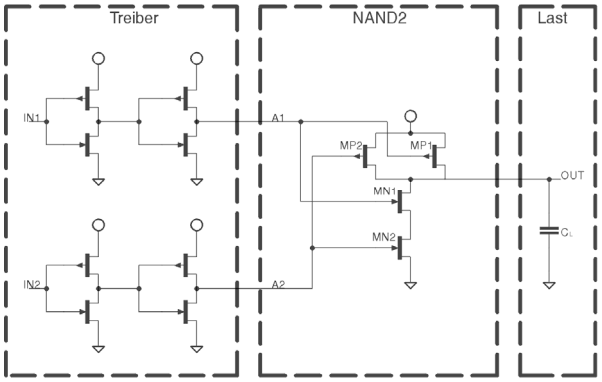


Abbildung 2: Schaltbild des eingebetteten NAND2-Gatter

Tabelle 1: Signalwechsel am Ausgang erzeugende Eingangsbelegungen der Länge 2

Eingangsbelegung				Ausgangswert	
t_n		t_{n+1}		t_n	t_{n+1}
IN1	IN2	IN1	IN2	OUT	OUT
0	0	1	1	1	0
0	1	1	1	1	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1

gesetzt. Die Umgebung wird als fehlerfrei angenommen. Kurzschlüsse werden zwischen den einzelnen Transistoranschlüssen injiziert. Weiterhin wird nach Berücksichtigung des Layouts noch ein Kurzschluss zwischen den beiden Eingänge *A1* und *A2* eingesetzt. Die Kurzschlüsse werden durch Einfügen eines Widerstandes modelliert, da während der Produktion auftretende Defekte sich zum Beispiel als nicht ideale Kurzschlüsse zeigen. Jeder Kurzschluss wird mit zehn verschiedenen Widerstandswerten aus dem Bereich von 10 Ω bis 15 000 Ω modelliert. Unterbrechungen werden durch Ersetzen der parasitären Leitungswiderstände durch höherohmige Widerstände modelliert. Es werden zehn verschiedene Widerstandswerte aus dem Bereich von 100 kΩ bis 100 MΩ verwendet. In der Netzliste sind 26 parasitäre Leitungswiderstände enthalten. Einige davon sind in Reihe geschaltet, so dass elf als Unterbrechungsstellen verwendet werden. Die Fehlerliste enthält somit 240 Fehler, die sich aus 130 Kurzschlüssen und 110 Unterbrechungen zusammensetzen.

Der Simulationszeitraum beträgt 20 ns, wobei bei 10 ns der Signalwechsel am Eingang stattfindet. Anhand der Elemente der Listen entstehen $|S|*|P|*|F|=6*240*10\,000=14\,400\,000$ modifizierte Netzlisten. Die Verteilung der Simulationen wird vom Fehlersimulator aFSIM automatisch durchgeführt. Unter Verwendung von 32 CPUs eines HPC-Clusters werden etwa zehn Tage benötigt.

4 Ergebnisse

Aus den Simulationen gehen zwei verschiedene Kategorien von Ergebnissen hervor. Zum einen gibt es Simulationen mit einem Signalwechsel am Ausgang. Bei diesen kann eine Verzögerungszeit bestimmt werden. Zum anderen gibt es Simulationen, bei denen im Beobachtungszeitraum kein Signalwechsel am Ausgang stattfindet. Dies entspricht einem Verhalten wie bei einem Haftfehler, dass auch als unendliche Verzögerungszeit interpretiert werden kann. In Tabelle 2 sind die Daten der vier simulierten Grundgatter zusammengefasst. Dargestellt sind die Anzahl der Testsequenzen, Fehler und Parameterwerte und die daraus resultierende Anzahl der Simulationen. Weiterhin ist der Anteil der Simulationen, bei denen ein Signalwechsel am Ausgang stattfindet und somit eine Verzögerungszeit bestimmt werden kann und der Anteil der Simulationen, bei denen kein Signalwechsel stattfindet, dargestellt.

Tabelle 2: Übersicht über die Anzahl der Elemente in den Listen, die nötigen Simulationen und die Ergebnisse

Gatter	#Testsequenzen	#Fehler	#Parameter-werte	#Simulationen	Signalwechsel	kein Signalwechsel
INV	2	150	10 000	3 000 000	55,2%	44,8%
NAND2	6	240	10 000	14 400 000	67,7%	32,3%
NOR2	6	240	10 000	14 400 000	62,2%	37,8%
XOR2	8	660	10 000	52 800 000	72,2%	27,8%

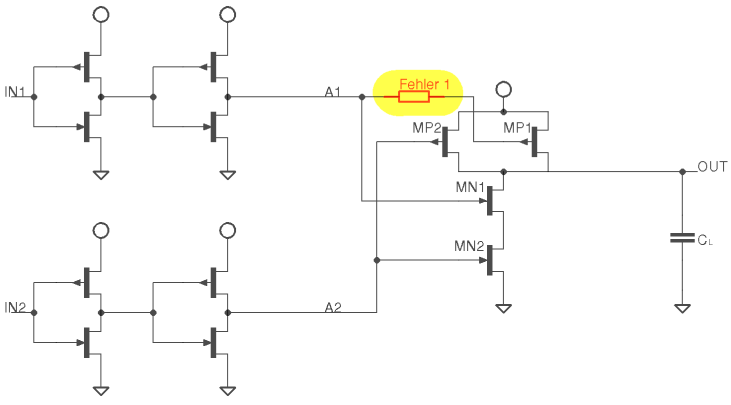


Abbildung 3: Schaltbild des eingebetteten NAND2-Gatters mit injiziertem Fehler 1

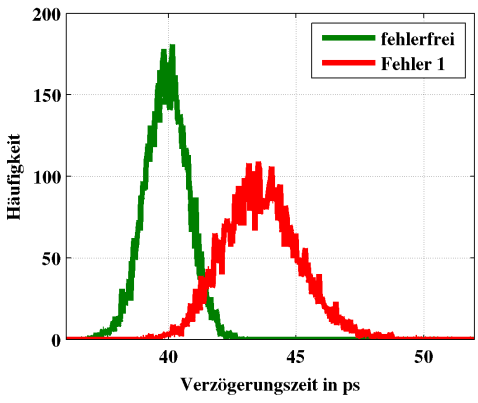


Abbildung 4: Histogramm für die fehlerfreie und Fehler 1 Schaltung über alle Parameterwerte

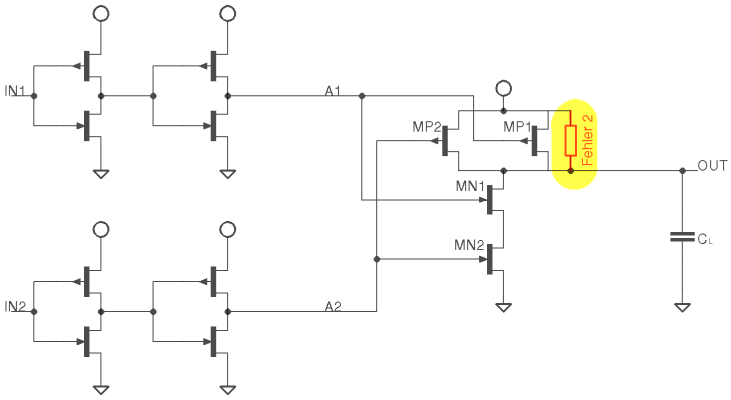


Abbildung 5: Schaltbild des eingebetteten NAND2-Gatters mit injiziertem Fehler 2

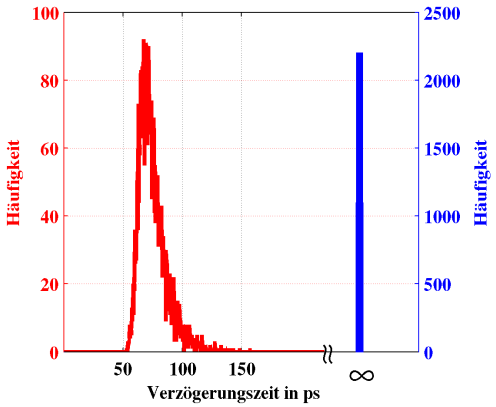


Abbildung 6: Histogramm für die Schaltung mit Fehler 2 über alle Parameterwerte

Die Simulationsergebnisse werden für weiterführende Analysen und Simulationen auf der Logikebene zusammengefasst. In einem ersten Schritt werden die Ergebnisse für jeweils eine Testsequenz und einen Fehler über alle Parameterwerte in Form von Histogrammen aufbereitet. Die erhaltenen Histogramme können zum Beispiel für eine SSTA unter Fehlereinwirkung verwendet werden.

Das Schaltbild des eingebetteten NAND2-Gatters mit einem injiziertem Fehler 1 ist in Abbildung 3 dargestellt. Er stellt eine Unterbrechung des Gateanschlusses des p-Kanal-Transistors MP1 dar. Modelliert wird diese Unterbrechung durch Ersetzen des parasitären Leitungswiderstandes von ca. 50 Ω, der die entsprechende Leitung verkörpert, durch einen Widerstand mit einem Wert von 500 kΩ. In Abbildung 4 sind die Verzögerungszeiten der fehlerfreien und der Schaltung mit Fehler 1 für das Eingangssignal, bei dem beide Eingänge von 0 auf 1 wechseln, über alle Parameterwerte in Form von Histogrammen dargestellt. Es ist zu erkennen, dass sich beide Histogramme überlappen und dass die Streuung für den Fehler 1 größer ist als für den fehlerfreien Fall.

In Abbildung 5 ist ein Fehler 2 in das Schaltbild eingezeichnet. Der Fehler 2 stellt einen Drain-Source-Kurzschluss des p-Kanal-Transistors MP1 dar, der durch Einfügen eines Widerstandes von 7 500 Ω zwischen dem Drain- und dem Source-Anschluss des Transistors modelliert wird. Das Ergebnis der Simulation ist für das Eingangssignal, bei dem beide Eingänge von 0 auf 1 wechseln, in Abbildung 6 dargestellt. Für die meisten der Parameterwerte findet ein Signalwechsel am Ausgang statt. Diese Resultate werden entsprechend dem Wert der Verzögerungszeit in die zutreffende Klasse eingeordnet. Die ca. 2 000 Ergebnisse, bei denen kein Signalwechsel stattfindet, werden in einer speziellen Klasse ∞ eingeordnet.

Nach Zusammenfassen über die Parameterwerte ergeben sich für das NAND2-Gatter 1 440 Einzelergebnisse in Form von Histogrammen. Davon enthalten 64,0% der Histogramme kein Element in der Klasse ∞, für 26,7% der Ergebnisse werden alle Elemente in die Klasse ∞ eingeordnet und für 9,3% sind sowohl Elemente in der Klasse ∞ als auch in den anderen Klassen enthalten. Eine Übersicht über die Histogramme der anderen Gatter ist in Tabelle 3 gegeben.

Die Ergebnisse können für eine SSTA verwendet werden. Für jedes Gatter einer Schaltung wird das entsprechende Histogramm aus der Bibliothek verwendet. Das Timing kann somit nicht nur

Tabelle 3: Ergebnisse als Histogramme aufbereitet

Gatter	#Simulationen	#Histogramme	Kein Element in ∞	Alle Elemente in ∞	Elemente in und außerhalb ∞
INV	3 000 000	300	51,0%	37,7%	11,3%
NAND2	14 400 000	1 440	64,0%	26,7%	9,3%
NOR2	14 400 000	1 440	58,8%	32,7%	8,5%
XOR2	52 800 000	5 280	68,5%	24,3%	7,2%

bezüglich Parameterschwankungen verifiziert werden, sondern auch bezüglich der Kombination von Fehlern und Parametervariationen.

5 Zusammenfassung und Ausblick

Im vorliegenden Beitrag wurde die elektrische Fehlersimulation unter Parameterschwankungen und der dazu verwendete Fehlersimulator aFSIM vorgestellt. Die Simulationen wurden für einige Grundgatter aus der Nangate 45 nm OCL durchgeführt. Es wurden die Verzögerungszeiten der Grundgatter unter Fehlereinwirkung und Parametervariationen bestimmt. Die Ergebnisse wurden in Form von Histogrammen über alle Parameterwerte aufbereitet. Die erzielten Resultate können für Analysen und Simulationen auf der Logikebene verwendet werden. Dazu gehört zum Beispiel die SSTA, wobei neben den Parameterschwankungen auch Fehlerwirkungen berücksichtigt werden können.

In weiteren Arbeiten soll die Bibliothek weiter ausgebaut werden. Dazu werden Simulationen für weitere Grundgatter sowie für andere Anstiegszeiten und Lasten durchgeführt. Weiterhin ist vorgesehen, die erzielten Ergebnisse weiter zusammenzufassen. Dabei sollen die Histogramme der einzelnen Fehler durch die Verwendung von Auftrittswahrscheinlichkeiten von Fehlern verbunden werden.

Literatur

[1] ITRS, "International Technology Roadmap for Semiconductors," 2009 Edition, http://www.itrs.net.

[2] F. Hapke, R. Krenz-Baath, A. Glowatz, J. Schloeffel, H. Hashempour, S. Eichenberger, C. Hora, D. Adolfson, "Defect-Oriented Cell-Aware ATPG and Fault Simulation for Industrial Cell Libraries and Designs," Proc. IEEE International Test Conference 2009, Austin, Texas, USA, November, 1-6, 2009.

[3] A. Srivastava; D. Sylvester; D. Blaauw, "Statistical Analysis and Optimization for VLSI - Timing and Power" Berlin : Springer, 2005.

[4] A. Bounceur, S. Mir, E. Simeu, and L. Rolíndez, "Estimation of Test Metrics for the Optimisation of Analogue Circuit Testing," Journal of Electronic Testing: Theory and Applications (JETTA), Vol. 23, No. 6, December 2007, pp. 471-484.

[5] S. J. Spinks, C. D. Chalk, I. M. Bell, and M. Zwolinski, "Generation and Verification of Tests for Analog Circuits Subject to Process Parameter Deviations," Journal of Electronic Testing: Theory and Applications (JETTA), Vol. 20, No. 1, February 2004, pp. 11-23.

[6] A. Khouas and A. Derieux, "Fault Simulation for Analog Circuits Under Parameter Variations," Journal of Electronic Testing: Theory and Applications (JETTA)). Vol. 16, No. 3, Juni 2000, pp. 269-278.

[7] K. Saab; N. Ben-Hamida; and B. Kaminska, "Parametric Fault Simulation and Test Vector Generation," Proc. of the Conference on Design, Automation and Test in Europe, DATE 2000, Paris, France, March 27-30, 2000, pp. 650–657.

[8] A. V. Gomes, R. Voorakaranam, and A. Chatterjee, "Modular Fault Simulation of Mixed Signal Circuits with Fault Ranking by Severity," Proc. 13th International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT '98), Austin, TX, USA, November 2-4, 1998, pp. 341-348.

[9] J. Peralta, G. Peretti, E. Romero, C. Marqués, "A New Performance Characterization of Transient Analysis Method," International Journal of Electronics, Communications and Computer Engineering (IJECCE), Vol. 1, No. 1, 2009. pp. 12-19.

[10] F. Liu, S. Ozev, "Statistical Test Development for Analog Circuits Under High Process Variations," IEEE Trans. on CAD of Integrated Circuits and Systems (TCAD), Vol. 26, No. 8, August 2007, pp. 1465-1477.

[11] G. Devarayanadurg, P. Goteti, and M. Soma, "Hierarchy based Statistical Fault Simulation of Mixed-Signal ICs," Proc. IEEE International Test Conference 1996, Test and Design Validity, Washington, DC, USA, October 20-25, 1996, pp. 521-527.

[12] R. C. Aitken, "Defect or Variation? Characterizing Standard Cell Behavior at 90 nm and Below," IEEE Tr. on Semiconductor Manufacturing, Vol. 21, No. 1, February 2008, pp. 46-54.

[13] U. Schlichtmann, M. Schmidt, H. Kinzelbach, M. Pronath, V. Glöckel, M. Dietrich, U. Eichler, J. Haase, "Digital Design at a Crossroads – How to Make Statistical Design Industrially Relevant" Proc. Design, Automation and Test in Europe, DATE 2009, Nice, France, April 20-24, 2009, pp. 1542-1547.

[14] Michael Orshansky, Sani Nassif, Duane Boning: "Design for Manufacturability And Statistical Design: A Constructive Approach", Springer, 2008.

[15] B. Straube, B. Müller, W. Vermeiren, C. Hoffmann, S. Sattler, "Analogue fault simulation by aFSIM," Design, Automation and Test in Europe Conference and Exhibition, DATE 2000 – User Forum, Paris, March 27-30, 2000, pp. 205-210.

[16] Nangate 45nm Open Cell Library, http://www.nangate.com.

[17] A. Hirata, H. Onodera, and K. Tamaru, "Estimation of Propagation Delay Considering Short-Circuit Current for Static CMOS Gates," IEEE Trans. on Circuits Syst. I, Fundamental Theory and Applications. Vol. 45, No. 11, Nov. 1998, pp. 1194-1198.

Analog IP Porting: A Puzzle of Topology Conversion, Optimization and Verification

Udo Sobe, Enno Böhme, Dietmar Mörtl, Achim Graupner
{Udo.Sobe, Enno.Boehme, Dietmar.Moertl, Achim.Graupner}@zmdi.com
ZMD AG, Grenzstraße 28, 01109 Dresden

Michael Pronath
Michael.Pronath@muneda.com
MunEDA GmbH, Stefan-George-Ring 29, 81929 München

Abstract

The article presents an approach to convert a circuit design from technology A to technology B. To do this the design data is converted from PDK A to PDK B, because the design environment uses the process design kit (PDK) of the technology. The methodology comprises of topology conversion to transfer design data and an optimization step for sizing. The simulation-based optimization and verification need qualified device models. There are differences between PDKs in the modelling of process deviation. A method is proposed to check availability of corner models and statistical models for all devices.

1 Introduction

In the domain of analog circuit design the circuit topology is usually regarded as the intellectual property (IP). Proven design topologies and concepts are reused in the development of new circuits. This is done by adapting all required circuit topologies to the new technology. The adaption includes mapping to available devices, the sizing of the devices and any necessary modifications of the topology [1]. Iterations of manual and interactive tasks dominate the process to transfer the design data between technologies.

Most of the design data of the circuit topology are bound to the topology's PDK inside the design environment. Hence, the transfer of design data from technology A to technology B is associated with design data transfer from PDK A to another PDK B. The design environment we use does not provide an efficient function for this; the search and replace function lacks the required functionality.

We know that the reuse of a given circuit topology is limited by many constraints. Hence, a feasibility estimation should be the first step. This can be done using simplified models of the topology e.g. using pencil and paper.

After this introduction, the porting strategy is explained. Details to the conversion and optimization step are given next. A method to check the device model support is presented, which is the basis for representative verification of the circuit performance. Finally an example using the porting methodology is given.

2 Porting Strategy

Our porting strategy first transfers the circuit topology, then followed by sizing, e.g. using optimization, and verification (see Fig. 1a). The conversion step is supported by our in-house tool called Technology2Technology converter (T2T), which provides a graphical user

interface (GUI) for interactive tasks as well as a scripting interface to process an entire library.

T2T supports the schematic transfer from technology A (source) to technology B (destination) with respect to the PDK. The tool converts library names, cell names and device properties like w, l, or m. A conversion from device property to user property is supported. Design variables and pPar parameters can also be transferred. T2T does not support mapping of different symbol geometries between PDKs. The transfer of performance parameters (e.g. gain, unity gain bandwidth) is not objective of the T2T tool (see section 3).

A conversion of a circuit topology to a different technology can reduce the circuit performance or even cause the circuit not to work at all, because the circuit performance is not considered at this stage. Therefore the functionality has to be analyzed and has to be restored by the following optimization step. The flow is also suitable to transfer design data between different PDK versions of one technology. In this case the following optimization step may be skipped.

Generally, the circuit needs resizing to restore its performance or to meet a new specification. The traditional approach is the manual sizing by the designer. MunEDA's optimization suite WiCkeD [2] is convenient for this task, because it supports an interactive optimization approach (see section 4).

A final verification step is used to check performance over PVT conditions (*P*rocess, operating conditions: *V*oltage and *T*emperature) with corner and Monte Carlo analyses before the layout is created. We use our in-house verification environment called zmdAnalyser [3] for this step.

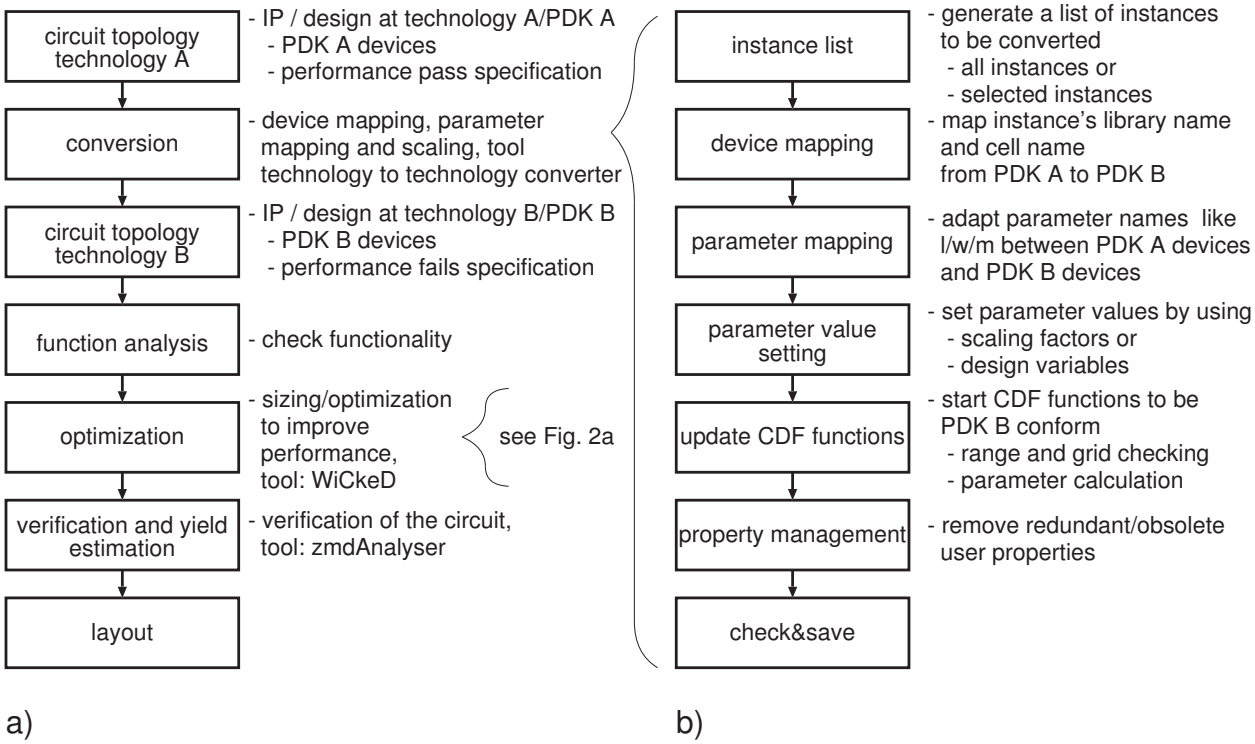


Figure 1: Our porting strategy is focused on daily design tasks. a) Main transfer steps. b) Details of transfer design data to another PDK.

3 Topology Conversion

The Fig. 1b shows details of the conversion steps. An instance list is built, which can contain either those selected in the Virtuoso Schematic Editor window or all instances of the schematic. The devices are converted to the new PDK B according to the device mapping table configured in the GUI. This step is simply a search and replace of two parameters: library name and cell name. Parameter name mapping is necessary for conversion to allow differing parameter names, e.g. *w* in PDK A vs. *width* in PDK B.

An important detail of the Cadence design environment [4] for the design data transfer is the default setup for parameter storage: Property values which are set to the PDK's defaults are not stored as instance parameters. The GUI does not determine the defaults from the PDK but provides fields for default parameter values, which are used if the property is not defined.

The parameter mapping and scaling is validated by updating the CDF (component description format) functions. These functions are provided by today's PDKs and implement many of PDK-related tasks in the background. The procedure used to trigger CDF callback functions is technology and PDK independent.

An additional property management step is necessary to control user properties. In our experience schematics tend to accumulate obsolete data over their history, which should be deleted at this point.

The conversion is finished by the final check and save step. In this step, standard checks grouped into logical, physical, name, inherited connection and AMS checks are performed.

4 Sizing by Optimization

Our approach to convert the circuit topology prepares the design data for PDK B in a GUI-based design environment, which dominates our reuse practice. Unlike other solutions as [5], this first step only transfers design data with an optional scaling of parameters without any consideration of the circuit's performance. The latter is achieved by optimization during the sizing process using an interactive optimization approach.

Generally WiCkeD's optimization strategy bases on three main steps (cf. Fig. 2a). First the operating points of basic structures, e.g. differential pairs, are optimized using a constraint matrix concept and the feasibility optimization (FO). Then the nominal optimization (NO) is carried out to improve circuits performance regarding operating range, such as supply voltage and temperature. Finally design centering, which is called yield optimization (YO), is carried out to maximize the yield.

Analog circuits are individual and may need a different optimization strategy. For such cases the tool suite WiCkeD allows a flexible flow of the tool features. User experiences and applications for various types of circuits are available in the user group section of reference [2].

5 Standard Verification using standard PDK

Today it is state of the art to use corner and Monte Carlo analyses to verify circuit performances with respect to process deviations at extreme conditions and over both global and local statistics respectively. These simulation based analyses need appropriate device models which are part of the PDKs.

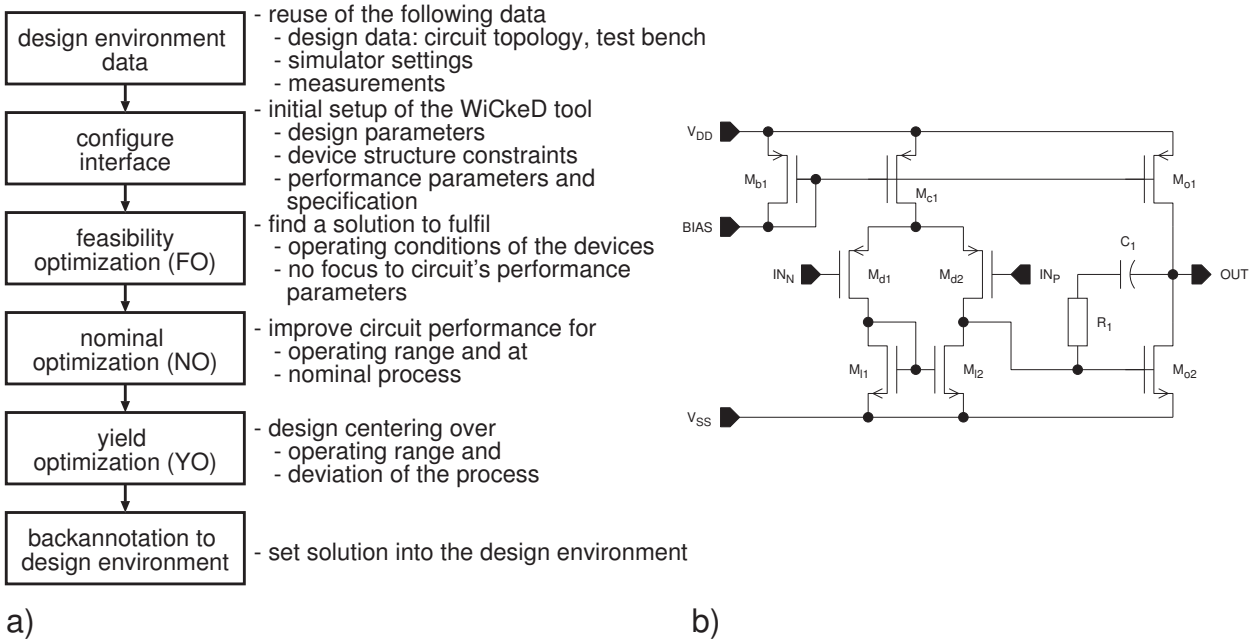


Figure 2: Sizing by optimization. a) WiCkeD's main optimizations steps. b) A simple example circuit.

The PDKs differ from process to process and sometimes even between releases of one process regarding the model support for the analyses. There is no guarantee to get models for extreme conditions, global and local statistics for every device. So it may not be possible to capture process deviations for all device types during analysis. This situation causes a verification gap which is not obvious for the designer, because no error or warning message is given.

The design environment we use does not report such a scenario to avoid such errors. Therefore we added such a check to our in-house verification environment zmdAnalyser [3]. This feature uses a matrix which covers every primitive device vs. process deviations. Fig. 3a shows a proposal to generate the matrix.

Key of the proposal is the test bench generation. Every primitive device is instantiated including an appropriate device test bench (cf. Fig. 3b). Standard corner and Monte Carlo analyses which are used for general circuit verification are carried out for the device test bench. The advantage of this simulation-based approach is the use of the same model setup and its associated tools for matrix generation and the verification of the circuits. The proposed method results in a matrix which gives the designer an overview or can be used for tool supported checking before verification.

6 Example for the T2T Conversion

The T2T converter has already been used, e.g. to transfer an analog front end to a new technology. We use the simple textbook operational transconductance amplifier (OTA) shown in Fig. 2b to illustrate a conversion from an 0.6 µm technology (power supply voltage $V_{\mathrm{DD}} = 3.3$ V to 5 V) to an 0.18 µm technology ($V_{\mathrm{DDnom}} = 1.8$ V). In this example the focus will be to save area, reduce the supply voltage and increase the bandwidth while keeping the remaining specification parameters except the gain (60 dB instead of 70 dB) the same.

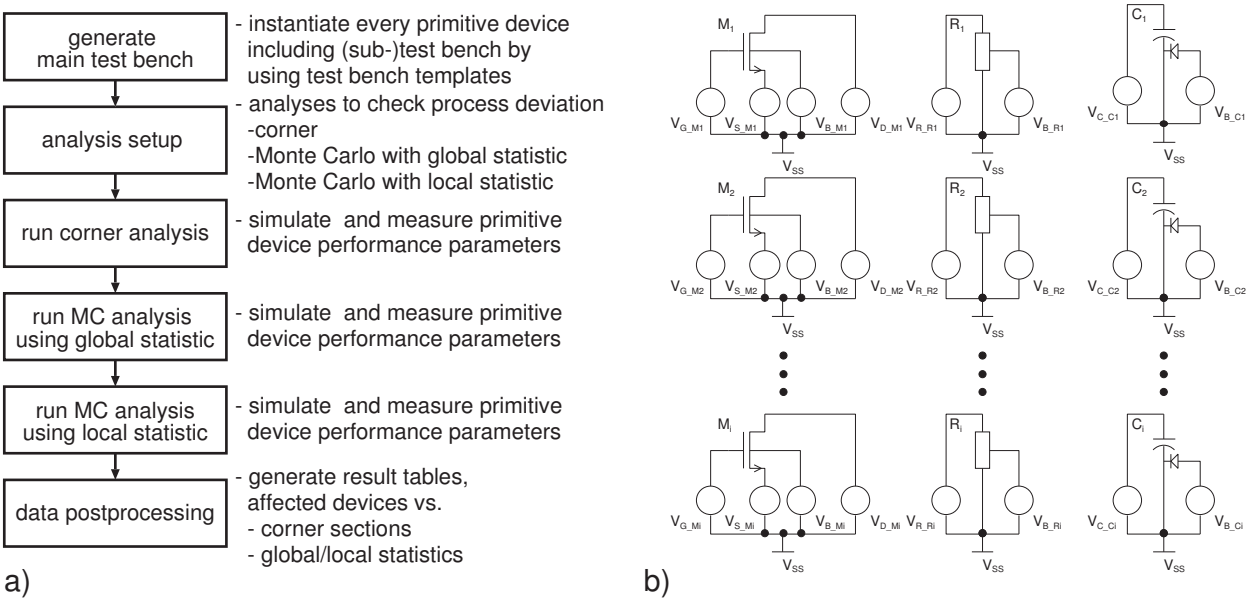


Figure 3: Check of the model support. a) Flow for simulation based model support checking. b) Example of the main test bench which consist of sub-test benches.

As already mentioned, the T2T converter allows to limit conversion to selected devices. To illustrate this feature, the differential pair was converted to medium-Vth transistors while all other transistor were converted to standard devices. This was achieved by first converting the differential pair by selection using a setup converting to medium-Vth devices and then all other devices of the source PDK using the standard setup. All setups can be saved and reloaded to simplify reuse.

For the initial sizing we scaled all transistor lengths L by a factor of 0.25 and set width W to the length using the property setup of the T2T converter; this is done without any regard for the circuit performance. Therefore, the resulting performance is very poor.

WiCkeD's optimization flow (cf. Fig. 2a) was used to restore circuit performances. Optimization details for this example are given in [6]. The results after all these steps are also shown in Table 1.

	Design Step	**Gain**	**BW**	**PM**	**I/I_{max}**	**SR**	**FC**	**Area**	**Yield**
	initial design (0.6 µm)	76.2	1.27	66.5°	0.978	1.632	0	100%	100%
1	after T2T (0.18 µm)	20.6	0.24	26.1°	0.784	1.304	4		
2	after FO	54.7	1.45	27.1°	0.683	0.981	0		
3	after typ. NO	66.0	3.19	52.2°	0.993	1.289	0		
4	after *wc* NO	69.8	3.11	62.1°	0.902	1.273	0		89%
5	after YO	68.4	3.25	67.3°	0.938	1.338	0	3.2%	100%

Table 1: Important performance parameters at different stages of technology conversion. For each performance, the value at its respective worst-case operating condition is given.
Note: Gain = Open-loop Gain in dB; BW = Unity Gain Bandwidth, normalized; PM = Phase Margin; I/I_{max} = Current, normalized; SR = Slew Rate, normalized; FC = number of violated constraints; Area = relative Transistor Area (without area for R and C); Yield = Yield estimated by Monte Carlo analysis. Normalizations are all with respect to the specifications of the initial design.

Of course, the topology of our example low-power OTA is well-known and sizing rules can be found in the literature. Overall, the entire conversion, sizing for worst-case operating conditions and design centering of this OTA including verification required only very little effort from the designer (approximately a quarter of an hour excluding run time).

7 Conclusions

Circuit topology is the real analog IP, which is reused at different technology generations. The manual conversion of a complex circuit with up to 100 devices to another technology PDK is a time consuming task. Sizing the circuit is also expensive.

The presented method reflects common analog design practice and is divided into design data conversion and sizing by optimization. Tools to support this flow are discussed.

The simulation-based verification and optimization using a standard PDK need qualified devices models. If the device models do not cover process deviation for all device types, a verification gap will occur which is not reported by commercial verification tools. A proposal to identify such a situation is given. Finally PDK's must be prepared to the design house's development processes to avoid such problems.

Acknowledgment— The work described in this report is supported by the German Ministry of Education and Research (*Bundesministerium für Bildung und Forschung*) BMBF, grant ID 01 M 3086 F. The authors are responsible for the contents of this publication.

References

[1] Peter Schwarz. Reuse of Mixed Analog-Digital Circuits. *it + ti*, pages 91–98, 2002.

[2] `www.muneda.com`.

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

[4] `www.cadence.com`.

[5] Sherif Hammouda et al. Chameleon ART: A non-optimization based analog design migration framework. In *DAC '06: Proceedings of the 43th annual conference on Design automation*, pages 885–888, New York, NY, USA, 2006. ACM.

[6] Udo Sobe, Enno Böhme, Achim Graupner, Andreas Ripp, and Michael Pronath. Analog IP porting by topology conversion and optimization. In *IP ESC 2009*, 2009.

Messverfahren zur Fehlerquellenanalyse im Bereich elektrisch leitfähiger Textilien

Ulf Wetzker,
ulf.wetzker@eas.iis.fraunhofer.de,
Fraunhofer-Institut für Integrierte Schaltungen,
Institutsteil Entwurfsautomatisierung Dresden

Kurzfassung

Die Entwicklung intelligenter Textilien basiert auf einem domänenübergreifenden Entwurfsprozess mit hohen Anforderungen an die Robustheit. Eine der Grundvoraussetzungen für den Entwurf eines anpassungsfähigen Systems stellt die Modellierung der verwendeten Komponenten inklusive der zugehörigen Fehlermodelle, z.B. Umwelteinflüsse wie Feuchtigkeit und Schweiß, dar. Dieser Beitrag präsentiert ein auf textile Leitungsstrukturen angepasstes Messverfahren zur Bestimmung elektrischer Eigenschaften in Abhängigkeit der gravimetrischen Feuchtigkeit. Die Ergebnisse dieser automatisierten Langzeitmessungen ermöglichen die Parametrisierung eines Fehlermodells. Mit Hilfe einer Feldsimulation konnte ein Messaufbau mit minimalen elektromagnetischen Wechselwirkungen zwischen der Messeinrichtung und dem Messobjekt entworfen werden.

Schlüsselwörter: smart textiles, Leitungsparameter, Feuchtigkeit, Fehlermodell

Stand der Technik

Die nächsten Generationen neuartiger textiler Produkte werden in ihrer Funktionalität maßgeblich durch die Integration nanoelektronischer und mikromechanischer Komponenten bestimmt werden. Kompakte Sensor-Aktor-Systeme, die unter Berücksichtigung der textilen Eigenschaften in Kleidung integriert werden [1], haben sich zu einem wichtigen Forschungsbereich der Medizintechnik entwickelt [2, 3]. Entsprechend dem Kerngedanken des 'ubiquitous computing' [4] zeichnen sich diese intelligenten Textilen durch alternative Benutzerschnittstellen aus und ermöglichen auf Grund ihrer Körpernähe eine unaufdringliche Überwachung unterschiedlicher Vitalparameter [5, 6].

Die in einer SFIT-Anwendung ('smart fabrics and interactive textiles') enthaltenen Komponenten werden entsprechend physiologischer Entwurfsrichtlinien [7] in Form eines Verteilten Systems miteinander vernetzt. Robustheit, Adaptivität und ein niedriger Energiebedarf sind typische Anforderungen an solch ein Body-Area-Network (BAN). Textile Leiterbahnen werden zunehmend für die lokale Vernetzung verwendet, da drahtlose Verfahren auf Grund der hohen Komplexität der Sende- und Empfangselektronik [8] sowie der spezifischen Absorptionsrate (SAR) des menschlichen Körpers [9] eine schlechtere Energieeffizenz aufweisen. Ein weiterer Vorteil drahtgebundener BANs ergibt sich aus der Möglichkeit der Energieübertragung.

Die robuste Signalübertragung unter Einwirkung textiltypischer Umwelteinflüsse erfordert die Charakterisierung der elektrischen Eigenschaften der verwendeten leitfähigen Strukturen [10, 11] sowie eine entsprechende Analyse möglicher Fehlerquellen. Grundlegende Ergebnisse wurden hierbei bereits für die Materialalterung durch Waschen [12] und die Auswirkungen elektrostatischer Entladungen [13] oder der relativen Luftfeuchtigkeit [14] erzielt.

Einflussfaktor Feuchtigkeit

Entsprechend der allgemeinen Leitungstheorie ist das Signalausbreitungsverhalten entlang eines Leiterpaares maßgeblich durch dessen Induktivitäts-, Kapazitäts-, Widerstands- und Ableitungsbelag gekennzeichnet. Die messtechnische Erfassung dieser Leitungsparameter wird mit Hilfe von Präzisions-LCR-Metern, Netzwerkanalysatoren oder über eine Signalanalyse unter Verwendung der Zeitbereichsreflektometrie durchgeführt.

Auf Grund ihres Anwendungsbereichs und ausgeprägter hygroskopischer Eigenschaften sind Textilien häufig der Einwirkung von Feuchtigkeit ausgesetzt. Die auf diese Weise entstehende Verbindung aus einer Flüssigkeit und dem Gewebematerial beeinflusst die Leitungsparameter. Neben direkten Auswirkungen, wie der gravimetrischen Feuchtigkeit des Gewebes, existiert eine Reihe indirekter Einflussfaktoren, die für eine repräsentative Analyse berücksichtigt werden müssen. Flüssigkeiten, welche wie Wasser aus polaren Molekülen mit einer hohen Beweglichkeit bestehen, besitzen auf Grund von Orientierungspolarisation eine temperaturabhängige relative Permeabilität. Eine weitere charakteristische Eigenschaft eines Dielektrikums aus polaren Molekülen ist eine ausgeprägte Frequenzabhängigkeit. Freie Ladungsträger, welche als chemische Bestandteile in einer Flüssigkeit enthalten sein können, stellen einen zusätzlichen Einflussfaktor auf die elektrischen Parameter leitfähiger Textilen dar. Die resultierenden Übertragungseigenschaften der Leitungsstruktur ergeben sich aus der Ausprägung der einzelnen Einflussfaktoren.

Versuchsaufbau

Zur Entwicklung eines auf diese Fragestellung zugeschnittenen Messverfahrens sowie zum besseren Verständnis der Auswirkung verschiedener Flüssigkeiten wurde ein initiales Experiment durchgeführt. Der Versuchsaufbau zur Nachbildung eines auf textilen Leitern basierenden BANs verwendete als Datenquelle einen Philips Signalgenerator PM5193. Diesem wurde ein Texas Instruments SN74CB3T3245 Pegelwandler nachgeschaltet, um der Amplitude und den Anstiegszeiten eines Mikroprozessors mit Low-Voltage-TTL-Logikpegel (LVTTL) zu entsprechen. Bei der untersuchten textilen Leitung handelt es sich um eine gewebte Zweileiterstruktur aus Elitex-Garn [15], deren Abmessungen Abbildung 1 zu entnehmen sind.

w (Breite)	s (Abstand)	h (Höhe)	Länge der Leiterbahn
5 mm	1 mm	0.15 mm	350 mm

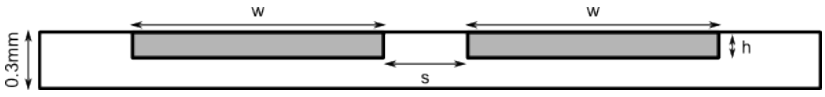


Abbildung 1: Abmessungen der textilen Leitungsstruktur

Für eine Gegenüberstellung der Übertragungseigenschaften unter textiltypischen Umweltbedingungen wurde der Stoff mit Flüssigkeiten unterschiedlicher Salzkonzentration befeuchtet: destilliertes Wasser, Kochsalzlösung einer Salinität von 1% (entspricht menschlichem Schweiß [16]) und Kochsalzlösung einer Salinität von 3% (entspricht Meerwasser). Die resultierende Signalform wurde mit Hilfe eines Tektronix Oszilloskops TDS3054 aufgezeichnet. Alle Messungen wurden bei einer gravimetrischen Feuchtigkeit von 50%, 25%, 10%, 5% und 0% durchgeführt. Die gravimetrische Feuchtigkeit (WG_m) stellt hierbei die Massenrelation zwischen trockenem Basismaterial (m_{Ts}) und der enthaltenen Flüssigkeit (m_w) entsprechend folgender Gleichung dar.

$$WG_m = \frac{m_w}{m_w + m_{Ts}} \cdot 100$$

Ergebnisse und Auswertung

Die Ergebnisse der Messung sind in Form der Signalkenngrößen in Tabelle 1 dargestellt und zeigen eine mit steigender Feuchtigkeitskonzentration abfallende Amplitude. Das Überschwingen des Signals nimmt hierbei zu. Mit zunehmender Salzkonzentration ist eine erhöhte Dämpfung der Amplitude und des Überschwingens zu verzeichnen (siehe Abbildung 2). Eine digitale Auswertung war unter der Einwirkung der Kochsalzlösungen bei 20%, 25% und 50% gravimetrischer Feuchtigkeit nicht mehr möglich. Die geringe Amplitude bei einer Salinität von 3% ließ unter diesen Bedingungen keine repräsentative Auswertung des Überschwingens und der Anstiegszeit zu (siehe Abbildung 2d).

Flüssigkeit	grav. Feuchtigkeit	Amplitude	Anstiegszeit	Überschwingen
destilliertes Wasser	50%	2,76 V	4,0 ns	53%
	25%	2,84 V	4,0 ns	53%
	10%	2,96 V	3,9 ns	52%
	5%	3,04 V	3,8 ns	51%
	0%	3,28 V	4,9 ns	15%
1% NaCl-Lösung	50%	1,04 V	5,8 ns	23%
	25%	1,16 V	6,3 ns	21%
	20%	1,72 V	5,5 ns	19%
	5%	2,32 v	5,1 ns	17%
	0%	3,28 V	4,8 ns	15%
3% NaCl-Lösung	50%	0,39 V	-	-
	25%	0,53 V	-	-
	20%	1,4 V	-	-
	5%	2,8 V	3,6 ns	59%
	0%	3,28 V	4,9 ns	15%

Tabelle 1: Messergebnisse eines 1 kHz LVTTL-Signals

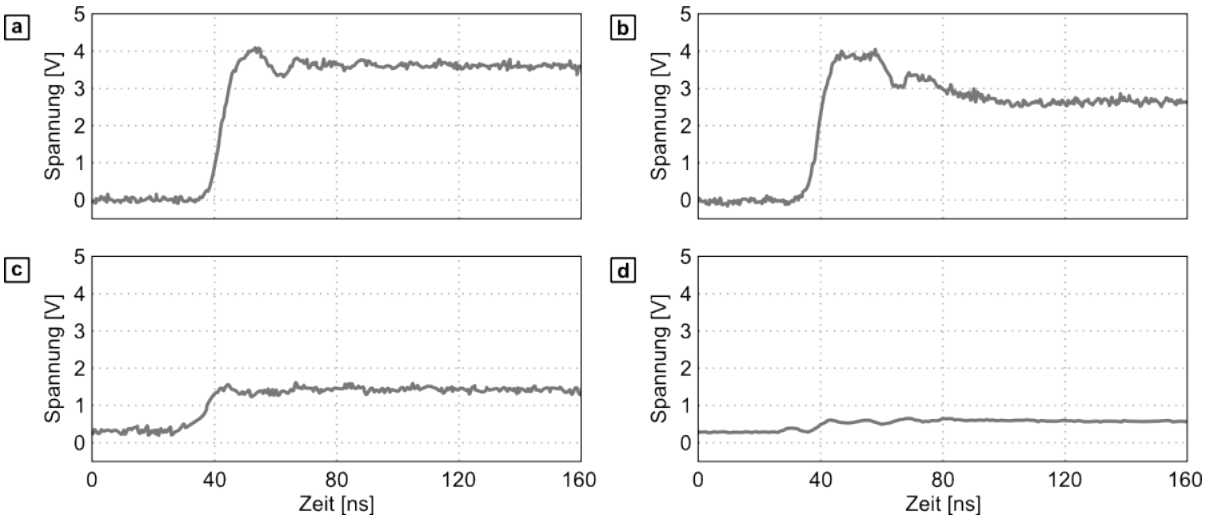


Abbildung 2: Signalflanke eines LVTTL-Impulses unter Einwirkung von Flüssigkeiten
(a) trockene Probe; (b) dest. Wasser, WG_m 50%; (c) 1% NaCl-Lösung, WG_m 50%; (d) 3% NaCl-Lösung, WG_m 50%

Eine Analyse der Messergebnisse ermöglicht Rückschlüsse auf die Entwicklung der Leitungsparameter unter Einwirkung der verschiedenen Flüssigkeiten. Die mit steigendem Salzgehalt stark abfallende Signalamplitude weist auf eine verringerte Isolationsfähigkeit des textilen Trägermaterials hin. Somit hat die Sanilität einer Flüssigkeit eine starke Auswirkung auf den Ableitungsbelag der Leitung. Für die Herstellung der textilen Leiterbahnen werden im Bereich der Daten- und Signalübertragung Fasern mit einem geringen ohmschen Widerstand verwendet [15]. In Flüssigkeiten enthaltene freie Ladungsträger haben daher nur eine minimale Auswirkung auf den Widerstandsbelag. Das durch die Flüssigkeit bedingte stärkere Überschwingen weist auf Änderungen des Wellenwiderstands hin. Unter Berücksichtigung der

polaren Eigenschaften von Wasser lässt dies auf ein Ansteigen des Kapazitäts- und Induktivitätsbelags schließen. Wie den Messwerten aus Tabelle 1 zu entnehmen ist, treten besonders starke Änderungen in der letzten Phase des Trockenprozesses auf. Eine Detailanalyse dieses Bereichs ist mit dem beschriebenen Versuchsaufbau nicht möglich.

Mit Hilfe des vorgestellten Experiments konnten die theoretischen Vorüberlegungen bestätigt und quantifiziert werden. Besonders die Feuchtigkeitskonzentration sowie die Salinität der untersuchten Flüssigkeit haben einen Einfluss auf die Leitungsparameter und können eine vollständige Unterbrechung der Signalübertragung verursachen.

Messverfahren und -aufbau

Für die Entwicklung robuster BANs auf Basis textiler Leitungen ist es somit notwendig, die Übertragungseigenschaften unter Einwirkung von Flüssigkeiten bereits beim Entwurf der Anwendung zu berücksichtigen. Eine frühzeitige Fehlererkennung und optimierte Dimensionierung eines Sensornetzwerks lässt sich mit Hilfe von Netzwerksimulatoren erreichen. Für die Entwicklung umfassender Übertragungs- und Fehlermodelle ist somit ein Messverfahren notwendig, welches eine Korrelation zwischen den elektrischen Parametern, der gravimetrischen Feuchtigkeit und der Salinität der Testflüssigkeit ermöglicht. Zur akkuraten Erfassung der Messwerte nach Änderungen der Feuchtigkeitskonzentration muss der verwendete Messaufbau externe Störgrößen und parasitäre Effekte weitestgehend ausschließen.

Auf Basis der gewonnenen Erkenntnisse des initialen Experiments wurde ein angepasstes Messverfahren zur computergestützten Ermittlung von Modellparametern entworfen und in Form eines Messaufbaus umgesetzt. Für eine Langzeiterfassung der im Textil enthaltenen Flüssigkeitsmenge wurde eine Mettler Toledo Präzisionswaage XS4002SDR verwendet, die um eine Einspannvorrichtung für textile Leiterbahnen erweitert wurde (siehe Abbildung 3). Der Ausschluss eines Temperaturdrifts des Wägeergebnisses erfordert die Konstanthaltung der Umgebungstemperatur. Zur Kompensation geringer Temperaturschwankungen erfolgt eine kontinuierliche Temperaturüberwachung, auf deren Basis eine automatische Justierung der Waage eingeleitet wird. Durch Luftbewegungen hervorgerufene Schwingungen der Wägefläche stellen eine weitere Störgröße dar. Die Luftdurchlässigkeit der Einspannvorrichtung minimiert diese Störquelle. Vibrationen oder andere Schwingungseinwirkungen äußern sich durch kurzzeitige Gewichtsänderungen und heben sich von den Langzeitänderungen des Trocknungsprozesses eindeutig ab. Die dadurch hervorgerufenen Abweichungen werden im Messprotokoll als undefinierte Messwerte markiert. Zur Verhinderung einer induktiven und kapazitiven Kopplung zwischen Messobjekt und Messaufbau wurden die geometrischen Abmessungen der Einspannvorrichtung auf Grundlage einer Feldsimulation mit dem Finite-Elemente-Simulator Ansys bestimmt. Hierfür wurde das in Abbildung 1 beschriebene Leitungspaar mit einem LVTTL-Pegel angeregt und eine Berechnung des sich in der Umgebung ausbreitenden Potentials durchgeführt. Das in Abbildung 4 dargestellte Ergebnis der Simulation zeigt ein im Abstand von 200 mm weitestgehend konstantes Potential, welches vom Messaufbau nur geringfügig beeinflusst wird.

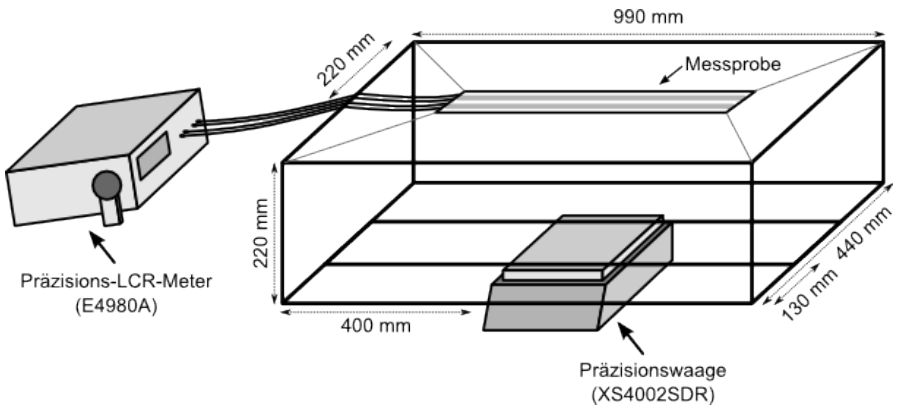


Abbildung 3: Schematische Darstellung des Messaufbaus

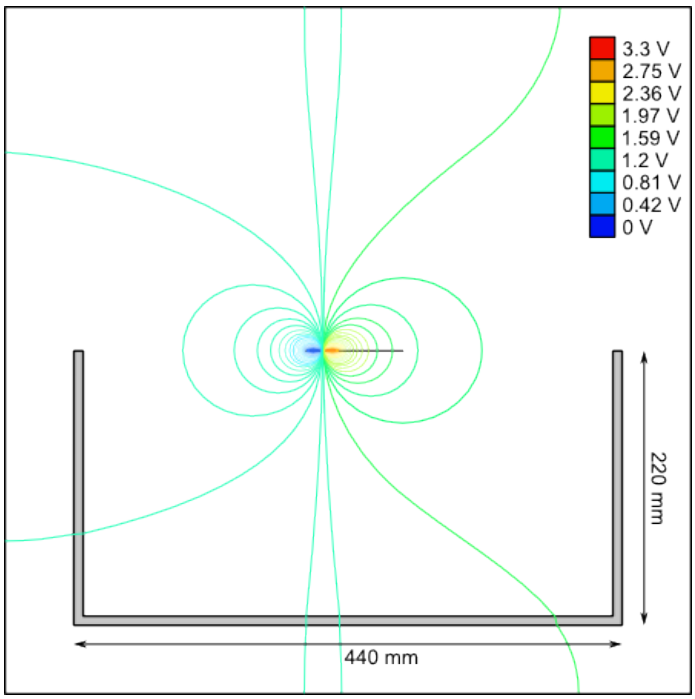


Abbildung 4: Elektrisches Potential einer textilen Leitungsstruktur im Einflussbereich der Einspannvorrichtung

Zur Bestimmung der Leitungsparameter wurde ein Agilent Präzisions-LCR-Meter E4980A eingesetzt, welches mit dem Messobjekt verbunden ist. Die Messkabel wirken hierbei als Federelemente auf die Wägeeinrichtung. Dünne, flexible Koaxialkabel des Typs RG174 minimieren diesen Einfluss auf den vorgestellten Messaufbau. Eine Steuerungssoftware ermöglicht die zeitgleiche Erfassung der Messwerte des Präzisions-LCR-Meters und der Präzisionswaage. Die Periode der Messwertaufnahme ist mit einer Auflösung von 100 ms frei konfigurierbar. Dies ermöglicht eine hochauflösende Beobachtung der zu erfassenden Parameter. Das hieraus resultierende Messverfahren ermöglicht eine detaillierte Gegenüberstellung der Leitungsparameter in Abhängigkeit der gravimetrischen Feuchtigkeit (siehe Abbildung 5). Der entwickelte Messaufbau bietet darüber hinaus die Möglichkeit, unterschiedliche Messgeräte zur Erfassung der Übertragungseigenschaften einzusetzen.

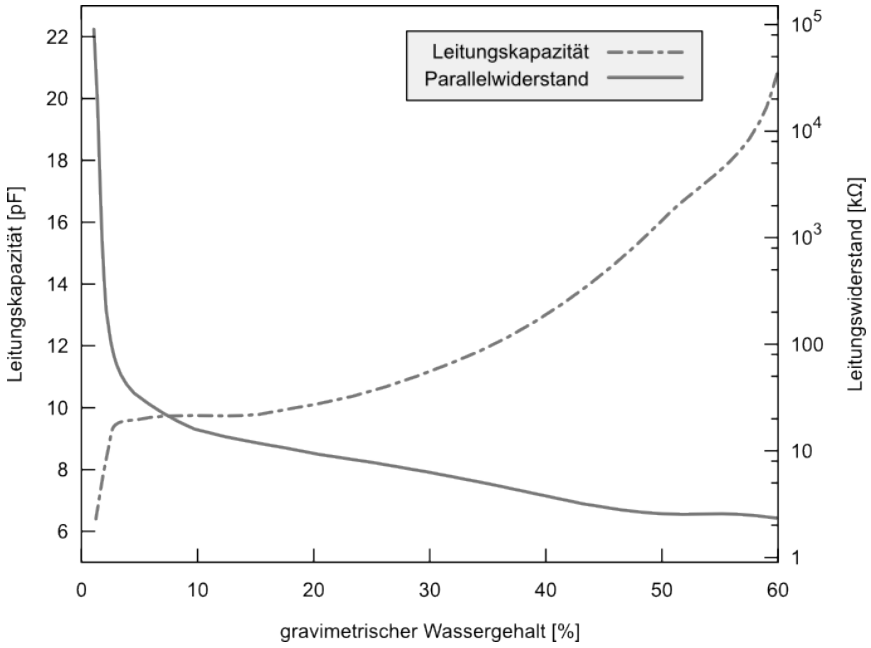


Abbildung 5: Beispielmessung der Leitungskapazität und des Parallelwiderstands eines textilen Leitungspaares bei 100 kHz

Zusammenfassung und Ausblick

Im Rahmen der Bestimmung der Auswirkung von Flüssigkeiten auf die Übertragungseigenschaften textiler Leitungsstrukturen konnte gezeigt werden, dass die gravimetrische Feuchtigkeit und Salinität von Flüssigkeiten starke Signalverformungen bis hin zum Übertragungsausfall hervorrufen. Zur Ableitung von Fehlermodellen bedarf es umfangreicherer Analysen mit Hilfe eines erweiterten Messverfahrens. Auf Basis detaillierter Vorbetrachtungen wurde ein flexibel einsetzbarer Messaufbau vorgestellt. Zur optimalen Erfassung der Parameter sind weitere funktionelle Verbesserungen geplant.

Das aktuell verwendete Präzisions-LCR-Meter besitzt eine obere Grenzfrequenz von 2 MHz und ermöglicht eine auf konzentrierte Bauelemente zugeschnittene Parametrisierung von Spice-Modellen. Alternativ könnte ein Netzwerkanalysator mit einem Messbereich von 9 kHz bis 6 GHz verwendet werden. Dieser ermöglicht die Bestimmung der S-Parameter der textilen Strukturen. Die somit zur Verfügung stehenden Leitungsmodelle können für verschiedene Anwendungen genutzt werden.

Zur Erstellung einer Modellbibliothek wird die beschriebene Messprozedur zukünftig auf unterschiedliche textile Leitungsstrukturen angewendet. Dabei spielen neben den geometrischen Abmessungen auch verschiedene Basismaterialien, wie leitfähige Garne oder Kupfer- bzw. Edelstahldrähte, eine wichtige Rolle. Die Ergebnisse dieser Untersuchungen werden zur Ableitung praktischer Einsatzempfehlungen und Applikationen verwendet.

Literatur

[1] Bossuyt, F., Vervust, T., Axisa, F. and Vanfleteren, J.: A new low cost, elastic and conformable electronics technology for soft and stretchable electronic devices by use of a stretchable substrate, EMPC 2009 (2009).

[2] Linz, T., Kallmayer, C., Aschenbrenner, R. and Reichl, H.: Embroidering electrical interconnects with conductive yarn for the integration of flexible electronic modules into fabric, ISWC2005 (2005).

[3] Carpi, F. and De Rossi, D.: Electroactive polymer-based devices for e-textiles in biomedicine, TITB (2005).

[4] Weiser, M.: The computer for the 21st century, ACM SIGMOBILE MCCR (1999).

[5] Roggen, D., Förster, K., Calatroni, A., Bulling, A. and Tröster, G.: On the issue of variability in labels and sensor configurations in activity recognition systems, Pervasive 2010 (2010).

[6] Patel, S., Lorincz, K., Hughes, R., Huggins, N., Growdon, J., St, D., Akay, M., Dy, J., Welsh, M. and Bonato, P.: Monitoring Motor Fluctuations in Patients With Parkinson's Disease Using Wearable Sensors, TITB (2009).

[7] Gemperle, F., Kasabach, C., Stivoric, J., Bauer, M. and Martin, R.: Design for wearability, ISWC1998 (1998).

[8] Wang, A.Y., Cho, S., Sodini, C.G. and Chandrakasan, A.P.: Energy efficient modulation and MAC for asymmetric RF microsensor systems, ISLPED (2001).

[9] Bilstrup, K.: A Preliminary Study of Wireless Body Area Networks, Halmstad University (2008).

[10] Cottet, D., Grzyb, J., Kirstein, T. and Troster, G.: Electrical characterization of textile transmission lines, IEEE Trans Adv Packag (2003).

[11] Locher, I. and Troster, G.: Screen-printed Textile Transmission Lines, Textile Research Journal (2007).

[12] Chedid, M., Belov, I. and Leisner, P.: Experimental analysis and modelling of textile transmission line for wearable applications, IJCST (2007).

[13] Chedid, M., Belov, I., and Leisner, P.: Modelling and characterization of electrostatic current noise induced mechanically in wired wearable applications, J Electrostatics (2010).

[14] Hertleer, C., Van Laere, A., Rogier, H. and Van Langenhove, L.: Influence of relative humidity on textile antenna performance, Textile Research Journal (2010).

[15] Elitex Datenblatt, Stand 26.05.2008.

[16] DIN EN ISO 105-B07:2009.

Herausforderungen bei der Automatisierung des Layoutentwurfs bei dreidimensionalen heterogenen Systemen

Robert Fischbach, Jens Lienig, Tilo Meister
{fischbach, lienig, meister}@ifte.de
Technische Universität Dresden, Institut für Feinwerktechnik und Elektronik-Design

Kurzfassung

Ein viel versprechender Ansatz, moderne heterogene Systeme zu verwirklichen, sind die in den letzten Jahren entwickelten dreidimensionalen (3D-) Systeme. Hierbei werden durch Verwendung moderner 3D-Integrationstechnologien mehrere Ebenen aktiver Bauelemente übereinander hergestellt. Die sich aus derartigen 3D-Systemen ergebenden neuen Anforderungen an den Layoutentwurf sind Gegenstand dieses Beitrags. Zuerst werden die Unterschiede zwischen dem 2D- und 3D-Layoutentwurf herausgestellt. Anschließend erfolgt eine Untersuchung der Entwurfsschritte Floorplanning, Platzierung und Verdrahtung, um die neuen Herausforderungen beim automatischen Layoutentwurf von 3D heterogenen Systemen aufzuzeigen.

Einleitung

Wie durch das Moore'sche Gesetz vorausgesagt, enthalten moderne hochintegrierte Schaltreise hunderte Millionen Transistoren. Dieser bemerkenswerte Fortschritt wurde hauptsächlich durch das Verkleinern der Halbleiterstrukturen erreicht. Die Kosten für jede weitere Strukturverkleinerung steigen jedoch dramatisch. Um die Leistung zukünftiger Schaltkreise weiterhin zu erhöhen, gewinnen daher zunehmend neue Technologien und Methoden an Bedeutung. Hier sind z.B. die Nutzung von Kupfer an Stelle von Aluminium für die Metallisierungsebenen als auch gestrecktes Silizium (engl. strained silicon) zu nennen. Eine weitere Möglichkeit, den Integrationsgrad von Schaltungen zu steigern, sind 3D-Systeme, bei denen moderne 3D-Integrationstechnologien es ermöglichen, mehrere Ebenen aktiver Bauelemente übereinander anzuordnen (Bild 1).

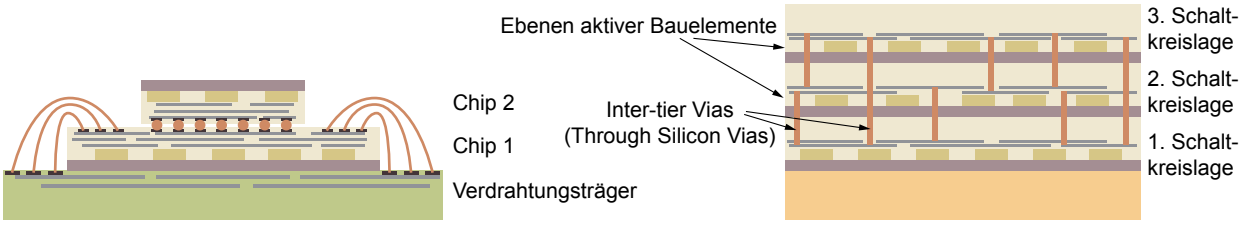


Bild 1: Beispiele von 3D-Systemen mit 3D-Baugruppen (links) und 3D integrierten Schaltkreisen (rechts). Während 3D-Baugruppen die vertikale Integration verschiedener heterogener Technologien erlauben (z. B. Package-on-Package, SoP), ermöglichen 3D integrierte Schaltkreise dichter gepackte Bauelemente aufgrund kürzerer Vertikalabstände.

Die Grundidee bei 3D-Systemen ist das Stapeln mehrerer herkömmlicher Schaltkreise, bestehend aus aktiven Bauelementen (z. B. Transistoren) und deren Verdrahtungsebenen, welche nachfolgend als *Schaltkreislagen* (engl. tier) bezeichnet werden. Hierbei lassen sich auf Wafer-Level Schaltkreise direkt übereinander erzeugen, was zu sog. *3D integrierten Schaltkreisen (3D-ICs)* führt (Bild 1, rechts). Vertikale Verbindungen zwischen den einzelnen Schaltkreisen werden durch sog. *Inter-tier Vias* (auch Through-silicon Vias, TSVs) erzeugt. Eine weitere Realisierungsmöglichkeit von 3D-Systemen ist das Stapeln verschiedener Chips zu einer *3D-Baugruppe*, wobei deren Verbindungen über die Außenanschlüsse (Pads) erfolgen (Bild 1, links).

Es ist offensichtlich, dass 3D-Systeme insbesondere Entwürfe auf Systemlevel, wie etwa System-in-Package (SiP), in denen verschiedene Technologien unabhängig voneinander in verschiedenen Systemkomponenten zum Einsatz kommen, unterstützen. So lassen sich

unterschiedliche Schaltkreisarten, wie Logik- und Speicher-ICs sowie analoge Komponenten, in einer einzelnen Baugruppe integrieren.

Während des Layoutentwurfs erzeugt man die reale geometrische Abbildung einer Schaltung. Dazu wird für jedes Schaltungselement das geometrische Abbild erstellt und ihre räumliche Anordnung (Platzierung), sowie die elektrische Verbindungsstrukturen (Verdrahtung) zwischen ihnen ermittelt. Im Ergebnis liegt die Layoutdarstellung der Schaltung vor, die nach ihrer Verifikation ebenenspezifisch auf Masken übertragen wird und so die Herstellung der integrierten Schaltung ermöglicht.

Da der Übergang von 2D zu 3D einen topologischen Wechsel darstellt, sind der Layoutentwurf und seine inhärenten Entwurfsalgorithmen stark betroffen. Zum Beispiel wird der Lösungsraum durch die zusätzliche dritte Dimension vergrößert. Neben der höheren Komplexität sind auch zusätzliche Randbedingungen zu berücksichtigen. Hierzu zählen z. B. Verdrahtungshindernisse aufgrund der relativ großen Inter-tier Vias als auch die stärkere Einbeziehung von thermischen Kriterien bei der Layoutanordnung.

Dieser Beitrag gibt einen Überblick über den Layoutentwurf im neuen 3D-Kontext. Dazu werden insbesondere Layoutentwurfsschritte, wie Floorplanning, Platzierung und Verdrahtung untersucht, um die neuen Herausforderungen beim Entwurf von 3D heterogenen Systemen aufzuzeigen sowie erste Lösungsansätze zu präsentieren.

Herausforderungen an den 3D-Layoutentwurf

Der Entwurf jedes elektronischen Systems ist ein komplizierter Prozess. Er lässt sich in mehrere Schritte unterteilen, die man oftmals überlappend abarbeitet (Bild 2). Während des Layoutentwurfs werden alle Schaltkreiskomponenten mit ihren geometrischen Repräsentationen abgebildet. Dazu sind alle Makros, Zellen, Gatter, Transistoren, usw., in ihrer geometrischen Darstellung (Form, Größe und Metallisierungsebene) einer Position zuzuweisen (*Floorplanning*, *Platzierung*) und ihre Verbindungen untereinander festzulegen (*Verdrahtung*). Als Ergebnis des Layoutentwurfs liegen Herstellungsspezifikationen vor, welche anschließend nochmals zu überprüfen sind (*Verifikation*).

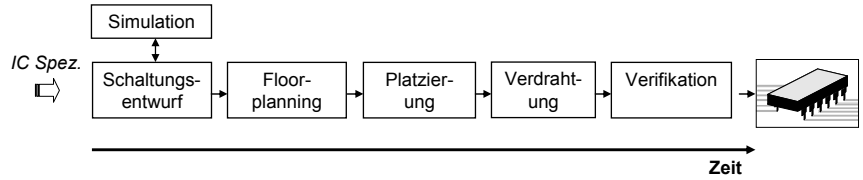


Bild 2: Die Grundschritte im Entwurfsfluss mit dem Schwerpunkt auf den Layoutentwurf. Moderne Entwurfsansätze tendieren dazu, die Grenzen der einzelnen Entwurfsschritte aufzuweichen und ineinander zu integrieren.

Die wesentlichen Grundschritte im 3D-Layoutentwurf sind die gleichen wie beim 2D-Entwurf (wie in Bild 2 dargestellt). Jeder individuelle Entwurfsschritt muss aber die besonderen Randbedingungen der 3D-Integration berücksichtigen. Der Layoutentwurf von 3D-Systemen lässt sich also nicht einfach als ein Stapel mehrerer 2D Entwürfe betrachten.

Nachfolgend sind die wichtigsten 3D-spezifischen Herausforderungen beim Layoutentwurf genannt:

- Netzanschlüsse befinden sich nicht mehr nur in einer aktiven Ebene, womit mehrere Schaltkreislagen umfassende Netztopologien notwendig sind.
- Der Einfluss von Inter-tier Vias unterscheidet sich deutlich von denen „regulärer“ Signalvias. Die maximal erreichbare Dichte von Inter-tier Vias ist mindestens um den Faktor 100 kleiner als die von Signalvias für alle gegenwärtigen und absehbaren 3D-Technologien [1].

– Blockierungen durch thermische und Inter-tier Vias verstärken die notwendige Interaktion zwischen Verdrahtung und der Platzierung aktiver Komponenten. Diese Vias erfordern eine neuartige „Blockade-Berücksichtigung“ insbesondere bei den Verdrahtungsalgorithmen.
– Da eine deutliche höhere Verlustleistungsdichte bei gleichzeitiger verminderter vertikaler Wärmeleitung vorliegt, ist ein komplexeres thermisches Management als bei herkömmlichen Schaltkreisen notwendig;. Somit benötigt man thermische Vias und weitere wärmeabführende Mechanismen, die beim Layoutentwurf zu berücksichtigen sind.

Sowohl zur Verdeutlichung dieser Herausforderungen als auch zum Aufzeigen von Lösungen werden nachfolgend die grundlegenden Schritte des Layoutentwurfs (Floorplanning, Platzierung und Verdrahtung) bei 3D-Systemen betrachtet.

Floorplanning

Während des Floorplanning werden die Form und die Position verschiedener Module (Schaltungspartitionen, wie digitale und analoge Blöcke) bestimmt. Die Floorplanning-Phase bestimmt also die äußeren Eigenschaften – Abmessungen und externe Pin-Positionen – eines jeden Moduls. Diese äußeren Kenndaten sind für die anschließende Platzierung und Verdrahtung notwendig, welche dann die internen Charakteristika des Moduls bestimmen. Floorplanning arbeitet also mit Zellengruppen anstatt mit individuellen Zellen, welche erst in den späteren Entwurfsphasen berücksichtigt werden.

Beim klassischen Floorplanning wurde bisher von einer einzelnen (planaren) Modulebene ausgegangen, auf welcher die Module anzuordnen sind. Hierfür entwickelte man eine Vielzahl verschiedener algorithmischer Ansätze. Bei 3D-Systemen mit *mehreren* Modulebenen müssen nun neue, 3D spezifische Eigenschaften des Floorplanning abgebildet werden. So haben Module beim 3D-Entwurf beispielsweise zusätzlich zu horizontalen auch vertikale Abhängigkeiten.

Es bieten sich zwei Möglichkeiten an, vertikale Abhängigkeiten abzubilden. Die Erste ist das mehrfache Verwenden klassischer (also für den 2D-Entwurf entwickelter) Datenstrukturen, der sog. 2,5D-Ansatz. Dabei sind zusätzliche Mechanismen zu implementieren, um die vertikalen Beziehungen zwischen den Modulen in unterschiedlichen Schaltkreislagen zu berücksichtigen. Beispiele hierfür sind das vertikale Ausrichten von Modulen als auch die Berücksichtigung von deren Überlappungen bzw. Überlappungsfreiheiten. Ein erfolgreicher Ansatz hierfür ist die „Combined-Bucket-Methode“ [2].

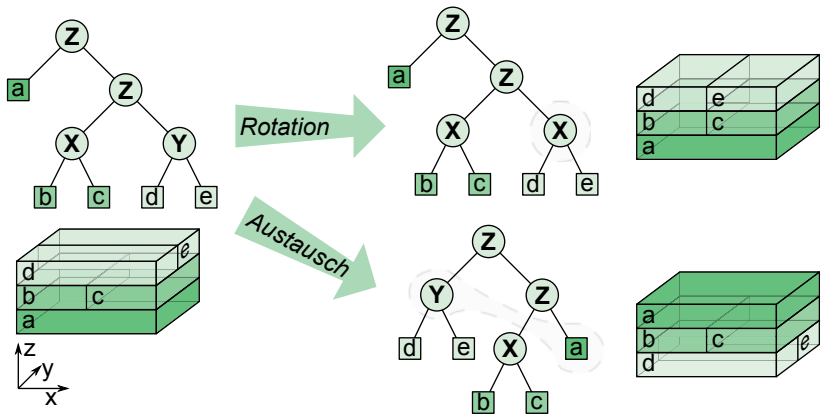


Bild 3: Darstellung der 3D-Schnittbaum-Operationen zur Permutation eines gegebenen 3D-Floorplans. Eine Rotation verändert einen inneren Knoten, welcher die Schnittebene vorgibt und führt zu einer physischen Rotation der Module des betroffenen Teilbaums. Ein Austausch wechselt zwei Teilbäume und resultiert damit auch in einem physischen Tausch der Module in den beiden Teilbäumen.

Es wurde jedoch schnell offensichtlich, dass vertikale Abhängigkeiten besser direkt in die Datenstrukturen zu integrieren sind. Aktuellere 3D-Datenstrukturen für das Floorplanning bilden daher mehrlagige Module kontinuierlich in allen *drei* Dimensionen ab. Der 3D-Schnittbaum [3] ist ein

Beispiel für eine solche 3D-Datenstruktur. Wie in Bild 3 dargestellt, kann man hier verschiedene Operationen, wie Modulrotation oder -tausch, ausführen und kombinieren, um so jeden möglichen Schnittbaum zu generieren. Lösungen, die mit einem 3D-Schnittbaum generiert werden, sind jedoch immer auf geschnittene Floorpläne begrenzt.

Platzierung

Nachdem durch das Floorplanning die äußeren Modulgrenzen und Pinpositionen bestimmt sind, ist der nächste Schritt im Entwurfsfluss die Platzierung. Hier wird die Position jeder Zelle innerhalb des jeweiligen Moduls (Partition) festgelegt. Dies erfolgt unter Einhaltung gegebener Randbedingungen (z. B. Überlappungsfreiheit zwischen Zellen) sowie dem Anstreben von Optimierungszielen (z. B. Minimieren der Gesamtverdrahtungslänge).

Während die Platzierung beim herkömmlichen 2D-Entwurf auf eine Ebene begrenzt ist, muss die 3D-Platzierung die Anordnung der Zellen auf mehreren aktiven Ebenen beherrschen. Hinzu kommen neue thermische Randbedingungen, die oft beim 2D-Entwurf vernachlässigbar waren oder im „Nachgang“ einbezogen wurden. Bei der 3D-Platzierung ist die Berücksichtigung thermischer Gesichtspunkte *während* der Platzierung sicherzustellen. Neben der Ermittlung von Temperaturverteilungen werden aufgrund der höheren Packungsdichte weitere zusätzliche Methoden benötigt. Dazu zählen vertikale Metallverbindungen, sog. *thermische Vias*. Diese sind zum Ableiten der Wärme, d. h. für das Erreichen einer gültigen thermischen Lösung, bei 3D-Systemen unverzichtbar. Grundsätzlich wirkt ein thermisches Via als Wärmeleitungselement, um thermische Energie von hochtemperierten Regionen, wie Zellenansammlungen innerhalb des Chips, zu einer Wärmesenke, wie etwa ein Kühlkörper auf der Oberseite der Baugruppe, abzuführen.

Während der 3D-Platzierung der Zellen sind damit die Positionen der relativ großen thermischen Vias zu berücksichtigen. Ein thermisches Via erstreckt sich über mehrere Schaltkreislagen und stellt somit ein Hindernis für die Zellenplatzierung und die Verdrahtung dar. (Herkömmliche Signalvias befinden sich nur auf den Metallisierungsebenen und sind damit nur für die Verdrahtung relevant.) Weiterhin ist die Platzierung der Zellen und der thermischen Vias miteinander verknüpft, da die Größe eines thermischen Vias von der Leistungsdichte der Zellen in seiner Umgebung abhängt. Eine Chip-umfassende thermische Analyse mit der Granularität individueller thermischer Vias wäre nötig, um diese Abhängigkeit aufzulösen. Dies würde aber zu einer unangemessenen thermischen Simulationsmatrix führen.

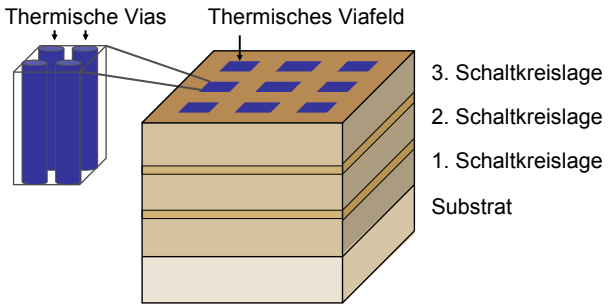


Bild 4: Regelmäßig verteilte Regionen thermischer Vias in einem 3D-IC zur vereinfachten Berücksichtigung thermischer Vias während der Platzierung.

Trotz des hohen Schwierigkeitsgrades dieser miteinander verknüpften Problemstellungen sind seit einigen Jahren hierfür Lösungen bekannt. So lässt sich z. B. durch vorgegebene, potentielle Viafelder, die von vornherein für die Platzierung und Verdrahtung gesperrt sind, eine Entkopplung der einzelnen Probleme erreichen (Bild 4). Unter Nutzung solcher Viafelder wird in [4] die Platzierung der thermischen Vias darauf beschränkt, die Viadichte in diesen zugelassenen Regionen zu ermitteln. Ein Vorteil dieser Methode ist die grobe Granularität, mit der die thermische Analyse durchgeführt werden kann. Die thermische Leitfähigkeit jeder Region

(Element) lässt sich als eine Entwurfsvariable behandeln, welche nachträglich in die präzise Anzahl nötiger thermischer Vias in dieser Region umgerechnet wird. Ein weiterer Vorteil ist die Regelmäßigkeit der erzeugten Hindernisse, die man so viel leichter bei der nachfolgenden Verdrahtung berücksichtigen kann.

Ein 3D-Platzierer, welcher auf der Kräfteplatzierung beruht, ist in [5] beschrieben. Thermische Randbedingungen werden hier direkt in eine Finite-Elemene-Analyse (FEA) einbezogen. Laut einer Analyse in [6] führt die relativ kleine Anzahl von Schaltkreislagen (Tiers) dazu, dass sich partitionierende Platzierungsansätze besser als kräftebasierende für den 3D-Grobentwurf eignen. Die Autoren stellen dazu einen hierarchischen Platzierer vor, der aus einer Globalplatzierung und einem Legalisierungsschritt besteht. Thermische Randbedingungen werden durch sog. „thermal resistance reduction nets“ berücksichtigt, welche abstoßende Kräfte darstellen. Damit lassen sich hochstromige Netze bevorzugt in der Nähe von Wärmesenken verlegen.

Verdrahtung

Ein Netz besteht aus mindestens zwei Zellpins gleichen elektrischen Potentials. In der Netzliste sind alle Netze eines Entwurfs enthalten. Während der Verdrahtung sind die Anschlüsse aller Netze mittels einer kreuzungsfreien Einbettung auf den verschiedenen Verdrahtungsebenen zu verbinden. Dabei sind Randbedingungen (z. B. Entwurfsregeln, Verdrahtungskapazitäten) einzuhalten und Optimierungsziele (z. B. Minimierung der Gesamtverdrahtungslänge, Maximierung der Takttoleranz) zu verfolgen.

Wie bereits erwähnt, liegt der Hauptunterschied zwischen 2D- und 3D-Verdrahtung darin, dass die Netzanschlüsse, und damit die Netze selbst auch, über mehrere Schaltkreislagen verteilt auftreten können (Bild 5). Deshalb werden zusätzlich zu den regulären Signalvias aufwändige (platzintensive) Inter-tier Vias benötigt, um auch die Metallebenen *verschiedener* Schaltkreislagen (Tiers) miteinander zu verbinden. Außerdem ist bei der 3D-Verdrahtung auf zusätzliche Randbedingungen zu achten, wie beispielsweise Blockierungen durch thermische und Inter-tier Vias. Dies führt zu anspruchsvollerem Ressourcenmanagement, insbesondere im Bereich der globalen Verdrahtung, und begünstigt Blockierungen im Vergleich zur regulären 2D-Verdrahtung. Die begrenzte Verfügbarkeit von Inter-tier Vias verlangt weiterhin die umsichtige Verteilung dieser wertvollen Ressource unter den Netzen. Der verstärke Einfluss der Temperatur im 3D-Entwurf muss ebenso bedacht werden. Kritische Netze sollten beispielsweise die heißesten Regionen eines Chips vermeiden, da ein Temperaturanstieg die Signalverzögerung eines Netzes erhöht.

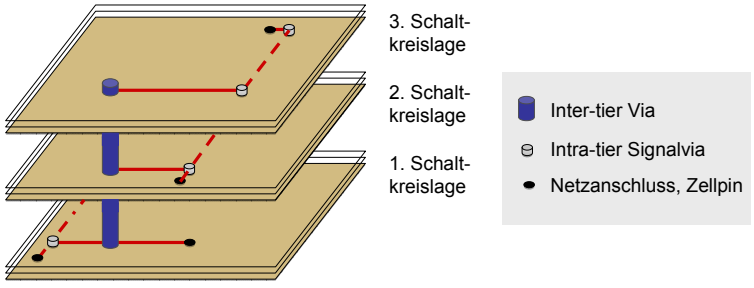


Bild 5: Beispiel für die Verdrahtung eines Netzes in einem 3D-System bestehend aus drei Lagen (Tiers), wobei im Gegensatz zum konventionellen Entwurf die Netzanschlüsse über mehrere aktive Ebenen in den verschiedenen Tiers verteilt sind.

Derartigen Anforderungen werden neue Verdrahter zunehmend gerecht, insbesondere durch die Einbeziehung von thermischer Dimensionierung und einer hierarchischen Vorgehensweise. Ein temperaturgesteuerter, mit unterschiedlichen Hierarchiestufen und Granularitäten arbeitender 3D-Verdrahter wird in [2] vorgestellt. Er zeichnet sich durch eine thermisch beeinflusste Vorabplanung der Vias aus. Ein weiterer erfolgreicher 3D-Verdrahter ist in [7] beschrieben. Hier werden nicht nur thermische Vias zur Verlustleistungsabführung benutzt, sondern auch das

Konzept von thermischen Leiterzügen (engl. thermal wires) eingeführt. Diese dienen zur lateralen Wärmeleitung. Während die thermischen Vias die hauptsächliche Wärmeleitung aus der Baugruppe heraus vornehmen, vernetzen die thermischen Leiterzüge die thermischen Vias untereinander.

Zusammenfassung

Das Hinzufügen einer dritten Dimension bei derzeitigen 2D-Schaltkreisen erhöht die Integrationsdichte, verringert Verbindungslängen und ermöglicht den Aufbau kompakter heterogener Systeme.

Um die Vorteile der 3D-Integration nutzen zu können, ist der Entwurfsprozess an die höhere Komplexität und die neuen Randbedingungen anzupassen. Dazu ist die Entwicklung geeigneter Layoutwerkzeuge notwendig, welche die engere Verknüpfung zwischen Verdrahtung und den Eigenschaften der aktiven Komponenten berücksichtigen. Diese Verknüpfungen resultieren unter anderem aus dem komplexeren thermischen Management und den vielfältigen Verdrahtungsblockierungen, die durch thermische und Inter-tier Vias verursacht werden.

Gegenwärtige Hindernisse einer weitergehenden 3D-Integration sind im Wesentlichen die schlechte vertikale thermische Leitfähigkeit aufgrund der verwendeten Materialien und die damit unzureichende Wärmeabfuhr bei gleichzeitig hoher Leistungsdichte. Dieses führt zu bisher unbekannten thermischen Problemstellungen. Die hier aufgeführten 3D-Entwurfsansätze zeigen jedoch, dass die sorgfältige Platzierung thermischer Vias effektiv zur Kontrolle der Temperaturen in 3D-ICs beitragen kann. Alternative Methoden, wie thermische Leiterzüge, sind ebenfalls auf dem Vormarsch.

Dieser Beitrag fasst den aktuellen Stand beim modernen 3D-Layoutentwurf zusammen. Obwohl die Komplexität des 3D-Entwurfs eine enorme Herausforderung darstellt, wird dabei offensichtlich, dass neue maßgeschneiderte Ansätze dieser Aufgabe immer besser gewachsen sind.

Literatur

[1] International Technology Roadmap for Semiconductors, ESIA, JEITA, KSIA, TSIA and SIA, 2007, http://www.itrs.net/reports.html

[2] J. Cong, J. Wei and Y. Zhang, A thermal-driven floorplanning algorithm for 3D ICs, International Conf. on Computer Aided Design (ICCAD), 2004, pp. 306-313

[3] L. Cheng, L. Deng and M.D.F. Wong, Floorplanning for 3-D VLSI design, Asia and South Pacific Conf. on Design Automation (ASP-DAC), 2005, pp. 405-411

[4] B. Goplen and S. Sapatnekar, Thermal via placement in 3D ICs, International Symposium on Physical Design (ISPD), 2005, pp. 167-174

[5] S. Das, Design Automation and Analysis of Three-Dimensional Integrated Circuits, PhD Thesis, Massachusetts Institute of Technology, 2004

[6] B. Goplen and S. Sapatnekar, Efficient thermal placement of standard cells in 3D ICs using a force directed approach, International Conf. on Computer Aided Design (ICCAD), 2003, pp. 86-89

[7] T. Zhang, Y. Zhan and S.S. Sapatnekar, Temperature-aware routing in 3D ICs, Asia and South Pacific Conf. on Design Automation (ASP-DAC), 2006, pp. 309-314

Ein Ansatz zur Modellierung CNT-basierter thermischer Vias für den effektiven Wärmetransport in elektronischen Schaltkreisen

Jörg Hertwig, Holger Neubert, Jens Lienig
Technische Universität Dresden
Institut für Feinwerktechnik und Elektronik-Design
01062 Dresden

Kurzfassung

Thermische Vias sind wichtige Elemente zur Verlustleistungsabfuhr in elektronischen Baugruppen. Die fortschreitende Miniaturisierung verlangt ständig sowohl besser Wärme leitende als auch kleinere thermische Vias. Ein großes Potenzial, diese Forderungen zu erfüllen, liegt in der Anwendung von Kohlenstoffnanoröhren (CNTs) als Viamaterial. Der Beitrag stellt ein Mehrskalen-Modell thermischer Vias und Viafelder mit CNT-Kompositwerkstoffen vor. Sein Kerngedanke besteht in der Berechnung der Eigenschaften einer Basiszelle aus Matrixwerkstoff, in die CNTs als Linienelemente eingebettet sind, mit der Finite-Elemente-Methode. Die berechneten Eigenschaften fließen in höhere Modellebenen ein. Damit gelingt erstmals die Modellierung CNT-basierter thermischer Vias und Viafelder. Das Ziel ist es, funktionell lohnende Ansatzpunkte für technologisch orientierte Forschungsarbeiten zu bestimmen.

Einleitung

Als unvermeidliches Nebenprodukt fällt in elektronischen Bauelementen und Leitungen Wärme an, mit negativen Auswirkungen auf die Funktion und Zuverlässigkeit. Die frei gesetzte Wärme muss von den Quellen zu Wärmesenken, z. B. in ein Kühlmedium, und letztlich in die Umgebung transportiert werden. Dazu sind entsprechende Strukturen und Elemente des thermischen Managements erforderlich. Diese sind bereits beim Entwurf elektronischer Baugruppen, z. B. von elektronischen Schaltkreisen, zu berücksichtigen. Die stetige Verkleinerung von Funktionselementen elektronischer Systeme bis hinab zu Nanometerabmessungen führt zu einer Vielzahl von Herausforderungen im Entwurfsprozess, wobei sich die thermischen wegen der permanent steigenden Verlustleistungsdichten als besonders kritisch erweisen.

Ein thermisches Modell der elektronischen Baugruppe beschreibt die Wärmepfade, entlang derer die Verlustleistung abgeführt wird. Ein Wärmepfad repräsentiert den thermischen Widerstand R_{th} von der Wärmequelle mit der Leistung P zur Wärmesenke:

$$R_{\mathrm{th}} = \frac{\Delta T}{P}, \tag{1}$$

wobei zwischen Quelle und Senke die Temperaturdifferenz ΔT entsteht. Bei eindimensionaler Wärmeleitung durch gerade prismatische Festkörper, wie sie typisch in elektronischen Bauelementen und Baugruppen sind, kann Gl. (1) spezifiziert werden:

$$R_{\mathrm{th}} = \frac{L}{\lambda \cdot A}, \tag{2}$$

mit der Länge L, der Querschnittsfläche A und der spezifischen thermischen Leitfähigkeit λ des prismatischen Körpers.

Ein besonderer Engpass der Wärmeübertragung ist der senkrechte Durchgang durch Leitungsebenen in Leiterplatten und Schaltkreisen. Gegenwärtig verwendete organische Substrate haben meistens eine geringe Wärmeleitfähigkeit im Bereich von 0,2 bis 0,5 W/(m·K), keramische von 1 bis 100 W/(m·K). Die anisotrope Struktur derartiger Ebenen hat zudem eine anisotrope Wärmeleitfähigkeit zur Folge. Aufgrund der höheren thermischen Leitfähigkeit der elektrischen Verbindungen im Vergleich zum Substratwerkstoff ist der Wärmetransport in der Leitungsebene rund zehnmal besser als senkrecht durch das Substrat hindurch.

Thermische Vias

Ein etabliertes Prinzip, Wärme durch Substrate zu leiten, sind thermische Vias. Ein thermisches Via ist eine Durchkontaktierung zwischen mindestens zwei Leiterebenen eines Verdrahtungsträgers aus gut Wärme leitendem Werkstoff mit der Funktion, den Wärmewiderstand zwischen den kontaktierten Ebenen zu verringern (
Bild 1).

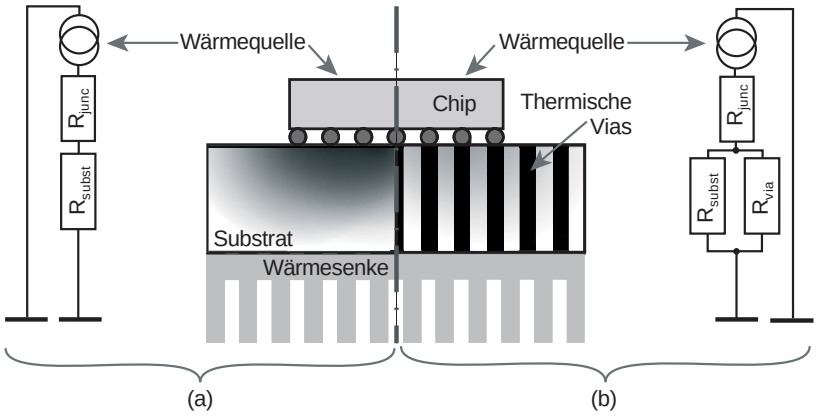


Bild 1: Wärmepfade und zugehöriges Netzwerkmodell (a) ohne und (b) mit thermischen Vias.

Im Allgemeinen wird für thermische Vias Kupfer eingesetzt, da es eine gute Wärmeleitfähigkeit besitzt, relativ kostengünstig und ohnehin in viele Prozessschritte integriert ist. Jedoch stößt die Anwendung solcher konventioneller thermischer Vias mit der weitergehenden Miniaturisierung an Grenzen, besonders in Schaltkreisen mit großer Verlustleistung. Der Trend zu 3D-Anordnungen verschärft diese Problematik durch die damit verbundenen höheren Verlustleistungsdichten und längeren Wärmepfade. Das Potential, das sich aus der optimierten Platzierung von thermischen Vias ergibt, ist bereits weitgehend ausgeschöpft [9]. Künftig werden daher thermische Vias mit geringerem Wärmewiderstand bei ungefähr gleichen oder sogar kleineren Abmessungen verlangt. Nach Gl. (2) gibt es drei Möglichkeiten, den Wärmewiderstand eines thermischen Vias zu senken:

1. Verkleinern der Länge des Wärmepfades, d. h. Verringern der Schichtdicke des Substrates oder des Verdrahtungsträgers,
2. Vergrößern der Querschnittsfläche des Vias sowie
3. Erhöhen der Wärmeleitfähigkeit des Viamaterials.

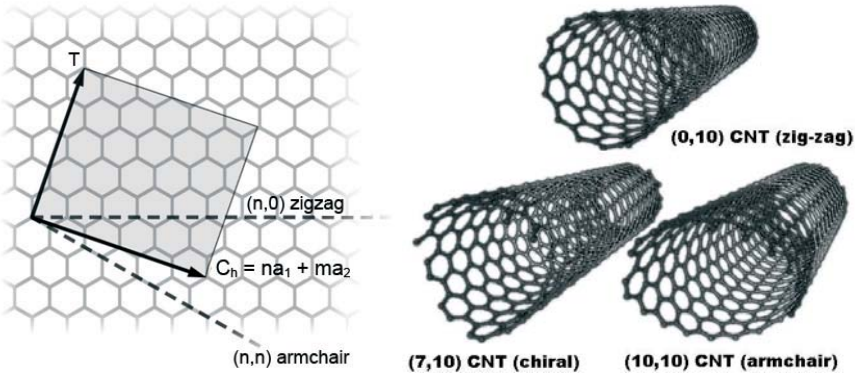
Der Verkleinerung der Schichtdicke des Substrates oder des Verdrahtungsträgers sind durch die notwendige mechanische Stabilität und die erforderlichen Lagenanzahl enge Grenzen gesetzt. Ebenfalls nur begrenzt anwendbar ist eine Vergrößerung der Querschnittsflächen, da die Fläche der thermischen Vias der elektrischen Verdrahtung verloren geht. Allein die Möglichkeit, durch Anwendung hochleitfähiger Werkstoffe zu kleineren thermischen Widerständen der Vias zu kommen, ist vielversprechend. Eine besondere Rolle werden hierbei in Zukunft Kohlenstoffnanoröhren spielen.

CNT-basierte Kompositwerkstoffe

Die Wärmeleitfähigkeit von Kohlenstoffnanoröhren (engl. **C**arbon **N**ano**T**ubes, CNTs) beträgt theoretisch 6000 W/(m·K); gemessen wurden Werte zwischen 2000 und 3000 W/(m·K) [12]. Damit sind sie die besten bekannten Wärmeleiter und etwa zehnmal bessere als Metalle.

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 (Bild **2**.a). Sie heißen armchair (*n*, *n*), zig-zag (*n*, 0) und chiral (*n*, *m*) (Bild 2.b). CNTs können außerdem einwandig (engl. Single-Walled Nanotubes, SWNTs) oder mehrwandig (engl. Multi-Walled Nanotubes, MWNTs) sein.



(a) (b)

Bild 2: Benennungsschema für CNTs repräsentiert durch (a) den Vektor des Rollwinkels der monomolekularen Kohlenstoffschicht und (b) unterschiedliche Chiralitäten [26].

Mit CNTs lassen sich Kompositwerkstoffe herstellen, die die Eigenschaften von Kompositmatrix und eingebetteten CNTs vorteilhaft kombinieren. In den bisher veröffentlichten Untersuchungen zu solchen Kompositen in thermischen Anwendungen wurden die CNTs meist ungeordnet in das Matrixmaterial gegeben, um die thermische Leitfähigkeit zu erhöhen. In der Regel bilden Kunststoffe die Matrix des Komposits, meist ein Epoxydharz. CNTs werden in einem Volumenanteil von 1 bis 10 Masseprozent eingebracht. Die thermische Leitfähigkeit solcher Komposite steigt gegenüber dem ungefüllten Basismaterial um einen Faktor bis 3,6 [13]. Untersuchungen mit anderen Polymeren und Metallen als Matrixmaterial, z. B. Kupfer und Aluminium, haben vergleichbare Ergebnisse [7, 25].

Offensichtlich verbessern ungeordnete, nicht ausgerichtete CNTs in einem Matrixwerkstoff die thermische Leitfähigkeit nur wenig. Dies ist verständlich, da erstens der Wärmetransport in CNTs hauptsächlich axial stattfindet und zweitens sich die ungeordneten CNTs kaum berühren, also zwischen ihnen die Wärmeleitung von der Matrix bestimmt ist. Um das Potenzial der hohen thermischen Leitfähigkeit von CNTs weitergehend zu nutzen, sind CNTs daher auszurichten und zu kontaktieren. Zur Ausrichtung sind folgende Möglichkeiten bekannt:

- strömungsmechanische Ausrichtung durch Einschwemmen in schmale Kanäle [15],
- Ausrichtung in einem elektrischen Feld durch Dielektrophorese [8, 13],
- gerichtetes Wachstum der CNTs auf einem Katalysator [1, 3],
- Rotationsbeschichtung [14] und
- manuelles Platzieren einzelner CNTs auf einer funktionalisierten Oberfläche [2, 23].

Die vorgestellten Ansätze erlauben die Ausrichtung einer geringen Anzahl von CNTs. Ihre Anwendung für Komposite ist nach wie vor eine Herausforderung. Zu beachten ist, dass Komposite mit ausgerichteten CNTs die Wärme anisotrop leiten.

Um mechanischen und damit besseren thermischen Kontakt von CNTs zu angrenzenden Festkörperoberflächen herzustellen, gibt es ebenfalls verschiedene Möglichkeiten:

- Löten mit metallischem Lot [6],
- Kontaktieren mit funktionalisierten Gruppen [5],
- galvanisches Verbinden von CNTs [13] oder

- mechanische Verbindung der CNTs [8].

Während ein mechanischer Kontakt zwischen CNT und angrenzender Festkörperoberfläche den thermischen Übergangswiderstand in der Grenzfläche verringert, erhöht sich der Wärmewiderstand der CNTs, wenn sie sich gegenseitig mit ihren Mantelflächen berühren. Dies beruht auf der Störung des ballistischen Phonentransportes in den CNTs [10]. Da CNTs dazu neigen, zusammenzuklumpen, ist die Ausrichtung unter weitgehender Vermeidung von gegenseitigen Berührungen in Kompositen eine gegenwärtig ungelöste Herausforderung [27].

Das Anwendungspotential der verschiedenen technologischen Ansätze für thermische Vias lässt sich aufgrund des komplexen Verhaltens von CNTs in Kompositen derzeit nur durch eine Vielzahl experimenteller Untersuchungen bestimmen. Deshalb sind Modelle wünschenswert, mit denen sich das zu erwartende thermische Verhalten von Elementen vorhersagen lässt, die nach diesen technologischen Ansätzen hergestellt sind. Ziel dieses Beitrages ist es daher, im Hinblick auf die Funktion lohnende technologische Entwicklungsansätze von thermischen Vias zu bestimmen.

Modellierung des thermischen Verhaltens von CNT-Kompositen

Die bisher veröffentlichten Modelle CNT-basierter Komposite basieren in der Regel auf der Theorie des effektiven Mediums. Sie unterscheiden sich hauptsächlich in den Vereinfachungen zur Modellierung der in die Matrix eingebetteten CNTs, in der Dimensionalität des Ansatzes und den verwendeten Berechnungsverfahren. Die Vereinfachungen modellieren die CNTs als Rotationsellipsoide [4], Hohlzylinder [22, 29], prismatische Stäbe [21, 24] oder Linien [20]. Den Berechnungen liegen zwei- [20] oder dreidimensionale [21, 22, 29] Basiszellen zugrunde. Als Lösungsverfahren werden die Randelementemethode [16], Randintegralgleichungen [20], die Finite-Elemente-Methode [22, 29] und die Element-freie Galerkin-Methode [11] eingesetzt. Manche Untersuchungen basieren auf analytischen Lösungen [19, 18, 28] oder Methoden der dissipativen Teilchendynamik [17]. Stochastische Aspekte in der Anordnung und den Eigenschaften der CNT sind kaum berücksichtigt [21].

Alle diese Untersuchungen beziehen jeweils nur ausgewählte Teilaspekte in die Modellierung ein. Sie sind damit ungeeignet, Strukturen in den typischen Abmessungen thermischer Vias und Viafelder zu modellieren, die um Größenordnungen über denen der CNTs und der Basiszellen liegen. Ein dafür geeigneter Ansatz muss, wenn er die Eigenschaften einer Basiszelle des effektiven Mediums berechnet, folgende Effekte berücksichtigen:

- Wärmetransport in den eingebetteten CNTs,
- Einfluss von gegenseitigen Berührungen der CNTs in der Matrix,
- Wärmetransport in der Matrix und
- Wärmeübergang zwischen CNTs und der Matrix.

Dazu wird ein hierarchischer Modellierungsansatz mit den Modellebenen CNT, Komposit-Basiszelle und thermisches Via bzw. Viafeld mit jeweils angepassten Detaillierungsgrad vorgestellt (Bild 3). 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.

Bedingt durch das hohe Aspektverhältnis und aufgrund der spezifischen Mechanismen des Wärmetransports findet die Wärmeleitung in CNTs nahezu eindimensional in Längsrichtung statt. Vernachlässigt man transiente Prozesse und beschränkt sich auf die Wärmeleitung in Längsrichtung, vereinfacht sich der Wärmewiderstand eines CNT zu:

$$R_{\mathrm{th-CNT}} = \frac{L_{\mathrm{CNT}}}{\Lambda}\,. \tag{3}$$

L_{CNT} ist die Länge des CNT und Λ ist ein CNT-spezifischer Parameter, der die auf eine Längeneinheit bezogene thermische Leitfähigkeit des CNT repräsentiert. Diese ist von vielen Einflussfaktoren abhängig, z. B. dem Typ, der Länge und dem Durchmesser eines CNT. Einen wesentlichen Einfluss hat auch die Temperatur. Eine Bestimmung des Parameters Λ ist mit molekulardynamischen Simulationen oder experimentellen Untersuchungen möglich, wobei beide Möglichkeiten aufwendig sind.

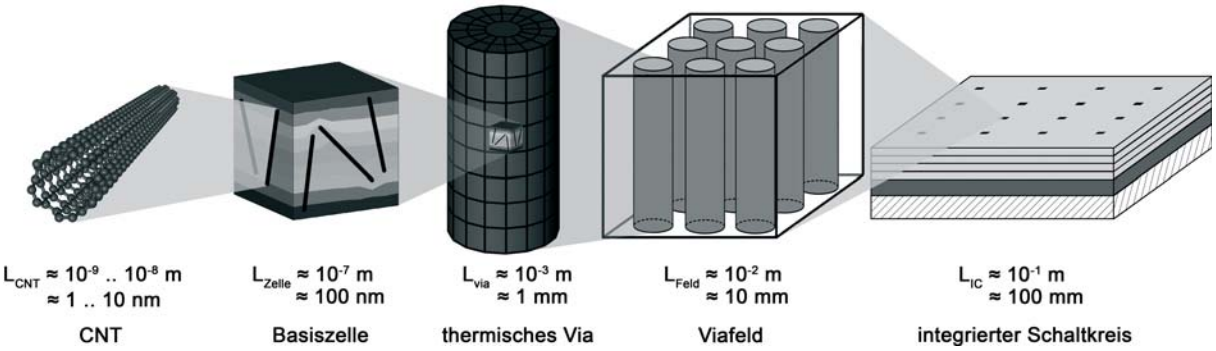


Bild 3: Im Rahmen dieser Arbeit entwickeltes Mehrskalenmodell thermischer Viafelder mit CNT-Kompositen.

Die Basiszelle des Komposits ist nach der Finite-Elemente-Methode modelliert, wobei die eingebetteten CNTs als Ketten von Linienelementen gemäß Gl. (3) in der dreidimensionalen Basiszelle dargestellt sind. Position und Orientierung der Linienelemente in der Basiszelle berücksichtigen die zufällige Anordnung der CNTs im Komposit mit Verteilungsfunktionen. Der Λ-Parameter jedes Linienelements hängt vom Abstand zum nächstgelegenen Knoten der Matrix oder benachbarter Ketten von Linienelementen ab. Damit ist der Einfluss von Kontakten von CNTs untereinander und mit der Matrix modelliert.

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

Zusammenfassung und Ausblick

Das entwickelte Mehrskalenmodell gestattet Parameterstudien hinsichtlich Typ und Abmessungen der CNTs, ihrer Anordnung im Komposit sowie Einflussfaktoren der Umgebung. Damit lässt sich die Wirkung technologischer Ansätze, z. B. zur Ausrichtung und Kontaktierung der CNTs, vergleichen und bewerten. Auch ist der optimale Volumenanteil von CNTs im Komposit abhängig von ihrer Länge und ein hinreichender Ausrichtungsgrad ermittelbar. Dabei werden erstmalig solche Einflüsse wie CNT-Durchmesser, Volumenanteil, Ausrichtung oder lokale inhomogene Anordnung gemeinsam in einem dreidimensionalen Modell erfasst. Perspektivisch lassen sich mit den Modellen die Eigenschaften verschiedener Anordnungen von thermischen Vias in Viafeldern simulieren und optimieren.

Literatur

[1] Yoshinori Ando, Xinluo Zhao, Toshiki Sugai, and Mukul Kumar. Growing carbon nanotubes. *Materials Today*, 7(10):22–29, October 2004.

[2] Phaedon Avouris, Zhihong Chen, and Vasili Perebeinos. Carbon-based electronics. *Nature Nanotechnology*, 2:605–615, October 2007.

[3] Yuji Awano. Carbon nanotube technologies for LSI via interconnects. *IEICE Transactions on Electronics*, E89-C(11):1499–1503, November 2006.
[4] Aniruddha Bagchi and Seiichi Nomura. On the effective thermal conductivity of carbon nanotube reinforced polymer composites. *Composites Science and Technology*, 66:1703–1712, 2006.
[5] Vasudevanpillai Bijua, Tamitake Itoha, Yoji Makitab, and Mitsuru Ishikawa. Close-conjugation of quantum dots and gold nanoparticles to sidewall functionalized single-walled carbon nanotube templates. *Journal of Photochemistry and Photobiology A: Chemistry*, 183:315–321, 2006.
[6] D.-A. Borca-Tasciuc, L. Pietruszka, T. Borca-Tasciuc, R. Vajtai, and P.M. Ajayan. Thermal transport measurements in multi-wall carbon nanotube strands using the 3/spl omega/ method. In *Semiconductor Thermal Measurement and Management Symposium, 2005 IEEE Twenty First Annual IEEE*, pages 247–252, 15-17 March 2005.
[7] Yang Chai, Kai Zhang, Min Zhang, P. C. H. Chan, and M. M. F. Yuen. Carbon nanotube/copper composites for via filling and thermal management. In *Proc. 57th Electronic Components and Technology Conf. ECTC '07*, pages 1224–1229, May 29 2007–June 1 2007.
[8] S. Evoy, M.A. Riegelman, N. Naguib, Haihui Ye, P. Jaroenapibal, D.E. Luzzi, and Y. Gogotsi. Dielectrophoretic assembly of carbon nanofiber nanoelectromechanical devices. *Nanotechnology, IEEE Transactions on*, 4(5):570–575, Sept. 2005.
[9] B. Goplen and S.S. Sapatnekar. Placement of thermal vias in 3-D ICs using various thermal objectives. *Computer-Aided Design of Integrated Circuits and Systems*, 25(4):692–709, April 2006.
[10] Dimitrios V. Papavassiliou Shigeo Maruyama Hai M. Duong, Namiko Yamamoto and Brian L. Wardle. Inter-carbon nanotube contact in thermal transport of controlled-morphology polymer nanocomposites. *Nanotechnology*, 20:1–23, March 2009.
[11] Morinobu Endo Indra Vir Singh, Masataka Tanaka. Effect of interface on the thermal conductivity of carbon nanotube composites. *ScienceDirect*, 46:842–847, 2007.
[12] P. Kim, L. Shi, A. Majumdar, and P.L. McEuen. Thermal transport measurements of individual multiwalled nanotubes. *Physical Review*, 87:1–4, 2001.
[13] Torsten Koker. *Konzeption und Realisierung einer neuen Prozesskette zur Integration von Kohlenstoff-Nanoröhren über Handhabung in technischen Anwendungen*. Dissertation, Schriftenreihe des Instituts für Automatisierungstechnik, Universität Karlsruhe (TH), Band 14, 2006.
[14] Melburne C. LeMieux, Mark Roberts, Soumendra Barman, Yong Wan Jin, Jong Min Kim, and Zhenan Bao. Self-sorted, aligned nanotube networks for thin-film transistors. *Science*, 321(5885):101–104, July 2008.
[15] J. Li, Q. Zhang, Y. Yan, S. Li, and L. Chen. Fabrication of carbon nanotube field-effect transistors by fluidic alignment technique. *Nanotechnology, IEEE Trans. on*, 6(4):481–484, July 2007.
[16] Y. J. Liu and X. L. Chen. Continuum models of carbon nanotube-based composites using the boundary element method. *Electronic Journal of Boundary Elements*, 1:316–335, 2003.
[17] Amitesh Maiti. Multiscale modeling with carbon nanotubes. *Microelectronics Journal*, 39:208–221, 2008.
[18] C.-W. Nan, Z. Shi, and Y. Lin. A simple model for thermal conductivity of carbon nanotube-based composites. *Chemical Physics Letters*, 375:666–669, 2003.
[19] Ce-Wen Nan, Gang Liu, Yuanhua Lin, and Ming Li. Interface effect on thermal conductivity of carbon nanotube composites. *Applied Physics Letters*, 85:3549–3551, 2004.
[20] N. Nishimura and Y. J. Liu. Thermal analysis of carbon-nanotube composites using a rigid-line inclusion model by the boundary integral equation method. *Comput Mech*, 35:1–10, 2004.
[21] K. Sanada, Y. Tada, and Y. Shindo. Thermal conductivity of polymer composites with close-packed structure of nano and micro fillers. *Composites: Part A*, 40:724–730, 2009.
[22] Young Seok Song. Evaluation of effective thermal conductivity for carbon nanotube/polymer composites using control volume finite element method. *Carbon*, 44:710–717, 2006.
[23] Ramsey Stevens, Cattien Nguyen, and M. Meyyappan. Nanomanipulation and fabrication by ion beam molding. *Nanotechnology, IEEE Trans. on*, 5(3):255–257, May 2006.
[24] Konstantinos I. Tserpes and Paraskevas Papanikos. Continuum modeling of carbon nanotube-based superstructures. *Composite Structures*, 91:131–137, 2009.
[25] Toshiyuki Ueno, Takashi Yoshioka, Jin ichi Ogawa, Nobuaki Ozoe, Kiminori Sato, and Katsumi Yoshino. Highly thermal conductive metal/carbon composites by pulsed electric current sintering. *Synthetic Metals*, 159:2170–2172, 2009.
[26] Wikipedia. http://en.wikipedia.org/wiki/file:cntnames.png, 01.10.2005.
[27] M. Wirts-Rutters, M. Heimann, J. Kolbe, and K. J. Wolter. Carbon nanotube (CNT) filled adhesives for microelectronic packaging. In *Proc. 2nd Electronics Systemintegration Technology Conf. ESTC 2008*, 1057–1062, 1–4 Sept. 2008.
[28] Q.Z. Xue. Model for thermal conductivity of carbon nanotube-based composites. *Physica B*, 368:302–307, 2005.
[29] Xin-She Yang. Modelling heat transfer of carbon nanotubes. *Modelling Simul. Mater. Sci. Eng.*, 13:893–902, 2005.

Technologiegerechte Produktentwicklung von MEMS

Kai Hahn, Thilo Schmidt, Matthias Mielke, Rainer Brück
[vorname.nachname]@uni-siegen.de
Universität Siegen, Institut für Mikrosystemtechnik, Hölderlinstr. 3, 57068 Siegen

Dirk Ortloff
dirk.ortloff@process-relations.com
Process Relations GmbH, Emil-Figge Str.76-80, 44227 Dortmund

Kurzfassung

Die Produktentwicklung von MEMS unterscheidet sich wesentlich von der Entwicklung rein elektronischer Systeme. Neben der Integration von Systemkomponenten aus unterschiedlichen Ingenieurdisziplinen, stellt insbesondere die Auswahl einer geeigneten Fertigungstechnologie eine erhebliche Herausforderung dar. Die große Vielfalt an Materialien und Fertigungstechniken und die damit verbundenen technologischen Randbedingungen sind maßgeblich für Dauer und Kosten der Produktentwicklung verantwortlich. Immer kürzere Produktzyklen und die steigende Komplexität der technologischen Randbedingungen erfordern eine effizientere Entwurfsmethodik, die auch Fertigungstechnologie mit einbezieht. In diesem Artikel wird eine integrierte Entwurfsumgebung für den Entwurf anwendungsspezifischer Fertigungsprozesse vorgestellt, die Technologiemanagement und Entwurfswerkzeuge kombiniert.

Motivation

Die Produktentwicklung von MEMS (Micro-electromechanical Systems) unterscheidet sich wesentlich von der Entwicklung rein elektronischer Systeme. Neben der Integration von Systemkomponenten aus verschiedenen Ingenieurdisziplinen, stellt insbesondere die Auswahl einer geeigneten Fertigungstechnologie eine erhebliche Herausforderung dar. MEMS Fertigungstechnologie zeichnet sich durch eine unüberschaubar große Vielfalt an Materialien und Verfahren mit jeweils eigenen Anwendungsgebieten und technologischen Randbedingungen aus. Wechselwirkungen zwischen Technologie- und Systementwurf haben unmittelbar Einfluss auf alle Aspekte der MEMS Produktentwicklung und sind maßgeblich für Dauer und Kosten der Produktentwicklung verantwortlich. Immer kürzere Produktzyklen und die steigende Komplexität der technologischen Randbedingungen erfordern eine effizientere Entwurfsmethodik, die auch die Fertigungstechnologie mit einbezieht.

Die anwendungsspezifische Konfiguration der Fertigungsprozesstechnologie erfordert ein flexibles, auf den MEMS-Entwurf ausgerichtetes Prozessmanagement. Umgekehrt müssen technologische Randbedingungen (z.B. in Form von Parametern und Design Rules) dem Systementwurf zu Verfügung gestellt werden.

Von einer durchgehenden Entwurfsautomatisierung, wie sie in der Mikroelektronik selbstverständlich ist, kann beim MEMS-Entwurf nicht gesprochen werden. Die Entwicklung einer systematischen Entwurfsmethodik hat in der Mikroelektronik die Grundlage für die Entwicklung leistungsfähiger Entwurfswerkzeuge gelegt, die heute einen hohen Grad an Entwurfsautomatisierung realisieren. Auch für MEMS kann eine systematische Entwurfsmethodik die Grundlage einer leistungsfähigen Entwurfsautomatisierung liefern.

In diesem Artikel wird daher eine MEMS Entwurfsmethodik vorgestellt die einerseits den Systementwurf mit der Fertigungstechnologie verbindet und andererseits auch Anknüpfungspunkte für eine Entwurfsautomatisierung nach dem Muster der Mikroelektronik

bietet. Diese Anknüpfungspunkte werden anhand konkreter Beispiele mit kommerziell verfügbaren und im Rahmen von Forschungsprojekten entstandener Softwaremodule vorgestellt.

Besonderheiten der MEMS-Produktentwicklung

MEMS Produktentwicklung unterscheidet sich in einigen entscheidenden Punkten von der auf den ersten Blick ähnlichen Entwicklung planarer, rein elektronischer Schaltkreise (IC-Entwurf und-Fertigung). Mechanische Bestandteile von MEMS (also in der Regel Sensoren und Aktoren) werden über Massen und Volumina definiert und sind dementsprechend dreidimensional strukturiert. Die Höhe von Strukturen und ihre Materialeigenschaften lassen sich jedoch nur über die entsprechende Auslegung der Fertigungsprozesse beeinflussen. Damit kommt den Prozessen und ihrer applikationsspezifischen Modifikation bzw. Parametrisierung eine viel größere Bedeutung zu, als es beim IC-Entwurf der Fall ist. Diese starke und wechselseitige Technologieabhängigkeit ist als MEMS-Law (*one product, one process*)[13] in die Literatur eingegangen und ist wesentlicher Bestandteil des Brezelmodells (Abb. 2) [1] auf das im nächsten Kapitel noch näher eingegangen wird.

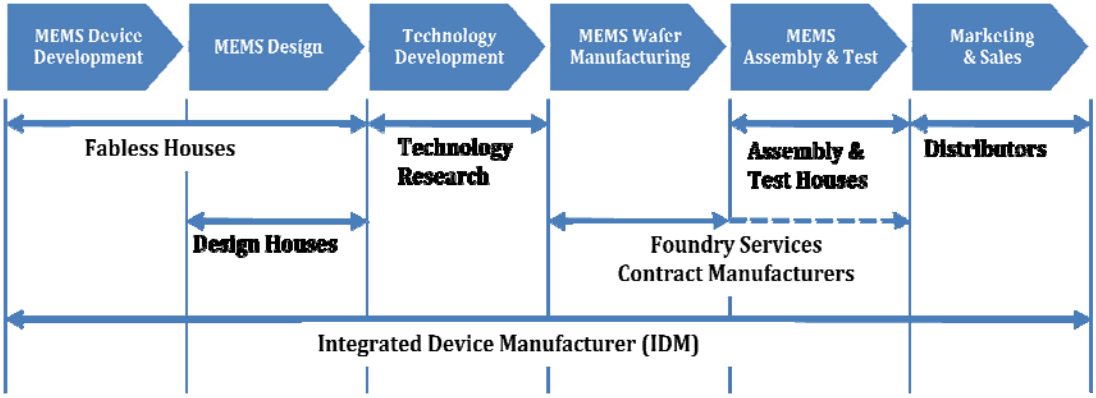


Abbildung 1 Wertschöpfungskette bei der MEMS-Entwicklung

Die enge Beziehung zwischen Systementwurf auf der einen und Prozessentwurf auf der anderen Seite legt eigentlich nahe, dass die komplette Produktentwicklung für MEMS, von der Konzeption bis zum Produktionsende hin, in einem Unternehmen stattfindet. Dies ist jedoch nur bei wenigen großen MEMS-Herstellern der Fall. Ein großer Teil des MEMS-Marktes ist geprägt von kleineren und mittleren Unternehmen die aufgrund ihrer Kompetenz nur einen Teil des Produktentwicklungszyklus‘ abdecken [13]. Abb. 2 illustriert die typisch Wertschöpfungskette bei der MEMS-Produktentwicklung. Neben den wenigen IDM (*Integrated Device Manufacturer*) auf dem Markt, gibt es viele Anbieter von Teilleistungen entlang der Wertschöpfungskette. So unterstützen beispielsweise kleinere Designhäuser ohne eigene Fertigung das sog. *Fabless Design*. Spezielle Fertigungsstätten sog. *Pure-Play-Foundries* bieten die MEMS Fertigung hingegen ohne Entwurfsunterstützung an. Wieder andere Unternehmen sind für Spezialaufgaben wie die Gehäusung der prozessierten Bauteile zuständig. Selbst bei Herstellern die große Teile der Produktentwicklung innerhalb der eigenen Unternehmens durchführen, werden diese Aufgaben häufig von unterschiedlichen Abteilungen, an verschiedenen Standorten ausgeführt.

Wie oben erwähnt sind die Schnittstellen zwischen Entwurf und Fertigung bei MEMS durch die wechselseitige Beeinflussung sehr viel komplexer und damit schwieriger zu handhaben, als dies z.B. beim klassischen Chipentwurf der Fall ist. Beim IC-Entwurf genügt es in der Regel die sog. *Design Rules*, also ein Satz geometrischer und elektrischer Entwurfsregeln einzuhalten, um die Fertigbarkeit der Schaltung zu garantieren. Solche formalisierten Regeln sind bereits in den PDKs (*Process Design Kit*) zu finden, die die Fertigung (die *Fab* oder *Foundry*) für die gängigen

EDA (*Electronic Design Automation*)-Entwurfswerkzeuge anbietet. Im Fall der MEMS-Produktentwicklung existiert bislang keine verbreitete Methodik, die auf die Besonderheiten des MEMS-Entwurfs Rücksicht nimmt. Für die großen Hersteller von integrierter IC-EDA-Software ist der MEMS-Bereich vergleichsweise eher ein Nischenmarkt und lohnt keine großen Investitionen. Die kleine Gruppe von Softwareanbietern (u.a. COVENTOR, PROCESS REALTIONS), die speziell die MEMS-Produktentwicklung unterstützen können nur jeweils Abschnitte der Wertschöpfungskette unterstützen.

Ein komplette *Product-Engineering*-Methodik und damit der Versuch einer möglichst lückenlosen softwaretechnischen Begleitung des MEMS-Entwurfs wird erstmals in einem laufenden EU-Projekt (CORONA CP-FP 213969-2) unternommen [3].

Der technologiegerechte MEMS-Entwurf

Aufgrund der Ähnlichkeit zur Mikroelektronik es sich auf den ersten Blick an, die dort verbreiteten Werkzeuge und Entwurfsmodelle, wie z.B. das Y-Modell nach Gaisky/Kuhn [4] auch für MEMS Entwurf einzusetzen.

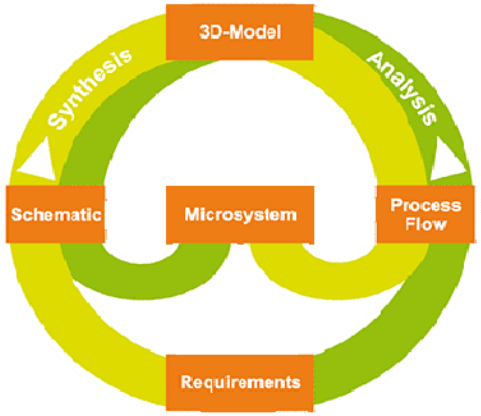


Abbildung 2 Brezelmodell für den MEMS Entwurf

Mikrosysteme zeichnen sich u.a. dadurch aus, dass ihre Funktionalität sich aus ihrer dreidimensionalen Strukturierung ergibt. Die bei MEMS erforderliche Strukturierung in der dritten Dimension bedeutet bei Silizium-Technologien folglich, anwendungsspezifisch eine jeweils eigene vertikale Schichtfolge zu entwerfen.

Damit ist es beim MEMS-Entwurf in der Regel nicht möglich, die beim IC-Entwurf so erfolgreiche Mead/Conway-Abstraktion [5] zu verwenden. Restriktionen der Fertigungstechnologie können nicht auf geometrische Design-Rules abstrahiert werden.

Im Unterschied zum IC Entwurf liegt die zentrale Herausforderung beim MEMS-Entwurf daher nicht beim Umgang mit eine sehr großen Anzahl von Entwurfsobjekten, sondern im „*Management of Technology*“.

Ein von den Autoren erstmals in [1] vorgestelltes Entwurfsmodell, dass die enge Verzahnung von Technologie- und Systementwurf berücksichtigt ist das in Abbildung 1 dargestellte Brezelmodell.

Das Brezelmodell unterscheidet fünf Zustände, die jeweils für einen bestimmten Satz von Entwurfsdokumenten stehen. Die einzelnen Zustände sind über zwei gegenläufige Entwurfspfade miteinander verbunden, was zu der namensgebenden Brezelform führt. Die beiden Pfade repräsentieren jeweils die synthetisierenden (Top-down) und die analysierenden (Bottom-up) Aktivitäten des MEMS-Entwurfs. Der Top-down-Pfad beginnt mit dem Entwurf einer abstrakten Verhaltens-/Strukturbeschreibung (*Schematic*). Auf Basis dieser Beschreibung entsteht im nächsten Schritt ein *3D-Modell* des Mikrosystems, auf dessen Grundlage dann eine geeignete Fertigungstechnologie (*Process*) ausgewählt bzw. neu entwickelt wird. Der Top-down-Pfad endet mit der Fertigung eines physischen Prototyps.

Der Bottom-up Ansatz startet ebenfalls mit den Produktanforderungen. Der erste Schritt ist eine Analyse ob die gewählte Technologie diese prinzipiell erfüllt (Stückzahlen, Ausbeute, Kosten etc.). Der nächste Analyseschritt ist bereits konkreter und dient der Überprüfung ob die Fertigungsanweisungen geeignet sind die gewünschten 3D-Strukturen zu erzeugen (z.B. durch

Prozesssimulation und Vergleich mit dem 3D-Modell). Die anschließende Vergleich des 3D-Modells mit der abstrakten Verhaltensbeschreibung kann ebenfalls mit Hilfe von Simulation (z.B. FEM) realisiert werden. Der letzte Schritt im Bottom-up-Pfad ist die Validierung des gefertigten Prototypen auf Basis des Verhaltensmodells.

Anhand des Brezel-Modells können ähnlich wie beim Y-Modell unterschiedliche Entwurfsabläufe definiert werden. Für Sonderfälle, wie z.B. eine fest vorgegebene Fertigungstechnologie kann auf einzelne Entwurfsschritte auch komplett verzichtet werden. Ein Entwurfsablauf der sich nur in der linken Hälfte des Brezelmodells bewegen würde entspräche einem klassischen IC-Entwurfsablauf. Vereinfachend kann man auch sagen, dass die linke Hälfte des Modells einen verhaltensnahen und die rechte Hälfte einen fertigungsnahen Entwurfsablauf modelliert. Durch den wechselseitigen Einsatz von synthetisierenden und analysierenden Entwurfsschritten kann ein Entwurfsablauf gemäß des des kybernetischen Modells von Rammig [6] realisiert werden.

Softwareunterstützung für den technologiegerechten Entwurf

Das Brezelmodell dient als Grundlage für einen technologierechten Entwurf, wie er im MEMS-Bereichnotwendig ist. Für einen effizienten Einsatz ist allerdings auch hier eine umfassende Entwurfsunterstützung notwendig.

Für den verhaltensnahen Entwurf existieren bereits integrierte Entwicklungsumgebungen, wie z.B. *CoventorWare* von *Coventor*, die sich an klassischen IC-Entwurfsabläufen orientieren. Der fertigungsnahe Bereich ist dagegen die Domäne von TCAD (Technology CAD) Werkzeugen, von Herstellern wie z.B. *SILVACO*, und Technologiemanagement-Werkzeugen, wie z.B. *XperiDesk* von *Process Relations.* Grundlegende Funktionalitäten von *XperiDesk* wurden von den Autoren im Rahmen des EU-Projekts PROMENADE (IST-507965) [14] in Zusammenarbeit mit *SILVACO* entwickelt. *Coventor* ist Partner im laufenden Projekt CORONA.

In Kombination ermöglichen CoventorWare und XperiDesk bereits eine recht umfassende Abdeckung des Brezelmodells [7]. Das *CoventorWare Architect* Modul und die damit verbundene Komponentbibliothek ermöglichen den effizienten Entwurf von MEMS-Verhaltensbeschreibungen (Schematic). Das *Designer* Modul kann verwendet werden, um aus der Verhaltensbeschreibung ein passendes 3D-Modell zu generieren. Das *Analyzer*-Modul ermöglicht eine Verhaltensanalyse des physikalischen Verhaltens des 3D-Modells. Im fertigungsnahen Bereich kann der Prozess-Emulator *SEMulator* für den Analyseschritt zwischen Prozess und 3D-Modell eingesetzt werden [12].

XperiDesk (Abb. 3) ist als *Process Development Execution System* (*PDES*) konzipiert und dient als solches primär zur Modellierung und Analyse der Fertigungsabläufe. Dazu gehört vor allem eine zentrale Wissensverwaltung, die die Vielfalt prozessrelevanter Daten und die komplexen Abhängigkeiten unter ihnen abbildet. Das System verfügt über Module zur regelbasierten Konsistenzprüfung [8], zum virtual Prototyping mit TCAD-Werkzeugen [9] und zur Planung und Durchführung von Laborexperimenten bzw. Prototypenfertigung [10]. Damit kann es die Syntheseschritte zwischen 3D-Modell und Prozess, sowie zwischen Prozess und Prototyp direkt unterstützen. Die Wissensbasis und die Konsistenzprüfung bieten Unterstützung für den Analyseschritt zwischen Anforderungen und Prozess und das *virtual Prototyping* unterstützt den Analyseschritt zwischen Prozess und 3D-Modell.

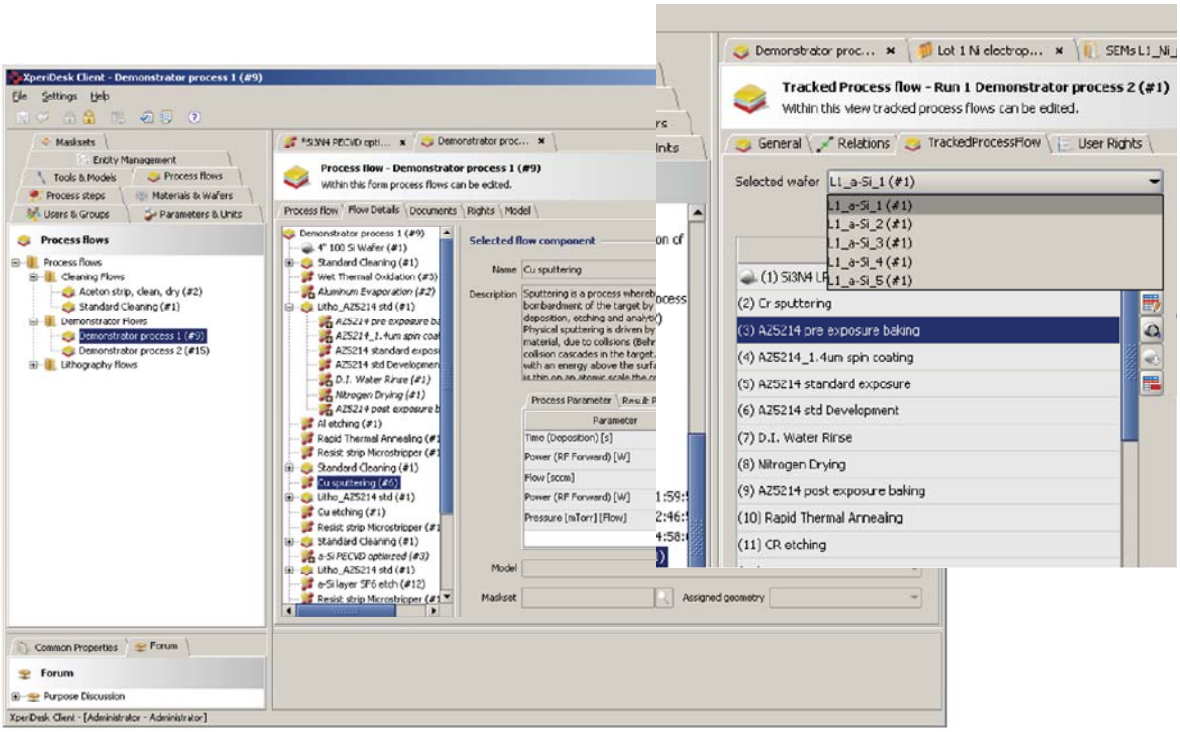


Abbildung 3 XperiDesk (Process Editor und Process Tracking)

Die Kerngebiete des technologiegerechten Entwurfs nach dem Brezelmodell sind daher bereits recht gut unterstützt. Ein laufendes EU-Projekt (CORONA) konzentriert sich bei der Softwareunterstützung daher primär auf zwei Schnittstellen im Produktentwicklungs-Flow:

Eine entscheidende Schnittstelle definiert den Datenaustausch zwischen dem 3D-Modell und dem Verhaltensmodell. Änderungen an dem Verhaltensmodell haben Auswirkungen auf das 3D-Modell und umgekehrt. Entsprechende Verhaltensmodelle werden meist aufwendig mit FEM Werkzeugen erzeugt und müssen für auch minimale Änderungen an Geometrie oder Material neu erstellt werden. Die Weiterentwicklung von Werkzeugen die eine jeweils automatische Anpassung von 3D- bzw. Verhaltensmodell ermöglichen ist ein Ziel der CORONA Projekts[11].

Die zweite relevante Schnittstelle ist die zwischen dem 3D-Modell und der Fertigungstechnologie. PDES-Programme wie XperiDesk unterstützen zwar den Ingenieur bei der Auswahl der Fertigungstechnologie. Der eigentliche Syntheseschritt vom 3D-Modell zum Prozess bleibt bisher jedochausschließlich der Kreativität und der Erfahrung des Ingenieurs überlassen und wird daher auch von Senturia als „Creative Art“[2] bezeichnet. Eine, zumindest teilweise, Entwurfsunterstützung dieses Vorgangs wird im Rahmen von CORONA angestrebt[8].

Fazit und zukünftiger Forschungsbedarf

Der technologiegerechte Entwurf von MEMS ergibt sich zwangsläufig aus dem großen Einfluss der Fertigungstechnologie. Da weder eine allgemeine Technologieplattform, wie CMOS in der Mikroelektronik, noch eine Abstraktion von Fertigungsconstraints nach dem Muster von Mead/Conway in Sicht ist, bleibt die unmittelbare Einbeziehung der Technologie in den Systementwurf alternativlos. Das Brezelmodell stellt den methodischen Rahmen für einen solchen Entwurfsablauf und gängige Softwarewerkzeuge wie CoventorWare und XperiDesk ermöglichen bereits heute eine recht gute Unterstützung dieses Modells.

Forschungsbedarf besteht allerdings weiterhin an der Schnittstelle, an der Systemverhalten und Fertigungstechnik aufeinander treffen. Das CORONA Projekt leistet dazu einen Beitrag, indem

es sowohl existierende Werkzeuge erweitert, als auch neue Konzepte wie die automatisierte Synthese von Fertigungsprozessen untersucht.

Literatur

[1] Wagener, A., Popp, J., Schmidt, T., Hahn, K., Brück, R., and Ortloff, D.: "*Environment for design and verication of mems fabrication processes*" in [Proceedings of the MST 2005], (2005). MST 2005.

[2] Senturia, S. D.: "*Microsystem Design*", Kluwer Academic Publishers (2001).

[3] Hahn, K., Schmidt, T., Mielke, M., Brück, R., and Ortloff, D., "*Micro and nano product engineering using data management for silicon-based fabrication process development*," in [Nanotechnology, 2009. IEEE-NANO 2009. 9th IEEE Conference on], 337{340 (July 2009).

[4] Gajski, D. D. and Kuhn, R. H.: "*New VLSI tools - Guest editor's introduction*," in [IEEE Computer], 16, IEEE (1983).

[5] Mead, C. and Conway, L.: "*Introduction to VLSI systems*", Addison-Wesley, Reading, Massachusetts (1980).

[6] Franz J. Rammig, F. J.: „*Systematischer Entwurf digitaler Systeme*", Vieweg+Teubner, Stuttgart (1989).

[7] Hahn, K., Schmidt, T., Mielke, M., Ortloff, D., Popp, J., Brück, R.: "*Comprehensive design and process flow configuration for micro and nano tech devices*," Smart Structures and NDE, SPIE (2010).

[8] Schmidt, T., Hahn, K., and Brück, R.: "*A knowledge based approach for MEMS fabrication process design automation*," in [33rd IEEE/CPMT International Electronics Manufacturing Technology Symposium], (Nov. 2008). IEMT 2008, Penang.

[9] Hahn, K., Wagener, A., Popp, J., and Brück, R.: "*Process Management and Design for MEMS and Microelectronics Technologies*," in [Proceedings of SPIE: Microelectronics: Design, Technology, and Packaging, Perth], 5274 (2003). SPIE Perth 2003.

[10] Ortloff, D., Popp, J., Wagener, A., and Hahn, K.: "*Breaking through the process development barriers*," in [Proceedings of the 12th International Conference on the Commercialization of Micro and Nano Systems, Puerto Vallarta], (2008). COMS 2008.

[11] Schröpfer, G.: "*From Physical to System-Level Design – Current and Next Generations of MEMS Design Tools"* (Invited), Proc. Micromechanics Europe 2009, Toulouse, France, 20-22 September 2009

[12] Hölzer, G., Knechtel, R., Breit, S. and Schröpfer, G.: "*3D Process Modeling - A Novel and Efficient Tool for MEMS Foundry Design Support"*, presented at SEMICON West 2009, San Jose, USA, July 2009, and also at SEMICON Europe 2009, Dresden, Germany, 5-7 October 2009

[13] Yole Developpement: "*MEMS: Global MEMS/Microsystems Markets and Opportunities 2008"*, Semi and Yole Developpement, 2008

[14] Ortloff, D., Popp J.,Schmidt, T., and Brück, R.: "*Process development support environment: a tool suite to engineer manufacturing sequences"*, Int. J. Computer Mater. Sci. and Surf. Eng., 2 (3/4):312-334, 2009

Modellierung und Simulation zeitschlitzbasierter Netzwerke mit TrueTime

Christian Roßberg, André Froß, Daniel Froß, Ulrich Heinkel

{christian.rossberg, andre.fross, daniel.fross, ulrich.heinkel}@etit.tu-chemnitz.de

Technische Universität Chemnitz

Professur Schaltkreis- und Systementwurf

Reichenhainer Strasse 70

09126 Chemnitz

Kurzfassung

In diesem Beitrag soll eine Erweiterung des TrueTime Netzwerk Simulators, welche die Simulation des Beacon-Enabled Modus des IEEE 802.15.4 unterstützt, gezeigt werden. Für energieeffiziente Netze ist der Beacon-Enabled Modus besonders von Vorteil. Dieser unterteilt den Kanal in eine aktive Phase und eine inaktive Phase. Ein anderer Vorteil des Modus ist, dass innerhalb eines Clusters Aktionen in Echtzeit ausgeführt werden können. TrueTime unterstützt eine Vielzahl von drahtgebundenen und drahtlosen Protokollen, jedoch nur den Non-Beacon-Enabled Modus des IEEE 802.15.4.

Einführung

Drahtlose lokale Netze gewinnen zunehmend an Bedeutung. In vielen Lebensbereichen sind sie nicht zu ersetzen. Sie werden in der Industrie- und Heimautomatisierung, PC-Peripherie und für vieles mehr eingesetzt. Vorteil gegenüber drahtgebundenen Netzen ist, dass sie in bereits bestehende Umgebungen eingebunden werden können. IEEE 802.15.4 [4] ist ein möglicher Standard für drahtlose Sensornetzwerke. In diesem werden unterschiedliche Buszugriffsverfahren implementiert. Einerseits ein Zugriff via CSMA/CA, andererseits lässt sich auch ein TDMA Verfahren anwenden. Letzteres ermöglicht eine energieeffiziente, echtzeitfähige Kommunikation. Dabei werden Daten verschiedener Sender in zugewiesenen Zeitschlitzen übertragen, wobei die Sender exklusiven Zugriff auf das Medium erhalten. Mit Hilfe der Simulation lassen sich Algorithmen entwickeln, die die Lebensdauer einzelner Knoten oder die Performance des Netzes abschätzen.

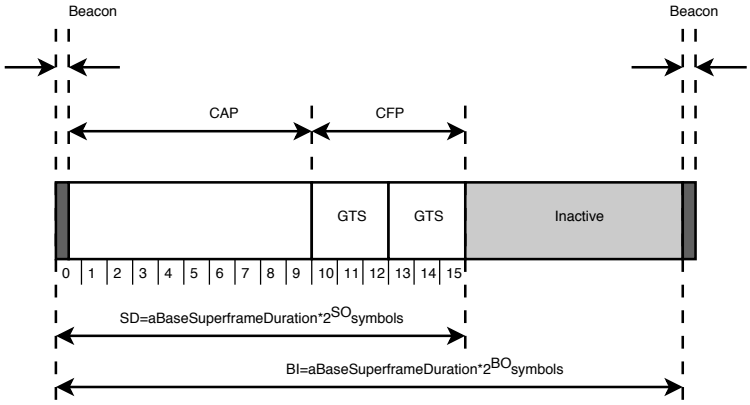


Abbildung 1: Rahmenstruktur des IEEE 802.15.4 Standards

Für drahtgebundene Netzwerke existiert eine Vielzahl von Simulationswerkzeugen, die bekanntesten sind NS2 [1] und OMNeT++ [2]. Beide erhielten erst im Laufe der Zeit Erweiterungen zur Unterstützung drahtloser Netzwerke. Einige Simulatoren wie z.B. TrueTime [3] erlauben eine Simulation von IEEE 802.15.4, allerdings meist nur im Non-Beacon-Enabled Modus. In dieser Veröffentlichung wird die Erweiterung des Simulators TrueTime um den Beacon-Enabled Modus, vgl. Abbildung 1, vorgestellt.

Für batteriebetriebene Geräte ist die erreichbare Lebensdauer ein entscheidendes Merkmal. Ein Simulink Block, welcher die Batterie und Funktionalität des Energiemanagements modelliert, ist bereits im Simulator vorhanden.

In [5] wurde bereits ein Ansatz für die Implementierung des Beacon-Enabled Modus von IEEE 802.15.4 vorgestellt. Dieser Ansatz ist allerdings auf eine Anwendung zugeschnitten und somit für die meisten Szenarien nicht zu gebrauchen.

Der Standard IEEE 802.15.4

Im Standard IEEE 802.15.4 definiert ist die physikalische Ebene und die Sicherungsschicht für Netze mit geringer Datenrate. Vorteile bei der Umsetzung des Standards sind die Einfachheit und die damit verbundenen geringen Kosten. Datenraten von bis zu 250 kb/s können in den Frequenzbändern 868 MHz, 915 MHz und 2,4 GHz erreicht werden. Innerhalb der Bänder werden Kanäle mittels FDMA aufgeteilt.

Es werden zwei Typen von Geräten unterschieden. Full Function Devices (FFD) beherrschen die komplette Funktionalität. Reduced Function Devices (RFD) umfassen nur einen kleinen Teil des Standards und sind deshalb lediglich als Endgerät nutzbar. RFD haben jedoch den Vorteil, dass auf Grund geringer Anforderungen ressourcenarme Hardware mit geringem Energiebedarf eingesetzt werden kann. Die Netzwerkteilnehmer können dabei als Stern, Cluster Tree oder Peer-to-Peer angeordnet sein.

Optional ist der Beacon-Enabled Modus verfügbar, welcher den Kanal in Zeitschlitze unterteilt, vgl. Abbildung 1. In der ersten Phase, der Contention Access Period (CAP), haben alle Knoten Zugriff auf das Medium. Danach folgt eine zweite Phase, die Contention Free Period (CFP), in welcher die Zeitschlitze einzelnen Knoten zugewiesen sind. Das ermöglicht einerseits eine Echtzeitkommunikation und garantiert andererseits eine Störungsfreiheit durch andere Knoten des Netzwerkes. Die Zeitschlitze werden deshalb Guaranteed Time Slot (GTS) bezeichnet, wobei ein GTS auch aus mehreren Zeitschlitzen gebildet werden kann.

Ebenso ist der Beacon-Enabled Modus für Netze mit geringem Energieverbrauch interessant. Da Aktivitäten im Kanal nur während der aktiven Phase erlaubt sind und der Beacon in festen Zeitschritten wiederkehrt, können die Knoten für diese Zeitdauer in den Ruhezustand versetzt werden. Ausgesandt wird der Beacon vom Koordinator und enthält Informationen über die Aufbau des Superframes.

TrueTime

TrueTime wird an der Universität Lund (Schweden) entwickelt. Der Simulator besteht aus Simulink Blöcken und MatLab C-Funktionen. Eine Vielzahl von Protokollen ist bereits implementiert. Vorteil von TrueTime ist, dass das Netzwerk und die Anwendung in einer Umgebung simuliert werden können. Der Network Block und Wireless Network Block repräsentieren den Kanal für drahtgebundene oder drahtlose Netzwerke. Der Kernel Block modelliert die Knoten des Netzwerkes.

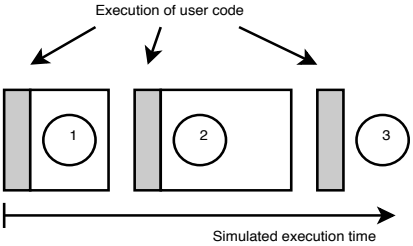


Abbildung 2: Ausführung des Anwenderprogramms

Während der Initialisierung der Blöcke werden die Anzahl der Ein- und Ausgänge, Scheduler, Tasks und Interrupt Handler definiert.

Der TrueTime Simulatorkern übernimmt die Funktionen der ersten und zweiten Schicht aus dem ISO/OSI 7-Schichten Modell. Das Anwenderprogramm stellt die höheren Schichten dar. Der Code ist dabei in Segmente unterteilt, vgl. Abbildung 2, die zeitlich nacheinander abgearbeitet werden.

Mit MatLab/Simulink lässt sich eine Vielzahl von Anwendungen entwerfen. TrueTime ermöglicht es, die Anwendung und das Netzwerk in einer Umgebung zu modellieren und simulieren.

Erweiterung von TrueTime

Für die Erweiterung von TrueTime musste zuerst die Funktionsweise analysiert werden. Die Rahmenstruktur, vgl. Abbildung 1, und die Variablen zur deren Verwaltung mussten angelegt werden. Hierfür wurde die Funktion *ttsetnetworkparameter* angepasst, welche die Verwaltung des Rahmens übernimmt. Die Generierung der Beacon Nachricht wurde dem Simulator hinzugefügt und geschieht automatisch, somit kann der Nutzer das Anwenderprogramm frei gestalten. Allerdings lässt sich über das Anwenderprogramm die Rahmenstruktur verwalten. Da verschiedene Buszugriffsverfahren verwendet werden, besteht die Senderoutine aus drei Teilen. Im ersten Teil verschickt der Simulatorkern den Beacon automatisch, anschließend werden die Daten der CAP verschickt und im letzten Teil die Daten der CFP. Hierfür sind die Variablen der Rahmenstruktur verantwortlich.

Des Weiteren wurden Funktionen für die Auswertung der Lebensdauer implementiert. Dadurch lässt sich während der Simulation bereits eine Vorhersage treffen, wie lange ein Netzwerkteilnehmer mit Energie versorgt werden kann. Dazu wurde auch ein neuer Block in der Simulink Block-Bibliothek angelegt, dessen Ausgang liefert die berechnete Lebensdauer in Abhängigkeit des Akkumulators und des Strombedarfs im Format [Jahr, Monat, Tag, Stunde, Minute, Sekunde].

Nachrichten werden in einer Queue abgelegt. Beim Ausführen des Simulatorkerns werden alle Nachrichten, die sich in dieser Queue befinden, versandt. Daher muss der Simulatorkern mit den Funktionen des Nutzers synchron arbeiten, was durch die Funktion *ttsynctimeslot* gewährleistet ist. Diese Funktion lässt die Funktion des Nutzers bis zum (n-1)-ten Zeitschlitz schlafen. Dies stellt sicher, dass Daten erst kurz vor dem Senden in der Queue abgelegt werden, vgl. Abbildung 3, und nicht zu einem falschen Zeitpunkt verschickt werden.

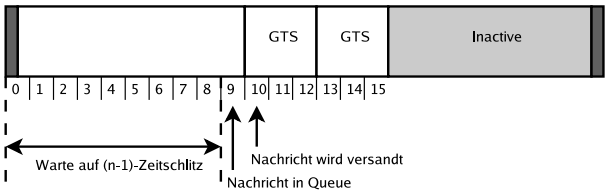


Abbildung 3: Funktionsweise ttsynctimeslot, Nachricht soll im Zeitschlitz 10 verschickt werden

Problematisch war die Anpassung des Simulatorkerns. Der Simulator reagiert nur, wenn Nachrichten in der Queue abgelegt wurden. Daher mussten Ereignisse geschaffen werden, die die Funktionalität des Simulatorkerns zu jedem Zeitschlitz ausführen, parallel zu den Nachrichten in der Queue.

Eine Vielzahl weiterer Funktionen wurden modifiziert bzw. neu implementiert, um den Beacon-Enabled Modus von IEEE 802.15.4 zu unterstützen und dem Nutzer die Programmierung seiner Anwendung angenehm zu gestalten.

Simulation und Ergebnisse

Als Fallbeispiel dient die Implementierung einer einfachen Regelung. Auf einem Knoten läuft der Regelalgorithmus und auf einem weiterem ein Mess- und Steuerprozess. Beide Knoten

besitzen zugewiesene GTS. Auf den Knoten können weiterhin unterschiedliche Schlafstrategien genutzt und untereinander verglichen werden.

Die minimale Abtastzeit für das System ist durch die minimale Länge des Superframes ohne inaktive Periode spezifiziert und beträgt 15,36 ms. Auf Grund der langsam veränderlichen Messgröße und der geringen Datenmenge wurde SO=1 und BO=3 festgelegt. Somit ergibt sich eine inaktive Periode, in welcher die Knoten in den Ruhezustand versetzt werden können. Die Sendeleistung wurde als konstant angenommen.

Schlafstrategie	Lebensdauer
Alle Komponenten	5d, 5h, 21m
Controller Power Down/ADC an	4d, 6h, 23m
ADC Power Down/Controller an	3d, 6h, 6m
Ohne Power Down	2d, 18h, 33m

Tabelle 1: Lebensdauer der Knoten mit unterschiedlichen Schlafstrategien (d, h, m entspricht Tagen, Stunden, Minuten)

Es hat sich gezeigt, dass bereits durch Abschalten nicht benötigter Komponenten eine erhebliche Verlängerung der Lebensdauer der Knoten erzielt werden kann. In Tabelle 1 sind die Ergebnisse der Simulation dargestellt. Die Abschätzung der Lebensdauer der Knoten erfolgt mit den Blöcken der TrueTime Bibliothek. Auf jedem Knoten ist eine andere Schlafstrategie implementiert. Es ist zu erwarten, dass sich die maximale Lebenszeit bei steigender Anzahl aktiver Komponenten verkürzt. Dies bestätigte die Simulation, der Faktor zwischen kürzester und längster Lebensdauer liegt bei rund 2. Im Beispiel wurde ein Akku mit 3,6 V Nennspannung und 1 Ah Kapazität sowie folgende Stromstärken angenommen:

- Controller: 12 mA
- ADC: 2,7 mA
- TX: 21,175 mA
- RX: 13,3 mA
- Power Down Controller: 0,015 mA

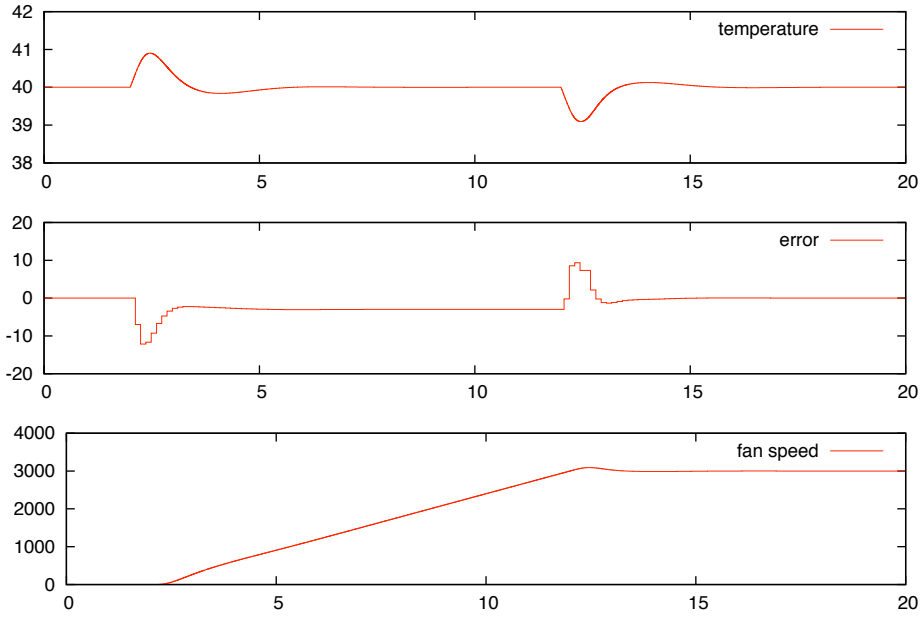


Abbildung 4: Simulationsergebnisse

Dem Regelungs- und dem Prozessknoten sind jeweils GTS Zeitschlitze zugeordnet. Der Regelungsknoten ist gleichzeitig der Koordinator des Netzwerkes. Der Prozessknoten sendet seine Daten in Zeitschlitz 13 und 14, welche zum einem ersten GTS Zeitschlitz zusammengefasst sind. Der Regelungsknoten sendet seine Daten in einem zweiten GTS. Abbildung 4 zeigt die Simulationsergebnisse. Dargestellt sind die gemessene Temperatur, der Reglungsfehler und die Geschwindigkeit des Lüfters.

Zusammenfassung

TrueTime, ein Simulationswerkzeug für drahtgebundene und drahtlose Netzwerke, wurde vorgestellt. Der Standard IEEE 802.15.4 und dessen Buszugriffsverfahren wurden eingeführt. Eine Simulatorerweiterung für TrueTime zur Unterstützung des Beacon-Enabled Modus von IEEE 802.15.4 wurde gezeigt. Der Simulatorkern unterstützt dabei die Schichten 1 und 2 des ISO/OSI 7-Schichten Modells. Darüberliegende Schichten müssen vom Anwender bereitgestellt werden.

Mittels der präsentierten Simulatorerweiterung können Algorithmen für das Routing und die Exploration eines Beacon-Enabled IEEE 802.15.4 Netzwerkes implementiert, simuliert und verglichen werden. Weitere Protokolle lassen sich einfach hinzufügen. Hierfür benötigt man Kenntnisse im Bereich C++ und MatLab/Simulink.

TrueTime unterstützt mobile Knoten sowie eine Anpassung der Sendeleistung zur Laufzeit. Die Leistungsfähigkeit der Visualisierung ist nicht sehr hoch und soll in weiteren Arbeiten gesteigert werden. Mit dem Batterie-Block lässt sich die Lebensdauer einzelner Knoten abschätzen und energieeffiziente Algorithmen entwickeln.

Bisher nicht vorgesehen ist ein Austausch von Bestätigungsnachrichten zwischen Knoten. Ebenso ist keine Laufzeitabschätzung für Funknachrichten implementiert.

Problematisch für die Simulation ist die Initialisierung des Netzwerkes. Zur Unterscheidung von Netzwerkclustern benutzt TrueTime IDs, welche über die gesamte Simulationszeit konstant bleiben. Ein dynamischer Wechsel der Clusterzugehörigkeit von Netzwerkknoten ist nicht modellierbar, wodurch die Simulation großer Netze erheblich erschwert wird.

Danksagung

Diese Arbeit wurde durchgeführt im Rahmen des InnoProfile Projekts „Generalisierte Plattform zur Sensordatenverarbeitung“ gefördert durch das Bundesministerium für Bildung und Forschung unter der Projektnummer 03IP505.

Literatur

[1] http://www.isi.edu/nsnam/ns.

[2] http://www.omnetpp.org.

[3] http://www.control.lth.se/truetime.

[4] Part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans), amendment 1: Add alternate phys. Technical report, IEEE Computer Society, IEEE, 3 Park Avenue, New York, NY 10016-5997, USA, 2007.

[5] N. Boughanmi, Y. Song, and E. Rondeau. Wireless networked control system using ieee 802.15.4 with gts. In *2nd Junior Researcher Workshop on Real-Time Computing - JRWRTC2008*, pages 21–25, 2008.

EDADB - eine Infrastruktur zur Dokumentation und Wiederverwendung von Schaltungstopologien

Volker Boos
volker.boos@imms.de
Institut für Mikroelektronik und Mechatronik Systeme GmbH
Ilmenau

Kurzfassung

Der Entwurf mikroelektronischer Schaltungen findet heute weitgehend mit kommerziellen Tools (z.B. Cadence, Mentor, Tanner) statt. Einige Aufgaben, wie die Erstellung hochwertiger Dokumentationen oder die Wiederverwendung von Schaltungen in anderen Systemumgebungen können bisher nur mit großem Aufwand erfüllt werden. Im IMMS wurde eine Infrastruktur entwickelt, die den Designer bei diesen Aufgaben durch Tools mit moderner Benutzoberfläche unterstützt.

Die Infrastruktur besteht aus einer XML Datenstruktur für Schaltungstopologien, in Cadence integrierte Import- und Exportmodule, einem durch Plugins erweiterbaren Grafikprogramm (EDADB-Manager), sowie einer Datenbasis mit Web-Interface.

Mit dieser Infrastruktur wird ein Designflow für die Wiederverwendung analoger IP-Blöcke in anderen System- und Technologieumgebungen bereitgestellt. Die Arbeiten sind Teil der Aufgabe „Methoden für die Nutzung der technologieunabhängigen IP-Datenbasis" im Projekt RapidMPSoC [1], das vom BMBF gefördert wird[1].

Einleitung

Der Entwurf analoger integrierter Schaltungen erfordert heute noch einen hohen Aufwand an Ingenieurarbeit [2]. Die vollautomatische Synthese aus einer Verhaltensbeschreibung ist bisher nur für Spezialfälle realisiert und hat für den allgemeinen Fall erst akademisches Niveau erreicht. In der Praxis zeigt sich aber, dass viele Schaltungstopologien immer wieder unter veränderten Systemumgebungen (andere Spezifikation, modernere Technologie) neu gezeichnet, dimensioniert und optimiert werden. Ziel der hier vorgestellten Arbeiten ist ein industrietaugliches, erweiterbares Framework auf Basis der Wiederverwendung bewährter Schaltungstopologien, das durch Integration in den Designflow eine effektive Nutzung ermöglicht.

Die vorhandenen Schaltungen müssen zunächst dokumentiert und für die Datenbasis aufbereitet werden. Mit diesen Informationen wird die Schaltung zu einem wiederverwendbaren IP Block. Ein Web-Interface mit Suchfunktionen, grafischen Vorschaubildern und Downloadfunktion ermöglicht einen sehr flexiblen Einsatz in Rechnernetzen.

Die Anpassung an die neue Systemumgebung erfolgt in zwei Schritten. Als erstes müssen die im IP-Block vorhandenen Bauelemente durch solche aus der technologischen Bibliothek (PDK) ersetzt werden. Danach erfolgt eine erste Dimensionierung, die für alle Bauelemente den richtigen Arbeitspunkt garantiert (Initialdimensionierung). Diese Arbeitsschritte werden durch eine grafische Benutzeroberfläche und effektive Algorithmen unterstützt.

Die so erhaltene simulierbare Schaltung kann danach mit den bekannten Designtools weiter bearbeitet werden.

[1] Förderkennzeichen 01 M 3085

Systemübersicht

Abb. 1 zeigt einen Überblick des gesamten Systems. Aus der Cadence DFII Entwurfsumgebung wird die Schaltungstopologie in das EDADB XML Format exportiert. Mit dem EDADB-Manager werden Anpassungen vorgenommen, sowie Dokumentationen erstellt. Die Schaltungen zur Wiederverwendung werden als IP-Block in die Topologiedatenbank eingepflegt.

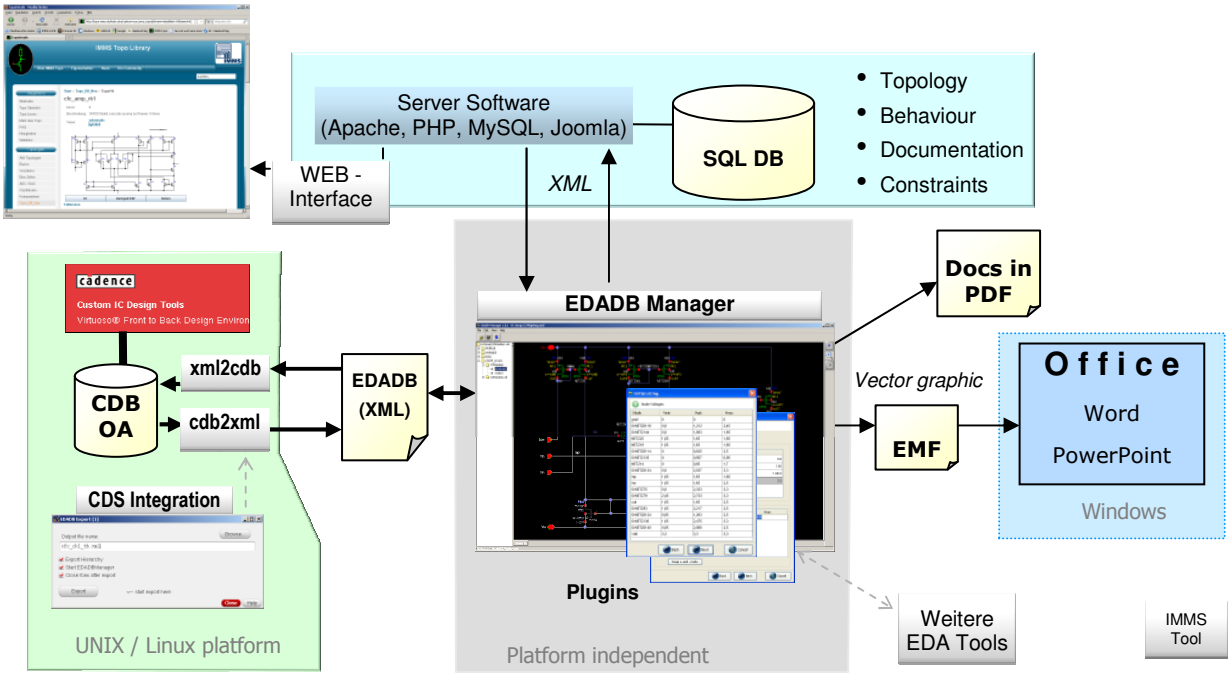


Abb. 1 Systemübersicht

Mit Hilfe des Web-Interfaces können IP-Blöcke zur Wiederverwendung anhand verschiedener Kriterien gesucht und im Browser dargestellt werden. Nach dem Download werden diese mit dem EDADB Manager für die neue Technologie- und Systemumgebung angepasst. Dabei werden für die Bauelemente passende Devices aus dem PDK ausgewählt und eine erste Dimensionierung vorgenommen. Anschließend wird die Schaltung in Cadence DFII importiert. So entsteht ein vollständiger Flow für die Wiederverwendung von analogen IP-Böcken.

Datenstruktur für Schaltungstopologien

Schaltungstopologien, die mit den kommerziellen Tools erstellt werden, enthalten neben der sichtbaren Liniengrafik auch Informationen zum Generieren der Netzliste, Parameter und Kommentare. Ein hierarchischer Aufbau erlaubt eine übersichtliche Darstellung und eine effiziente Datenspeicherung.

Die Datenstruktur EDADB enthält all diese Informationen im XML Format. Das ermöglicht eine einfache Weiterverarbeitung, da fast alle Programmiersprachen XML unterstützen.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE EDADB SYSTEM "edadb.dtd">
<EDADB>
  <LIBRARY name="xh035_circuits" >
    <CELL name="C2NOpAmp" >
      <CELLVIEW name="schematic" type="schematic" uperuu="160" uuname="inch">
       <LP purpose="drawing" layer="wire">
        <ELLIPSE bbox="405 -515 415 -505" > <points>405 -515 415 -505 </points> </ELLIPSE>
        <LINE bbox="380 -510 380 -470"  net="Vss"> <points>380 -510 380 -470 </points> </LINE>
       </LP>
       <INSTHEADER lib="PRIMLIB" cell="nmos" view="symbol">
        <INSTANCE name="MN5" x="30" y="-460" orientation="R0" numinst="1">
            <INSTTERM termnet="D" connet="NETZ064"/>
            . . .
```

Abb. 2 Auszug aus einer Topologiebeschreibung im EDADB XML Format

Integration in Cadence DFII

Die Akzeptanz neuer Tools wird stark erhöht, wenn diese in den Designflow eingebunden werden. Der in Skill geschriebene Modul **`edbexport.il`** erzeugt im File-Menü (Version 6) bzw. im Design Menü (Version 5) des Virtuoso Schaltungseditors den Menüeintrag „Export EDADB XML …“, mit dem der Dialog für den Export geöffnet wird (Abb. 3). Auf Wunsch wird sofort der EDADB-Manager geöffnet, der die exportierte Schaltung grafisch darstellt.

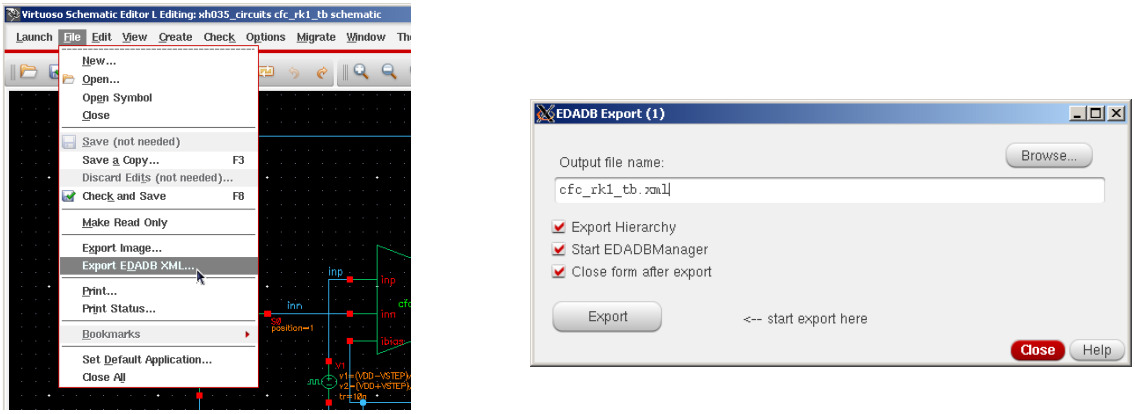


Abb. 3: Integration in die Cadence 6 Umgebung

Der EDADB Manager

Der EDADB Manager ist ein erweiterbares Programm zur Visualisierung und Bearbeitung der Schaltungstopologie. Er ist in Java geschrieben und läuft so auf allen Plattformen, die Java 1.6 unterstützen. Er ist kein vollwertiger Schaltplaneditor, sondern stellt spezielle Funktionen für die Technologieanpassung und für die Erstellung von Dokumentationen zur Verfügung .

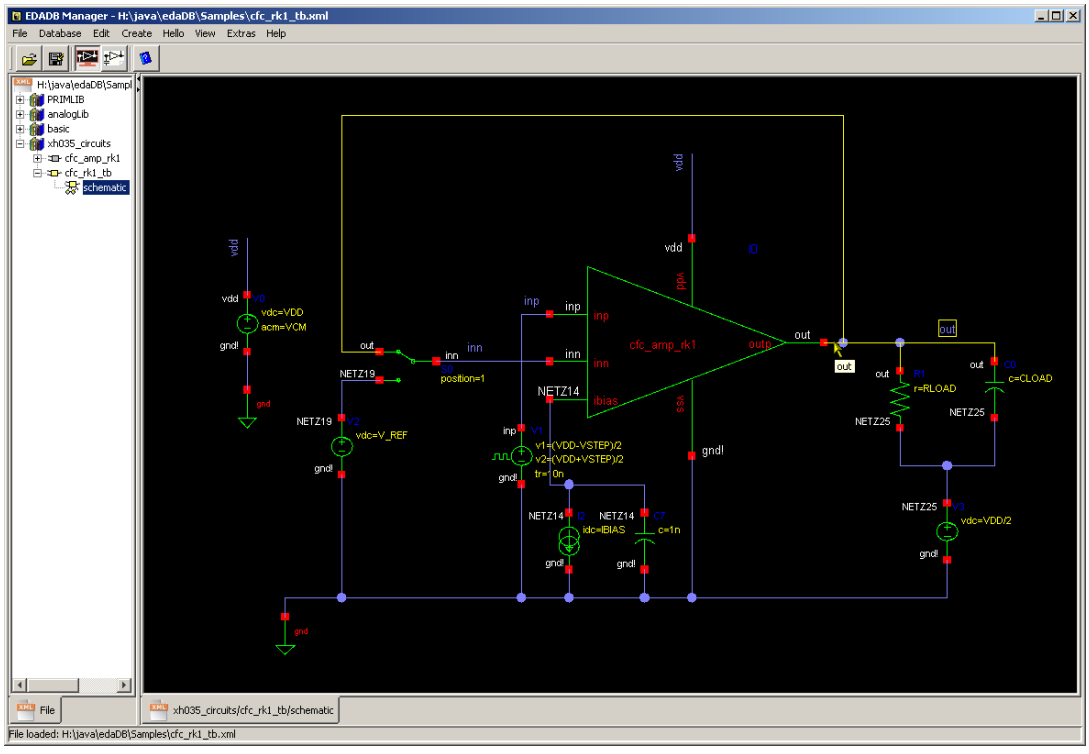


Abb. 4 : Screenshot des EDADB Managers (Mauszeiger über dem Netz „out“)

Die Funktionalität umfasst das Austauschen von Symbolen, das Ändern des Farbschemas, Ein- und Ausblenden bestimmter Informationen, sowie das Einfügen und Bearbeiten kommentierender Elemente (Texte, Rechtecke, Linien). Die angezeigte Grafik lässt sich dann in Vektor- und Rasterformate (EPS, PS, PDF, EMF, SVG, PNG) konvertieren und in Textverarbeitungs- und Präsentationsprogramme importieren. Ein Kopieren über die Zwischenablage ist ebenfalls möglich.

Beim Bewegen der Maus wird das darunterliegende Schaltungselement oder das entsprechende Netz grafisch hervorgehoben und der Name angezeigt (Abb. 4). Durch Anklicken werden die Elemente markiert. Durch Rechtsklick erscheint ein Kontextmenü, das Funktionen zum Anzeigen der Eigenschaften und zum Bewegen in der Hierarchie bereitstellt.

Eine Besonderheit ist die Erstellung einer verlinkten PDF-Dokumentation für eine kompletten Schaltungshierarchie. Durch gezieltes Ein- und Ausblenden von Informationen können so anspruchsvolle Dokumentationen für Kunden und Mitarbeiter erstellt werden.

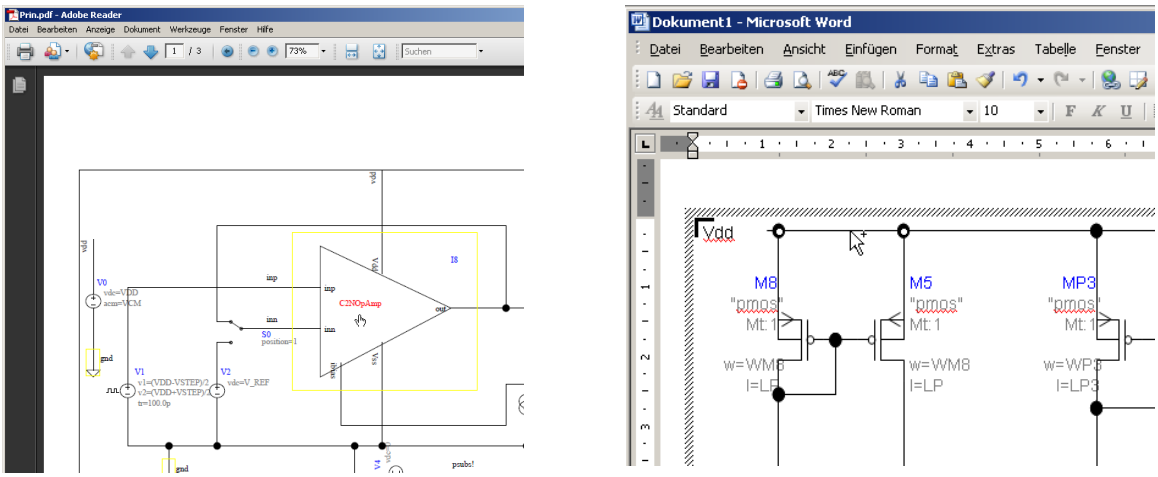


Abb. 5 Verlinktes PDF Dokument (links) und eingefügte Vektorgrafik in MS Word (rechts)

Erweiterung des EDADB Managers mit Plug-Ins

Die Wiederverwendung und Synthese von analogen IP-Blöcken sind heute noch ein großes Forschungsgebiet. Um für zukünftige Anforderungen flexibel zu bleiben, wurde der EDADB-Manager mit einem Plugin Konzept ausgestattet. Ein Plugin ist ein Java-Archiv (jar), das beim Start des EDADB-Managers automatisch geladen wird. Es muss ein vorgegebenes Interface besitzen und kann damit auf die Daten und die Grafik der geladenen Schaltung zugreifen.

Als Beispiel soll hier das Plugin „InitSize“, ein Assistent zur Initialdimensionierung von analogen Schaltungsblöcken, vorgestellt werden. Grundlage ist der in [3] vorgestellte Algorithmus, der zunächst aus den Vorgaben für die Arbeitspunkte der Bauelemente einen Graph mit Spannungs- und Strombedingungen erstellt. Daraus werden die für den DC Arbeitspunkt optimalen Spannungen und Ströme berechnet. Die Dimensionierung erhält man durch Simulation des entsprechenden Transistormodells mit den vorgegebenen Spannungen und Strömen.

Die grafische Oberfläche besteht aus mehreren Dialogseiten, auf denen schrittweise die für den Algorithmus notwendigen Informationen eingegeben werden. Bei jedem Schritt erfolgt eine Kontrolle der Daten. Grafisches Feedback, z.B. durch Hervorheben des ausgewählten Netzes oder Instanz in der Schaltplangrafik, vereinfacht die Zuordnung zwischen den Namen in den Dialogfeldern und der Schaltplangrafik.

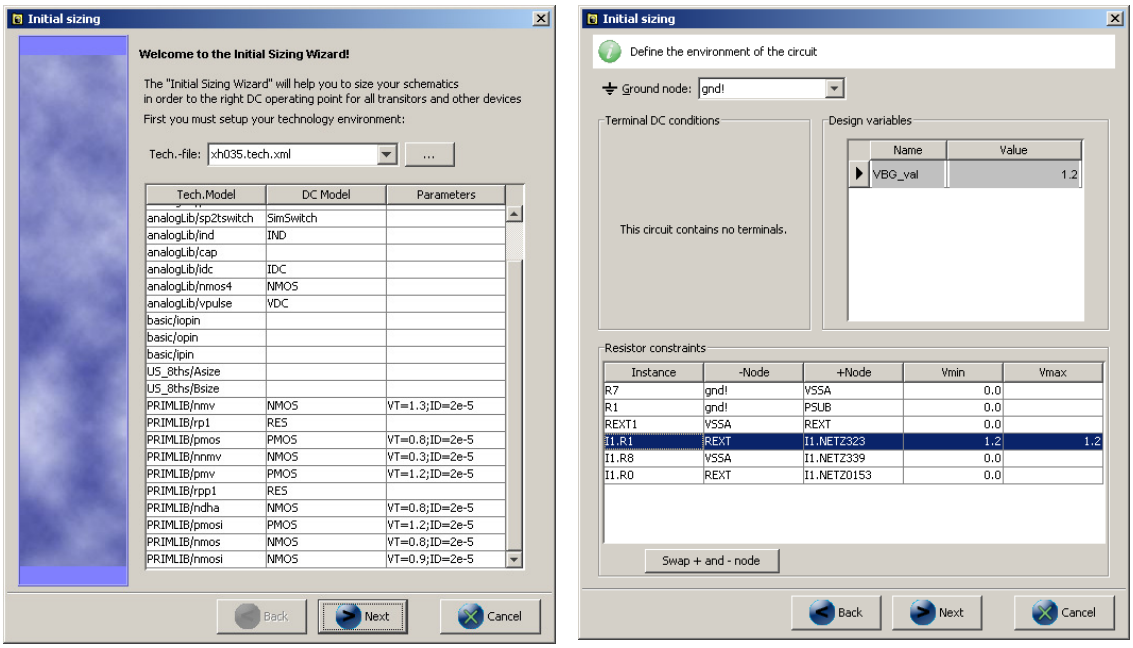


Abb. 6 : Die ersten zwei Dialoge des Plugins zur Initialdimensionierung

Ein weiteres Plugin dient zur Erstellung einer Netzliste für die symbolische Analyse mit dem Tool „Analog Insydes“. Mit der erstellten Netzliste ist eine erste Analyse und Bewertung der Schaltung in der neuen Systemumgebung möglich.

Datenbasis und Web-Interface

Die Serverumgebung besteht aus Web-Server, SQL-Datenbank und Content-Management System. Hier kommt ausschließlich freie Software (Linux, Apache, MySQL, PHP, Joomla) zum Einsatz.

Die Datenbasis enthält Tabellen und Relationen, die der bekannten Hierarchie Library – Cell – Cellview entsprechen. Jeder Cellview enthält ein großes Textfeld zur Speicherung der Daten. Das sind bei Schaltungen und Symbolen grafische Daten im EDADB-XML Format oder beliebige andere Texte, wie z.B. Verhaltensbeschreibungen . Ein Upload der Daten ist sowohl mit dem Web-Interface als auch im EDADB-Manager für autorisierte Benutzer möglich.

Das Web-Interface der Topologiedatenbasis verwendet das in PHP programmierte Content Management System „Joomla“. Für die speziellen Erfordernisse der Schaltungstopologien wurde ein Erweiterungsmodul entwickelt, das die grafische Benutzeroberfläche, die Generierung eines Vorschaubilds und Downloadfunktionen enthält. Zur Pflege der Topologiebibliothek stehen im Backend Funktionen für das Upload von Topologien und Verwaltungsaufgaben zur Verfügung. Nach internen Tests wurde ein Prototyp der Topologie-Datenbasis als Demo auf einem öffentlichen Webspace der TU Ilmenau installiert [6] .

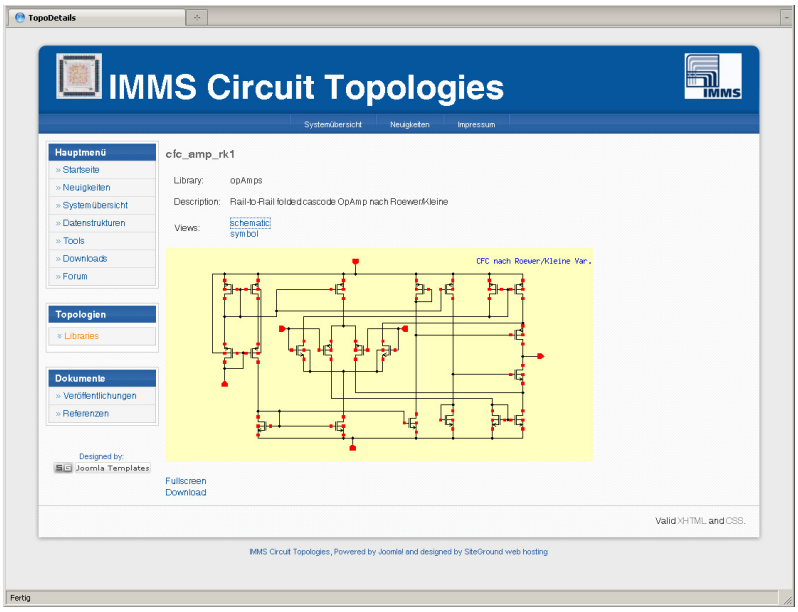


Abb. 7 : Web-Interface der Topologiedatenbank

Zusammenfassung und Ausblick

Die in diesem Beitrag vorgestellte Infrastruktur zur Wiederverwendung analoger IP-Blöcke lässt sich durch die Integration und die Schnittstellen zu kommerziellen Tools gut in den Designflow integrieren. Grafische Benutzeroberflächen und interaktive Unterstützung tragen wesentlich zur Akzeptanz durch den Anwender bei. Das Plugin-Konzept erlaubt eine einfache, flexible Erweiterung.

Die Datenbasis mit dem Web-Interface kann sowohl auf einem internen Server zum Aufbau von Firmen Know-How, als auch auf extern zugänglichen Web-Servern für Lehre und Forschung verwendet werden.

Das XML-Format für die Schaltungsdaten ist leicht zu generieren und zu verarbeiten, deshalb eignet es sich auch besonders gut für Synthesetools. Als Textformat kann es leicht übertragen und kontrolliert werden. Zusammen mit dem Plugin-Konzept ist das vorgestellte System auch als Framework für die Analogsynthese gut geeignet.

Literatur

[1] Homepage des Projekts RapidMPSoC, http://www.edacentrum.de/rapidmpsoc

[2] R. A. Rutenbar: Design Automation for Analog: The Next Generation of Tool Challenges, ICCAD 2006

[3] V. Boos: Graphentheoretischer Ansatz zur Initialdimensionierung analoger Schaltungen, DASS, 2007

[4] V. Boos: Strategien zur Initialdimensionierung von analogen Schaltungen, ANALOG ’08, Siegen, 2008

[5] V. Boos, R. Sommer, E. Hennig: Gaining Insight into Analog Circuit Behavior – Tools and Extensions to Cadence DFII, CDNLive! EMEA 2009 Conference in Munich, Germany

[6] V. Boos : IMMS Circuit Topologies, www.tu-ilmenau.de/imms_topo

Modulares Framework für die synchronisierte Erfassung, Verarbeitung und Aufbereitung heterogener Sensornetzdaten

Matthias Vodel, René Bergelt, Matthias Glockner, Wolfram Hardt

vodel | berre | hardt @cs.tu-chemnitz.de

TU Chemnitz, Professur Technische Informatik

Kurzfassung

Die vorgestellte Arbeit präsentiert ein geschlossenes Konzept für die synchronisierte Erfassung, Verarbeitung und Aufbereitung beliebiger Sensor-Informationen. Es ist nun möglich, heterogene Sensornetzwerke sowie dedizierte, autarke Messsysteme zeitlich zu koordinieren und entsprechend in Relation zu setzen. Auf Basis von XML erfolgt die ganzheitliche Beschreibung des Monitoring-Szenarios und die Einordnung der einzelnen Datensätze. Die Informationen können nun in beliebigen Ausgabeformaten anwendungsspezifisch definiert und visualisiert werden**.** Zusätzlich ermöglichen Mechanismen zur gezielten Messwert-Vorverarbeitung und -Filterung eine Senkung des benötigten Datenvolumens.

Die Funktionalität des vorgestellten Frameworks wird am Beispiel eines geschlossenen Test-Systems aus dem Automotive-Bereich evaluiert. Dabei wird die vorhandende Sensorik eines Kfz-Bordnetzes mit einem zusätzlichen Netzwerk aus Sensorknoten verfeinert. Die korrelierten Daten werden anschließend für die Visualisierung mittels Google Earth, jBEAM sowie FlexPro aufbereitet.

Einleitung

Im Rahmen aktueller Forschungen auf dem Gebiet drahtloser Sensornetzwerke existieren zahlreiche Projekte, welche mit unterschiedlichsten, proprietären Hardwareplattformen arbeiten. Derzeit hat jedes Messszenario meist anwendungsspezifische und voneinander unabhängige Verarbeitungsprozesse für die Datenaufnahme, -speicherung und -auswertung.

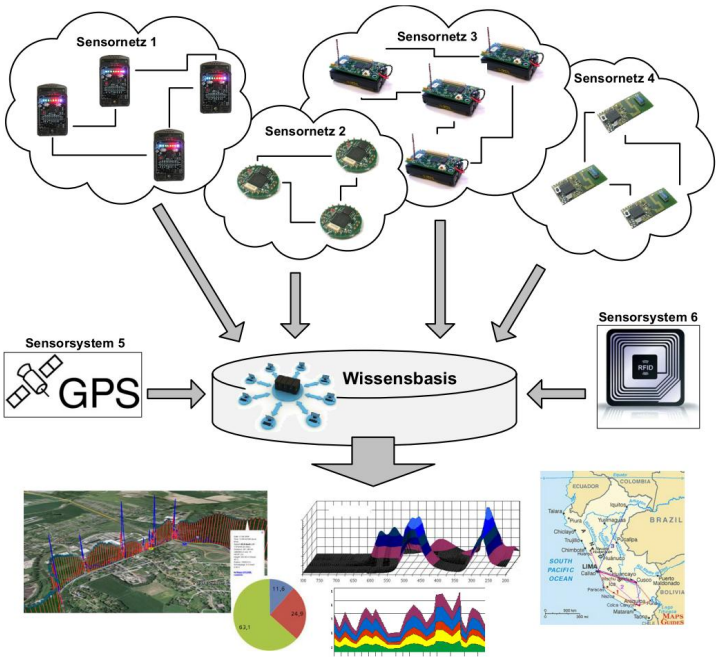


Abbildung 1: Integration unterschiedlicher Sensornetze durch ein synchronisiertes Verarbeitungsframework

Zwischen den autarken Messsystemen existieren dabei keinerlei Synchronisationsmöglichkeiten. Infolgedessen ist eine zielgerichtete, detaillierte Aufbereitung der Informationen innerhalb einer gemeinsamen Datenbasis nicht ohne Weiteres möglich. Die Datensätze der jeweiligen Sensorplattformen verfügen dazu über keinen einheitlichen Primärschlüssel, wodurch sie nicht eindeutig in Relation zueinander gesetzt werden können.

Stand der Technik

Es existieren bereits diverse Systeme zur Messdatenaufnahme und Messdatenauswertung am Markt, die jedoch meist funktionalen oder systembedingten Einschränkungen unterliegen. Viele Anbieter von Sensoren liefern so beispielsweise speziell angepasste

Auslesesoftware, die auf Geräte einer bestimmten Produktreihe beschränkt ist. Andere Sensorsysteme überlassen das konkrete Auslesen der Daten ganzheitlich dem Anwender und bieten keinerlei Unterstützung bei der Aufbereitung.

Bei produktgebundenen Auswertungstools muss der Hersteller die Messdatenaufnahme systembedingt so allgemein wie möglich gestalten. Für die automatisierte bzw. teilautomatisierte Aufbereitung der Messreihen müssen spezielle Anwendungen hinzugezogen werden, um den Nutzer anwendungsspezifisch zu unterstützen. Bekannte Vertreter dieser Art von Messanalysesoftware sind LabView von National Instruments [1], jBEAM der Firma AMS [2] sowie die Datenanalyse-Software FlexPro [3]. Die Anwendungen ermöglichen sowohl die Auswertung von importierten Offline-Daten als auch die Live-Analyse von Messdaten, die unmittelbar in das System einfließen. jBEAM und LabView sind hierbei plattformunabhängig einsetzbar. Sie unterstützen eine Vielzahl von Datenformaten und Schnittstellen. Durch den modularen Aufbau und durch die Implementierung des ASAM[1]-Standards [4] erlaubt jBEAM darüber hinaus eine einfache Erweiterbarkeit durch eigene Komponenten. FlexPro bietet eine Vielzahl zusätzlicher Module für visuell ansprechende Präsentationen der Messergebnisse.

Der entscheidende Nachteil aller vorgestellten Anwendungen resultiert aus der Tatsache, dass diese aufgrund der Funktionsvielfalt und den damit einhergehenden Systemvoraussetzungen nicht für die reine Datenaufnahme in ressourcenbeschränkten Hardware-Umgebungen geeignet sind. Sie kommen meist in einer nachfolgenden Instanz zum Einsatz, um die bereits aufgenommen Daten zu verarbeiten. Energieoptimierte und platzsparende Systeme, die nur Daten sammeln, aber nicht auswerten sollen, können diese Anwendungen nicht nutzen.

Es fehlt folglich an Hilfsmitteln und Standards, um beliebige Sensorkonfigurationen und -informationen generisch in eine gemeinsame Verarbeitungsinstanz zu leiten, welche die Daten anschließend frei definierbar analysieren und auswerten kann. Dies führt dazu, dass nahezu jedes Sensorprojekt unterschiedliche, meist proprietäre Softwarelösungen nutzt, die zueinander inkompatibel sind. Änderungen in den Konfigurationen des Messszenarios sind infolgedessen nur mit zusätzlichem Zeit- und Ressourcensaufwand realisierbar.

Konzept

An dieser Stelle setzt das vorgestellte Konzept an und bietet dem Nutzer nun ein modulares Softwareframework für die synchronisierte Erfassung beliebiger Sensorinformationen. Um die Funktionalität zu gewährleisten, müssen zunächst zentrale Anforderungen definiert werden.

Anforderungen

Primäres Ziel ist es, eine Verarbeitungsumgebung zu schaffen, die flexibel auf Veränderungen in der Konfiguration und in den Analyseanforderungen reagieren kann. Im vorgestellten Framework wird daher eine generische, flexible Messwert-Aufnahme und -verarbeitung speziell im Umfeld ressourcenbeschränkter Systeme umgesetzt.

Der Verarbeitungsprozess muss dabei in zwei räumlich, zeitlich und plattformtechnisch getrennte Phasen unterteilbar sein. Auf der einen Seite arbeiten alle Komponenten zur Datenaufnahme. Alle für die Datenaufbereitung relevanten Module laufen unabhängig in einer zweiten Phase ab. Dies steht im Gegensatz zu vorhandenen Lösungen, in denen diese klare Trennung entweder nicht existiert oder verschiedene Konfigurationen gänzlich unterschiedliche Methoden verwenden, die nicht ohne weiteres zusammengeführt werden können. Der Fokus des Frameworks liegt somit auf einem standardisierten Transport und einer individuell konfigurierten Synchronisation beliebiger Messdaten aus mehreren unabhängigen Sensorikkomponenten. Änderungen im Auswertungsprozess haben folglich keinerlei Einfluss auf die Komponenten der Datenaufnahme. Dies ist bei schwer zugänglichen oder aufwändig zu konfigurierenden Messaufbauten ein erheblicher Vorteil.

---

[1]ASAM…… *Association for Standardisation of Automation and Measuring Systems* [4]

Des Weiteren sollen alle Abläufe, auf die der Nutzer über eine graphische Oberfläche Zugriff hat, auch automatisiert bzw. teilautomatisiert gestartet und überwacht werden können. Auch hier hebt sich das entwickelte Framework von den bereits genannten Anwendungen ab, bei denen dies aufgrund des Anwendungsfokus nicht konsequent umgesetzt wurde. Um aber kontinuierlich und unbeaufsichtigt Sensordaten aufzunehmen und auszuwerten, ist diese Funktionalität häufig zwingend erforderlich. Alle zentralen Anforderungen können wie folgt aufgeschlüsselt werden:

- Synchronisation unterschiedlicher, autarker Sensorsysteme
- Modulare Erweiterbarkeit / einfache Modifizierbarkeit
- Nutzung von XML (*Extendible Markup Language*) als Austauschformat anstatt proprietärer Formate
- Auswertung der Daten aus einer Datenbank, aber auch direkt aus Logdateien
- Grafische Konfigurationsoberfläche
- Automatisierte Auswertungsmechanismen

Das Framework soll hierbei als Vermittler zwischen anwendungsspezifischen Komponenten fungieren, jedoch selbst keine anwendungsspezifische Logik besitzen, um universell einsetzbar zu sein.

Struktur

Durch den modularen Aufbau des Sensornet-Frameworks wird dieses vollständig unabhängig von der gewählten Sensorkonfiguration. Die Umgebung bedient sich dabei der sogenannten Stundenglasarchitektur (Ende-zu-Ende-Prinzip) [5], die eine maximale Interoperabilität zwischen verschiedenen Komponenten ermöglicht. Dies bedeutet, dass eine Vielzahl unterschiedlicher Sensoren und dazugehöriger Auslesekomponenten existieren, die durch den *SensorController* auf ein spezifiziertes, allgemeingültiges Format abgebildet werden (siehe Abb. 2). Ausgabeseitig stellt eine *SensorLogReader* Komponente verschiedenste Auswertekomponenten zur Verfügung. Somit liegt sowohl auf der Eingangs- als auch auf der Ausgangsseite eine große Diversität vor, die durch die Uniformität im Mittelteil verbunden wird.

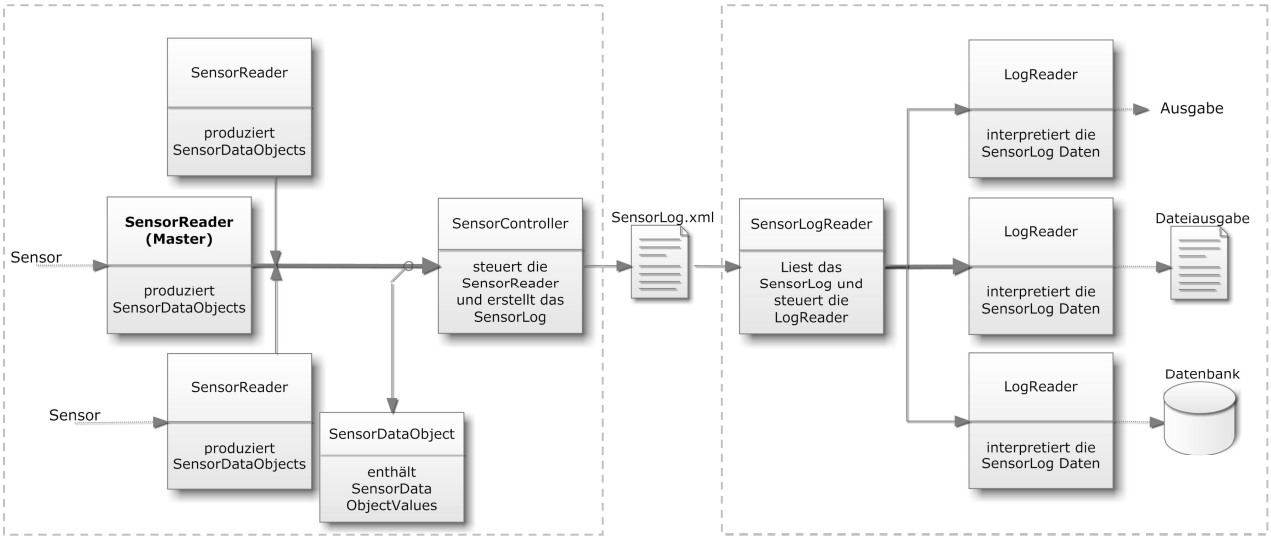


Abbildung 2 Struktur des Sensornet-Frameworks
Deutlich zu erkennen ist die Zweiteilung in Datenaufnahme (links) und Datenauswertung (rechts)

Das Resultat ist eine klare Zweiteilung des Frameworks in Prozeduren zur Datenaufnahme und Datenauswertung. Die XML-Repräsentation des gesamten Messszenarios bildet dabei das Bindeglied des Frameworks und ist eine Grundvoraussetzung für den Austausch und die Bearbeitung beliebiger Sensorkonfigurationen.

Vorhandene Sensorkonfigurationen können nun einfach durch das Hinzufügen beziehungsweise Entfernen von Komponenten verändert werden ohne das der Datenfluss des Systems dadurch verändert oder angepasst werden muss.

Prinzipiell ist es auch möglich, einen direkten Datenfluss von den Sensordaten-Aufnahmekomponenten zu den Sensordaten-Auswertekomponenten ohne die XML Repräsentation zu realisieren. Diese Laufzeitbeschleunigung verhindert jedoch eine vollständige und ersichtliche Beschreibung des Messszenarios. Erst die ganzheitliche XML-Beschreibung ermöglicht an dieser Stelle eine Wiederverwendung der Messdaten.

Implementierung

Das Framework wurde in Java implementiert, um eine Plattformunabhängigkeit zu gewährleisten. Prinzipiell können auch Komponenten unterschiedlicher Programmiersprachen zusammenarbeiten, wenn der implementierte *SensorController* dies unterstützt. Dazu wurde eine zentrale *Core-Library* geschaffen, die die gesamte Framework-Logik enthält und zur Modulentwicklung genutzt werden kann. Die Kommunikation des *SensorControllers* und des *SensorLogReaders* mit den jeweiligen Modulen erfolgt dabei über definierte Interfaces, die so allgemein und generisch wie möglich gestaltet wurden. Die erhöhte Flexibilität vereinfacht die Einbindung von Drittanbieter-Modulen und gewährleistet eine Abwärtskompatibilität im Zuge von Weiterentwicklungen [7]. Konfigurationen und Einstellungen werden ebenfalls im XML-Format übermittelt. Dies begünstigt eine spätere Erweiterbarkeit für anwendungsspezifische Problemstellungen.

Ablauf

Beim Start des *SensorControllers* lädt dieser eine Konfigurationsdatei, die Informationen zum aktuellen Projekt sowie zur Struktur der *SensorReader*-Komponenten enthält. Der *SensorController* initialisiert nun alle genutzten Sensorkomponenten und startet die Datenaufnahme. Die einzelnen *SensorReader*-Module arbeiten dabei threadbasiert und somit parallel. Mit dem Empfang eines Datensatzes an einem *SensorReader* wird von diesem ein *SensorDataObject* erzeugt, welches zuvor durch das Framework definiert wurde. Zusätzliche Informationen, wie bspw. physikalische Einheiten oder Messgrößen, können dabei ebenfalls hinzugefügt werden. Das erstellte Objekt wird nun dem *SensorController* gemeldet, der es weiterverarbeitet, in Relation zu den Daten anderer *SensorReaders* setzt und XML-konform abspeichert.

Beim Lesen der XML Logdatei durch den *SensorLogReader* werden die enthaltenen Datensätze wiederum in definierte Framework-Objekte umgewandelt und den Auswerte-Komponenten zur Verfügung gestellt. Ein zentraler Vorteil des Systems ist dabei die Eigenschaft, dass die Datenquelle für die Auswertung nicht zwingend auf eine lokale Logdatei beschränkt wird, sondern beispielsweise auch durch Daten aus einer Datenbank repräsentiert werden kann.

Anwendung

Das vorgestellte Framework wurde im Zuge mehrerer Sensornetz-Szenarien an der TU Chemnitz entwickelt. Im Folgenden soll deshalb der Einsatz anhand eines konkreten Anwendungsbeispiels aus dem Bereich *Automotive Sensoring and Monitoring* erläutert werden [8]. Im Rahmen mehrerer Langzeitmessungen mit Serienfahrzeugen wurde die Bordelektronik mit zusätzlicher Sensorik erweitert, die bordnetzunabhängige Messdaten bezüglich Temperatur, Helligkeit, Beschleunigung sowie Magnetfeldstärke zur Verfügung stellt. Zusätzlich wurde ein hochpräziser GPS-Sensor im Wagen verbaut, der kontinuierlich Positiondaten, Geschwindigkeitsangaben sowie aktuelle Höheninformationen des Wagens liefert.

Mit Hilfe der gewonnenen Sensorinformationen sollen Rückschlüsse auf bestimmte Fahrsituationen geschlossen werden, die anhand der GPS-Daten später verifiziert werden können. Auch Verschleißanalysen stehen im Fokus der Betrachtungen. Durch die Korrelation aller Sensorinformationen mit den fahrzeuginternen Borddaten entstehen auf diese Weise detaillierte

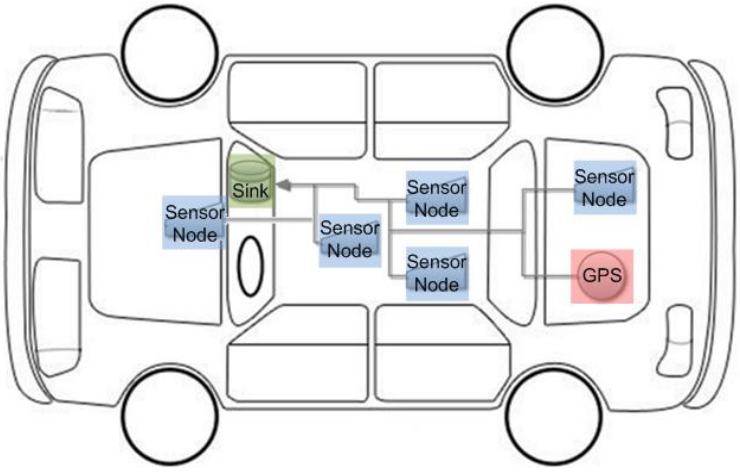


Abbildung 3 Versuchsaufbau - Die Daten der Sensorboards und des GPS-Sensors laufen in der Datensenke des Versuchsfahrzeuges zusammen.

Fahrerprofile, welche wiederrum der Optimierung der Fahrzeugcharakteristik dienen. Abbildung 3 verdeutlicht schematisch den Aufbau des Messszenarios. Neben dem GPS-Empfänger befinden sich im Fahrzeug fünf Sensorboards an strategischen Positionen. Alle Informationen fließen in einer gemeinsamen Datenbasis zusammen.

Für das Auslesen der Sensorboards wurde ein herstellerspezifisches *SensorReader*-Modul implementiert, welches über den seriellen Port ankommende Daten klassifiziert und als abstrahierte Datenobjekte an den Controller übergibt. Gleiches gilt für ein separat implementiertes Modul für den GPS-Sensor, welches alle NMEA[2] standard-konformen Geräte [6] unterstützt. Zur Synchronisation der Daten wird während der Versuchsinitialisierung ein ausgewählter *SensorReader* als *Master* definiert. Da der GPS-Sensor neben den reinen Positionsdaten auch ein exaktes Zeitsignal liefert, repräsentiert dieser im vorgestellten Szenario den globalen Taktgeber *(Master)* des Systems.

Das Framework synchronisiert dabei nicht zwingend auf Basis eines Zeitsignals. Der Taktgeber des Systems ist durch den Anwender frei definierbar. Dies ist besonders für die Integration autarker Sensoriksysteme ohne eigene Scheduling-Mechanismen interessant.

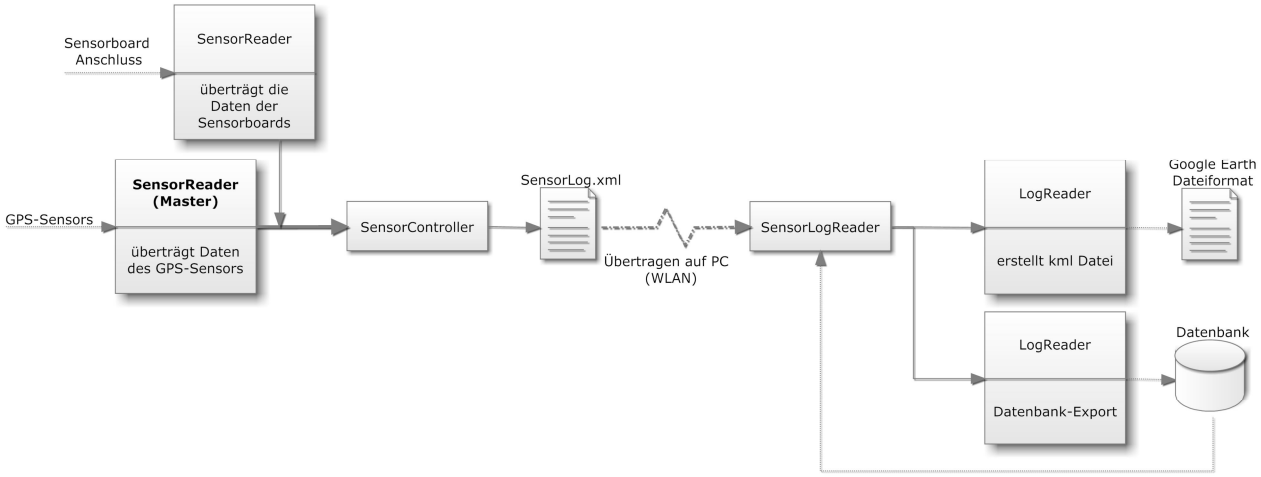


Abbildung 4 Anwendung des Frameworks auf das beschriebene Szenario.

Auswertung

Für die Auswertung der ermittelten Daten wurden zwei Auswertekomponenten geschaffen. Dabei handelt es sich einerseits um ein Exportmodul, welches die Sensorwerte für die Einspeisung in die zentrale Datenbank vorbereitet und anschließend überträgt und andererseits um eine zweite Komponente, die die Informationen für Google Earth aufbereitet. Dabei dienen die in die Datenbank übertragenen Datensätze nicht nur der Archivierung, sondern können ihrerseits wieder in einen *SensorLogReader* eingespeist und somit durch andere Auswertemodule weiterverarbeitet werden. Die Auswertemodule müssen dabei nicht angepasst werden.

---

[2]NMEA 0183…… *National Marine Electronics Association* 0183 - Standard für die Kommunkation zwischen Navigationsgeräten untereinander und mit Computern

Für Google Earth können die Messkurven bspw. zusätzlich als dreidimensionale Höhenverläufe entlang der gefahrenen Strecke visualisiert werden. Der umgesetzte Gesamtprozess zur Datenerfassung, -verarbeitung und -aufbereitung wurde dazu in Abbildung 4 dargestellt. Alle geschaffenen Komponenten sind generisch und folglich für eine Vielzahl möglicher Anwendungsszenarien und Hardwarekonfigurationen einsetzbar.

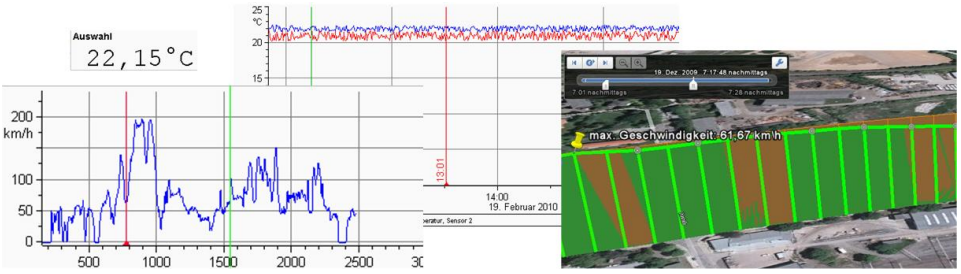


Abbildung 5 beispielhafte Messwertdarstellungen (Google Earth / FlexPro)

Für weiterführende, detaillierte Auswertungen der Messergebnisse ist nun auch der Datenimport in komplexe Analysetools wie jBeam oder FlexPro möglich und sinnvoll (siehe Abbildung 5). Über ein entwickeltes CSV-Exportmodul (*Comma Separated Values*) lassen sich die Daten problemlos konvertieren.

Zusammenfassung und Ausblick

Die vorgestellte Arbeit beschreibt die Umsetzung einer umfassenden Datenerfassungsumgebung, die eine flexible Basis für weiterführende wissenschaftliche Arbeiten im Bereich der Sensordatenaggregation und Datenfusion bietet. Im Zuge der Messwertanalyse können nun komplexe Algorithmen zum Einsatz kommen, welche auf einer gemeinsamen und umfänglichen Datenbasis arbeiten. Im vorgestellten Szenario sind somit gezielt Rückschlüsse auf bestimmte Fahrerprofile sowie auf modellspezifische Merkmale der Fahrzeugcharakteristik möglich.

Das Framework erfüllt hierbei die Anforderungen der Flexibilität, der ressourcensparenden Ausführung und der vollen Automatisierbarkeit, die durch andere Softwarelösungen nicht oder nur teilweise realisiert werden können. Es kommt bei einer Vielzahl unterschiedlicher Sensordatenprojekte an der Technischen Universität Chemnitz zum Einsatz. Im Rahmen der Messdatenaufnahme entsteht so ein standardisierter Prozess, der neben der Datenaufbereitung auch wesentliche Vorteil für die Datenarchivierung bietet. Die Umsetzung neuer Projekte kann deutlich vereinfacht und beschleunigt werden. Das Ergebnis stellt eine projektübergreifende, umfängliche Wissensbasis dar, auf die einfach und universell zugegriffen werden kann.

Literaturverzeichnis

[1] National Instruments, LabView http://www.ni.com/labview/

[2] AMS GmbH, jBEAM http://www.jbeam.de/german/produkte/jbeam.html

[3] FlexPro http://www.weisang.com/

[4] ASAM-Spezifikation http://www.asam.net/

[5] Salter, J.; Reed, D.; Clark, D.: *End-to-End Arguments in System Design, ACM Transactions on Computer Systems 2*, http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf

[6] Kaplan, E.; Hegarty, C.: *Understanding GPS: Principles and Applications*. Artech House Publishers; second edition (November 30, 2005)

[7] Brown, A.: *Component-Based Software Engineering*. Wiley-IEEE Computer Society Pr; (September 30, 1996)

[8] Vodel, M.; Lippmann, M.; Caspar, M.; Hardt, W.: *A Capable, High-Level Scheduling Concept for Application-Specific Wireless Sensor Networks.* 4th International Symposium on Information Technology, June 2010

Eingebaute Selbstreparatur zur Kompensation von Produktions- und Alterungsfehlern

Tobias Koal und Heinrich Theodor Vierhaus
Lehrstuhl Technische Informatik
BTU Cottbus, PF 101344, Deutschland
{tkoal, htv}@informatik.tu-cottbus.de

Kurzfassung

Durch stetig sinkende Produktionsprozesse erhöht sich die Anfälligkeit von Schaltungen gegenüber Herstellungsfehlern und Fehlern im Feld. In dieser Veröffentlichung wird ein auf Software basierendes Verfahren vorgestellt, welches sich zur Kompensation von permanenten Fehlern nach der Produktion als auch im Feld eignet. Durch einen zweistufigen Reparaturprozess können permanente Fehler in bestimmten Teilen des Prozessors, durch das Zuschalten zusätzlich implementierter Redundanz, kompensiert werden. Mit diesem Ansatz wird eine Steigerung der Zuverlässigkeit des Gesamtsystems erreicht. Weiterhin ist eine Erhöhung der Produktionsausbeute möglich.

1 Einleitung

Sinkende Strukturgrößen bei der Fertigung von integrierten Schaltungen führen unweigerlich zu Produktionsprozessen, welche anfälliger für Herstellungsfehler sind [1]. Gleichzeitig sind diese kleineren Strukturen auf den ICs höheren physikalischen Belastungen ausgesetzt, welche die Zuverlässigkeit für lange Betriebszeiten gefährden [2]. Die sich daraus ergebende Aufgabe besteht darin, sowohl Test- als auch Reparaturmethoden zu entwickeln, welche beide Fehlertypen berücksichtigen. Maßnahmen zur Fehlertoleranz, welche universell einsetzbar sind können sowohl die Ausbeute bei der Produktion als auch die Zuverlässigkeit im Feld erhöhen.

Methoden zur Steigerung der Ausbeute findet man hauptsächlich im Bereich der Speicherbausteine. Hier werden zusätzliche Reihen bzw. Spalten an Speicherelementen vorgehalten, welche fehlerhafte Elemente ersetzten können. Diese Verfahren kommen meist direkt nach der Produktion zum Einsatz und können durch das Durchbrennen von Sicherungen permanent gemacht werden [3, 4]. Ebenfalls sind diese Reparaturansätze weiterentwickelt worden, um im Feld zur Steigerung der Zuverlässigkeit Anwendung zu finden [5, 6]. Auch für die Anteile von Logik auf den ICs gibt es Ansätze zur Steigerung der Ausbeute. Hier können Methoden der Fehlertoleranz, welche ursprünglich für transiente Fehler entwickelt worden sind, beispielsweise durch Codes aber auch durch Mehrheitsentscheidung, einen positiven Einfluss auf die Ausbeute haben [7, 8].

Das bestehende Problem definiert sich nun darin, dass die Maßnahmen zur Kompensation von Produktionsfehlern für Logik, welche ursprünglich zur Vermeidung von transienten Fehlern entworfen wurden, einen negativen Einfluss auf die Zuverlässigkeit der Systeme im Bezug auf Alterungsfehler haben können. Denn die zusätzliche Logik für Codes oder für Mehrheitsentscheidungen unterliegt ebenfalls massiv dem Einfluss von Alterungseffekten.

Ein weiteres Problem besteht darin, dass Produktionstests wie der sogenannte "Burn-in" Test das Auftreten von Alterungsfehlern beschleunigen kann und damit als Testmethode zunehmend ungeeignet sind.

Wir stellen eine Test und Reparatur Methodik für die Anteile von Logikbaugruppen auf ICs vor, die sich sowohl für Produktionsfehler als auch für Fehler im Feld eignet. Dieser Ansatz basiert auf einem Software-basierten Test [9, 10], welcher zur Lokalisation von Fehlern dient.

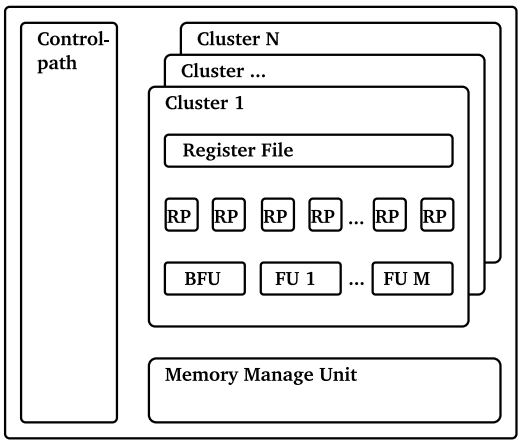


Abbildung 1: Originale generische VLIW Architektur

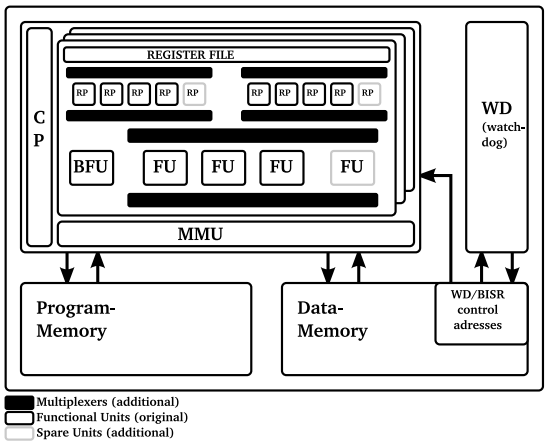


Abbildung 2: Architektur mit zuschaltbarer Redundanz und Watchdog

Grundsätzlich werden dazu Befehle des Befehlssatzes verwendet, um im Vorfeld generierte Testmuster an interne Schaltungsteile anzulegen und deren Antwort auszuwerten. Zuschaltbare Redundanz für homogene Baugruppen wird ebenfalls durch Software verwaltet und kann fehlerbehaftete Komponenten ersetzen.

Diese Veröffentlichung ist wie folgt untergliedert: In Kapitel 2 wird eine VLIW Architektur vorgestellt, für die dieses Verfahren umgesetzt wurde. Kapitel 3 beschreibt die auf Software basierende Test und Reparatur Prozedur. Abschließend folgen die erzielten Ergebnisse sowie eine Zusammenfassung.

2 Architektur & Redundanz

Der in Abbildung 1 dargestellte VLIW Prozessor wird hier als Beispiel verwendet, um die Anwendbarkeit dieser Methodik zu zeigen. Der gesamte Prozessor besteht aus n- Clustern mit jeweils m- funktionalen Einheiten (FU), welche jeweils 2 Leseports (RP) besitzen. Diese Lesports werden benötigt, um auf die jeweilige Registerbank zugreifen zu können. Jeweils eine funktionale Einheit (BFU) pro Cluster existiert, welche zusätzliche Addressberechnungen ausführen kann. Gesteuert wird die gesamte Architektur durch einen einheitlichen Kontrollpfad. Zugriffe auf den Datenspeicher werden durch eine weitere Memory-Management Unit (MMU) realisiert. Diese Architektur ist adaptierbar auf verschiedene Signalverarbeitungsalgorithmen, welche eine bestimmte Form an Parallelität nutzen können. Für den Ansatz in dieser Veröffentlichung wurde eine Konfiguration mit 3 Clustern mit jeweils 3 FUs beispielhaft gewählt.

2.1 Redundanz & zusätzliche Hardware

In Abbildung 2 ist die erweiterte Architektur dargestellt. Um die originale Architektur für diesen Ansatz anzupassen wurden zuerst mehrfach ausgelegte Baugruppen auf RT-Level identifiziert und mit zuschaltbarer Redundanz versehen. In diesem Fall wurden für die drei FUs pro Cluster eine Ersatzeinheit implementiert. Die acht RPs pro Cluster wurden in zwei Gruppen unterteilt, denen jeweils ein Ersatzelement zur Verfügung steht. Die drei BFUs aller Cluster erhielten ebenfalls eine Ersatzkomponente. Die gewählten Ersetzungsstrategien wurden im Vorfeld auf Effizienz geprüft. Für dieses gewählte Beispiel stellten sie sich als optimal heraus. Der Austausch der Ersatzkomponenten kann durch bestimmte Adressen

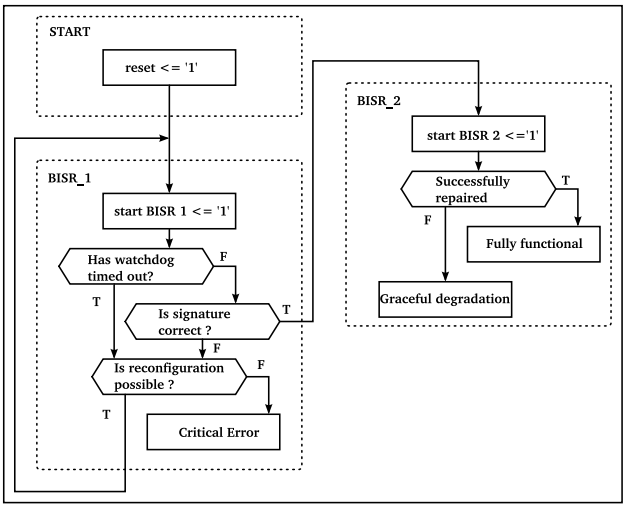


Abbildung 3: ASM Diagramm des gesamten Reparaturprozesses

des Datenspeichers, welche mit den Schaltern fest verdrahtet sind, erreicht werden. Somit ist eine Rekonfiguration des Prozessors durch Software möglich. Die zusätzliche Watchdog Komponente, welche ebenfalls durch festgelegte Adressen des Datenspeichers ansprechbar ist, sorgt für die Auflösung von möglichen Verklemmungen bei der Befehlsausführung auf defekter Hardware. Diese Baugruppe wird benötigt, damit die Test-, und Reparaturprozedur trotz Fehlern in nicht reparierbaren Teilen des Prozessors einen definierten Endzustand erreicht. Diese Watchdog Komponente hat ebenfalls Zugriff auf die Rekonfigurationsadressen der Ersatzelemente und kann gegebenenfalls einen Austausch veranlassen. Details der Ersetzungsstrategie werden im folgenden Kapitel beschrieben.

3 Test und Reparatur Prozedur

Die gesamte Test und Reparaturprozedur ist in Abbildung 3 dargestellt. Sie besteht aus zwei unterschiedlichen Softwareroutinen, die nach einem Neustart des Prozessors nacheinander von ihm abgearbeitet werden. Die erste Software testet alle Baugruppen des Prozessors, welche keine Redundanz besitzen (CP, MMU, Register-File). Weiterhin wird die BFU aus Cluster 1 und die dazugehörigen RPs getestet. Dabei wird diese Software durch die Watchdog Komponente überwacht. Diese erste Routine wird benötigt, um eine Ausführung der zweiten Software auf fehlerfreien Komponenten zu gewährleisten.

Die zweite Software ist eine diagnostische Testprozedur, welche sukzessiv alle verbleibenden Baugruppen testet und Fehler in ihnen diagnostiziert. Diese Software ist auch in der Lage, fehlerhafte Baugruppen auszutauschen. Weiterhin benötigt diese Software keine Überwachung durch den Watchdog, da sie auf fehlerfreien Baugruppen ausgeführt wird.

3.1 Testsoftware Teil 1

Die Ausführung diese ersten Software ist zwingend notwendig, um die fehlerfreie Ausführung der zweiten Software zu gewährleisten. Ohne diese Routine wäre die folgende nicht in der Lage Fehler richtig zu diagnostizieren noch den korrekten Austausch von Baugruppen vorzunehmen. Mit einem Neustart des Systems wird der Watchdog gestartet und die Befehlsabarbeitung dieser Routine beginnt. Der Test wird durch das Anlegen von Testmustern an die beschriebenen Komponenten umgesetzt. Im fehlerfreien Fall wird zusätzlich durch die Software eine

Signatur erzeugt. Diese wird nach dem Ablauf der Software in den Datenspeicher gesichert. Danach vergleicht der Watchdog diese Signatur mit einer gespeicherten Referenz. Bei Übereinstimmung der Signaturen gilt dieser Teil des Prozessors als fehlerfrei und der diagnostische Test beginnt.

Ein Fehler in den getesteten Komponenten kann folgende Auswirkungen haben:

- Signatur ungleich wegen:
  - Datenfehler in der MMU
  - Datenfehler in der BFU oder ihrer RPs
  - Datenfehler durch Defekte in Registern
  - Datenfehler durch den Kontrollpfad (CP)
- Watchdog timeout wegen:
  - Bit-Flips auf Adressen
  - Fehler bei der Befehlsdekodierung
  - Fehlerhafte Flags
  - Endlosschleifen
  - Fehler im Kontrollpfad, welche die Befehlsausführung betreffen

Für beide Fälle versucht die Watchdog Komponente eine Trial- and Error Rekonfiguration der verwendeten BFU und ihrer RPs aus Cluster 1 und startet die Routine und den Watchdog erneut. Falls die Signatur durch die Rekonfiguration einer Überprüfung durch den Watchdog standhält, wird die zweite Routine mit dieser neuen Konfiguration gestartet. Führt keine Konfiguration zu einem positiven Ergebnis gilt der Prozessor als defekt.

3.2 Testsoftware Teil 2

Auf der gefundenen fehlerfreien Konfiguration der vorherigen Routine wird nun eine diagnostische Software ausgeführt. Diese erkennt Fehler in den verbleibenden Baugruppen und ersetzt diese gegebenenfalls durch das Beschreiben bestimmten Adressen im Datenspeicher. Die Watchdog Komponente wird für diesen Teil des Reparaturprozesses nicht mehr benötigt und abgeschaltet. Wird eine fehlerfreie Konfiguration durch die Testsoftware gefunden, dann kann der Prozessor mit der eigentlichen Befehlsabarbeitung beginnen. Gibt es keine fehlerfreie Konfiguration, wird das Ergebnis der Test und Diagnose Software gespeichert. Mit dieser Information kann eine Anpassung der Algorithmen stattfinden und der Prozessor kann seine Aufgabe mit eingeschränkter Genauigkeit ausführen [11].

4 Ergebnisse

In diesem Kapitel werden die erzielten Ergebnisse für diese Beispielarchitektur dargestellt und erläutert. In Tabelle 1 ist der zusätzliche Hardware Aufwand für das gewählte Beispiel aufgelistet. Der gesamte Mehraufwand beträgt 17.54%. In Spalte 3 ist der zusätzliche nicht zu reparierende Anteil aufgelistet, welcher im Wesentlichen aus Schaltern besteht die zur Rekonfiguration benötigt werden. Die restlichen Spalten beinhalten den Mehraufwand für die Ersatzkomponenten. Alle prozentualen Angaben beziehen sich auf die gesamte Architektur.

Tabelle 1: Größe und zusätzliche Hardware für diesen Selbstreparaturansatz

	ALL	**CP**	**BFU**	**FU**	**RP**
VLIWmm^2	1.87	0.86	0.05	0.37	0.58
BISRmm^2	2.2	0.90	0.07	0.5	0.73
HW-Overhead(%)	17.54	2.08	0.95	6.69	7.82

Tabelle 2 zeigt den Effekt dieses Ansatzes auf die Produktionsausbeute. Dabei bezeichnet die erste Spalte die ursprüngliche Produktionsausbeute der originalen Architektur. Spalte 2 und 3 beinhalten die Anzahl der funktionierenden Chips auf einem Wafer mit 30cm Durchmesser.

Tabelle 2: Funktionierende Chips pro Wafer für unterschiedliche Ausbeuten

Die Yield	**Working Dies / Wafer**		**Ratio**
	VLIW	VLIW(BISR)	
25%	82	138	1.68
50%	164	196	1.19
75%	247	246	0.99
95%	313	270	0.86

Abbildung 4 zeigt den Effekt auf die Zuverlässigkeit für beide Architekturen bei einer konstanten Fehlerrate pro Flächenequivalent. Die Modellierung der Ausbeute und der Zuverlässigkeit wurde mit Hilfe von N aus M Systemen umgesetzt. Ebenfalls wird eine unabhängige Gleichverteilung der Fehler angenommen.

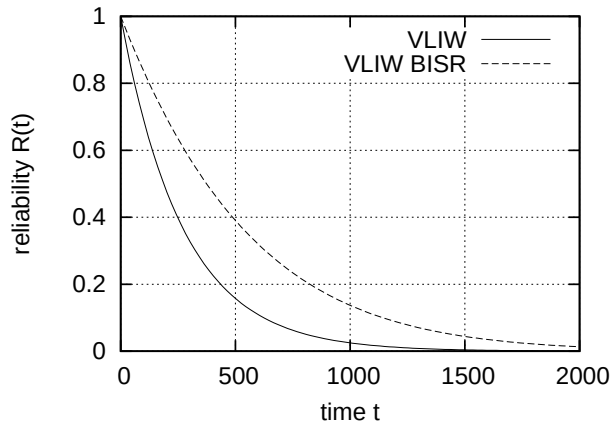


Abbildung 4: Zuverlässigkeit mit einer konstanten Fehlerrate pro Flächenequivalent

In Tabelle 3 ist die Fehlerüberdeckung und die benötigten Instruktionen der einzelnen Softwareroutinen für die jeweiligen Baugruppen aufgelistet.

Tabelle 3: Fehlerüberdeckung und Anzahl von Befehlen der einzelnen Softwareroutinen

	SW 1 %	**SW 2 %**	**ALL %**	**SW 1**	**SW 2**
CP	98.84	–[1]	98.84	53424	–[2]
BFU	$33.\overline{33}$	$66.\overline{66}$	100	3210	6420
ALU	–[1]	99.57	99.57	–[2]	95760
RP	$8.\overline{33}$	$91.\overline{66}$	100	15874	47622

[1] Fehlerüberdeckung nicht explizit berechnet
[2] Keine Ausführung von Testbefehlen für diese Komponente

5 Zusammenfassung

Zukünftige integrierte Schaltungen werden anfälliger sein gegenüber permanenten Fehler nach der Produktion als auch im Feld. Damit beide Typen von Fehlern behandelt werden können, benötigt man Test- und Reparaturmechanismen auch für den Logikanteil von SoCs. Der in dieser Veröffentlichung vorgestellte Ansatz ist in der Lage sowohl die Produktionsausbeute als auch die Zuverlässigkeit von Prozessoren zu erhöhen. Wobei die Ausbeute nur für Produktionsprozesse erhöht werden kann, die über einen schlechte Ausbeute verfügen. Dies liegt an der zusätzlichen Hardware, welche natürlich ebenfalls Produktionsfehlern unterliegt. Eine Erhöhung der Zuverlässigkeit für konstante Fehlerraten um den Faktor 2 wurde erzielt. Durch die Verwendung von Software für den Test und die Reparatur des Prozessors ist dieser Ansatz flexibel einzusetzen, dabei ist ein "start-up" als auch ein periodischer Test im Feld möglich. Durch das Erkennen von Fehlern mit Hilfe von Software beträgt der gesamte zusätzliche Aufwand zur Organisation der Reparatur nur 2.08%. Die restliche Mehraufwand an Hardware von 15.46% besteht aus den zusätzlich implementierten Redundanzen. Methoden zur Kompensation von transienten Fehlern sind durch diesen Ansatz nicht abgedeckt.

Literatur

[1] "International technology roadmap for semiconductors," ITRS, http://public.itrs.net, Tech. Rep., 2007.

[2] Jedec, "Failure mechanisms and models for semiconductor devices," JEDEC SOLID STATE TECHNOLOGY ASSOCIATION, Tech. Rep., 03 2006.

[3] M. Choi, N. Park, F. Lombardi, Y. B. Kim, and V. Piuri, "Optimal spare utilization in repairable and reliable memory cores," in *Proc. Records of the 2003 International Workshop on Memory Technology, Design and Testing*, 2003, pp. 64–71.

[4] Y. Zorian and S. Shoukourian, "Embedded-memory test and repair: Infrastructure ip for soc yield," *IEEE DESIGN & TEST*, vol. 20, no. 03, pp. 58–66, 2003.

[5] I. Kim, Y. Zorian, G. Komoriya, H. Pham, F. P. Higgins, and J. L. Lewandowski, "Built in self repair for embedded high density sram," *itc*, vol. 00, p. 1112, 1998.

[6] J.-F. Li, J.-C. Yeh, R.-F. Huang, C.-W. Wu, P.-Y. Tsai, A. Hsu, and E. Chow, "A built-in self-repair scheme for semiconductor memories with 2-d redundancy," *itc*, vol. 00, p. 393, 2003.

[7] A. Nieuwland and R. Kleihorst, "The positive effect on ic yield of embedded fault tolerance for seus," in *Proc. 9th IEEE On-Line Testing Symposium IOLTS 2003*, 2003, pp. 75–79.

[8] M. Valinataj and S. Safari, "Fault tolerant arithmetic operations with multiple error detection and correction," in *Proc. 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems DFT '07*, 26–28 Sept. 2007, pp. 188–196.

[9] L. Chen and S. Dey, "Defuse: a deterministic functional self-test methodology for processors," in *Proc. 18th IEEE VLSI Test Symposium*, 2000, pp. 255–262.

[10] A. Pillai, W. Zhang, and D. Kagaris, "Detecting vliw hard errors cost-effectively through a software-based approach," in *Proc. 21st International Conference on Advanced Information Networking and Applications Workshops AINAW '07*, vol. 1, 2007, pp. 811–815.

[11] P. Pawlowski, A. Dabrowski, and M. Schölzel, "Proposal of vliw architecture for application specific processors with built-in-self-repair facility via variable accuracy arithmetic," in *DDECS*, 2007, pp. 313–318.

Möglichkeiten und Grenzen der Software-basierten Selbstreparatur in statisch geplanten superskalaren Prozessorarchitekturen

Mario Schölzel
mas@informatik.tu-cottbus.de
Lehrstuhl für Technische Informatik, Brandenburgische Technische Universität Cottbus

Kurzfassung

In statisch geplanten superskalaren Prozessorarchitekturen ist es die Aufgabe des Compilers die Operationen der Anwendung an die vorhandenen Ausführungseinheiten zu binden. Treten dann permanente Fehler im Datenpfad auf, so wird die festgelegte Bindung ungültig und das durch den Compiler erzeugt Programm kann nicht fehlerfrei auf dem Prozessor ausgeführt werden. Auf der anderen Seite bieten diese Architekturen den Vorteil, dass aus dem Programmablaufplan abgeleitet werden kann, welche Komponenten durch welche Operationen genutzt werden. Durch eine geschickte Umplanung der Operationen kann dann die Verwendung fehlerhafter Komponenten vermieden werden. In diesem Beitrag werden auf analytische Weise die Komponenten eines statisch geplanten Prozessors auf Fehlerpunkte untersucht und software-basierte Methoden zur Behandlung von permanenten Defekten in diesen Fehlerpunkten vorgestellt. Es zeigt sich, dass dadurch eine sehr feingranulare Behandlung von Defekten möglich wird.

Einleitung

Auf Grund immer kleinerer Strukturgrößen bei der Fertigung von CMOS-Schaltungen nimmt deren Zuverlässigkeit mit steigender Integrationsdichte ab. Dadurch sind Nano-CMOS-Schaltungen anfälliger für transiente Fehler, da die umzuladenden Kapazitäten überall sehr gering sind und deshalb durch externe Ereignisse (z.B. Strahlung) leicht verändert werden können [1, 3]. Weiterhin können kleinste Variationen im Produktionsprozess zu einem veränderten Schaltverhalten von Transistoren führen, das sich dann außerhalb der Spezifikation befinden kann. Solche Fehler führen zu permanenten Fehlern, die frühzeitig auftreten können [12]. Nano-Strukturen sind aber auch höherem Stress ausgesetzt. Dies führt zu Verschleißerscheinungen, die zu einem späteren Ausfall oder Fehlverhalten von Transistoren führen können [10]. Um die Zuverlässigkeit solcher Systeme und insbesondere die der darin verwendeten programmierbaren Prozessoren zu erhöhen, können die Prozessoren mit Funktionen zur Selbstreparatur ausgestattet werden, die es erlauben, permanente Fehler während des Einsatzes zu behandeln.

Vorhandene Arbeiten

Es existieren zahlreiche Ansätze, um programmierbare Prozessoren mit Selbstreparaturfunktionen auszustatten. Sie basieren in der Regel darauf, dass die Verwendung defekter Komponenten vermieden wird. Dabei können zwei wesentliche Strategien unterschieden werden. Erstens: Es findet eine dauerhafte Umkonfigurierung des Systems statt. Dies ist beispielsweise bei der Verwendung von FPGAs der Fall [9]. Auch ASICs, die mit entsprechenden Möglichkeiten ausgestattet sind, können dauerhaft umkonfiguriert werden [4, 8]. Zweitens: Während der Ausführung der Operationen im Prozessor wird dynamisch die Verwendung defekter Komponenten vermieden [2]. Auch hier ist der ASIC mit zusätzlichen Hardwarestrukturen (z.B. Multiplexern) ausgestattet, so dass während der Abarbeitung einer Anwendung die Verwendung fehlerhafter Komponenten vermieden werden kann. Insbesondere in statisch geplanten Prozessoren muss die Umplanung von Operationen auf andere Ausführungseinheiten nicht dynamisch erfolgen, sondern kann statisch im Programmcode vorgenommen werden. Dieser Ansatz bietet den Vorteil, dass die gesamte Administration zur Vermeidung der Nutzung von fehlerhaften Komponenten in Software durchgeführt werden kann. Arbeiten, die mehrere Versionen desselben Programms bereitstellen, wobei jede Version eine bestimmte Menge von Fehlersituationen handhaben kann, sind [5, 7]. Das Problem hier ist, dass die unterschiedlichen Versionen eines Programms vorausberechnet und im Programmspeicher abgelegt werden müssen. Schon für mittelgroße Anwendungen ist dieser Ansatz nicht mehr praktikabel. Die in diesem Beitrag vorgestellte Methode

basiert auf der Anpassung des Binärcodes des Programms an permanent aufgetretene Fehler im Prozessor. Es wird gezeigt, dass sich damit Fehler auf einer sehr feingranularen Ebene behandeln lassen, was insbesondere bei der Behandlung von Mehrfachfehler wichtig ist.

Software-basierte Selbstreparatur

In diesem Beitrag werden die Möglichkeiten und Grenzen einer software-basierten Selbstreparatur von statisch geplanten superskalaren Prozessorarchitekturen analysiert. Grundlage für die Betrachtungen bildet die Prozessorarchitektur in Abbildung 1 (a). Die software-basierte Selbstreparatur basiert auf der Manipulation des Binärcodes im Programmspeicher des Prozessors. Dadurch kann die Verwendung fehlerhafter Komponenten im Prozessor vermieden werden. Schrittweise werden Teile des Programms in den Datenspeicher transferiert, dort durch verschiedene Transformationen an einen aufgetretenen Fehler adaptiert und das manipulierte Programm in den Programmspeicher zurück geschrieben [11]. Da die Selbstreparatur in einem Prozessor ausgeführt wird, der sich im Einsatz befindet, ist eine wesentliche Anforderung, dass die Modifikation des Binärcode schnell ablaufen kann.

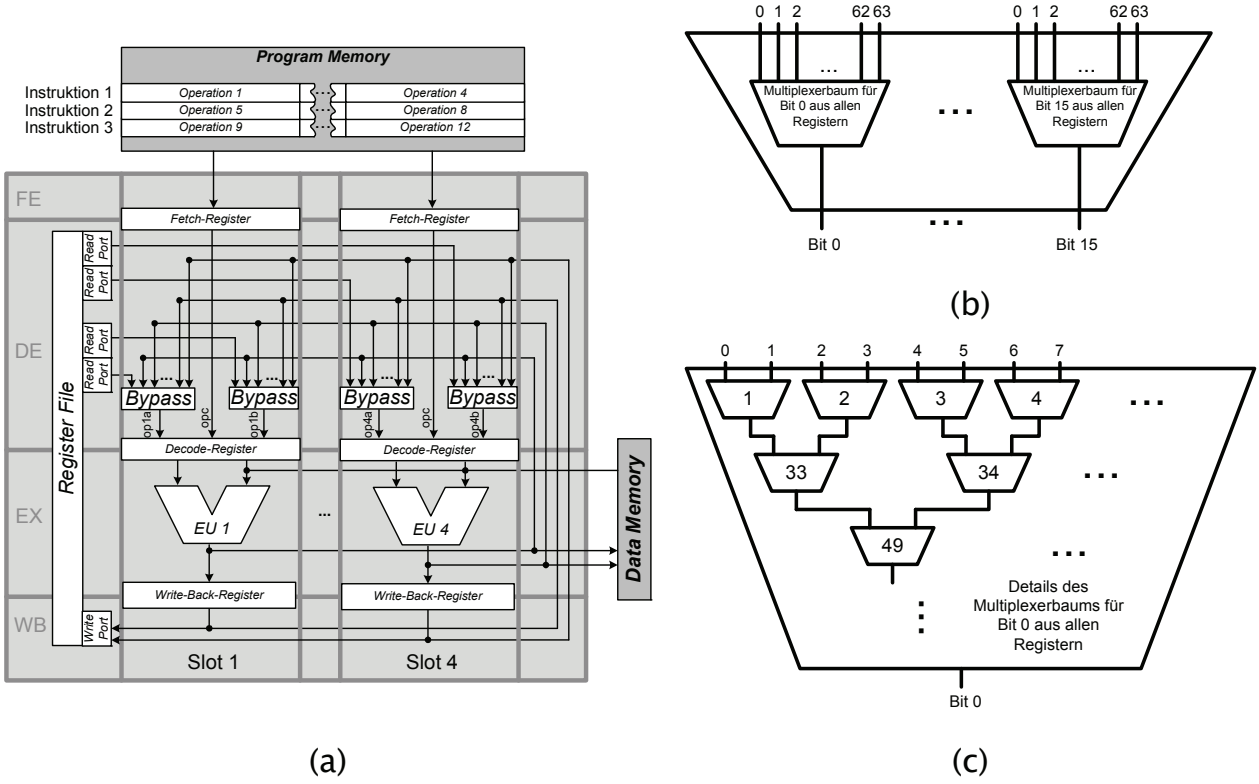


Abbildung 1: (a) Modell eines VLIW-Prozessors. (b) Leseport der Registerbank bestehend aus 16 Multiplexerbäumen. (c) Details eines Multiplexerbaums bestehend aus 2:1-Multiplexern.

Der Prozessor in Abbildung 1 (a) besitzt eine vierstufige Pipeline (*FE*, *DE*, *EX*, *WB*). In jedem Takt wird eine Instruktion, in der 4 Operationen kodiert sind, aus dem Programmspeicher geholt und jede der Operationen in dem zugehörigen *Fetch-Register* abgelegt. Die Operationen in einem Instruktionswort sind statisch an einen *Slot* des Prozessors gebunden. Alle vier Operationen durchlaufen parallel die drei folgenden Pipelinestufen. In der *Decode-Phase* werden die benötigten Registerwerte bereitgestellt, die dann in der *Execute-Phase* durch die Ausführungseinheit (*EU*) verknüpft werden. Das Ergebnis der Operation wird im *Write-Back-Register* abgelegt, bevor es in der letzten Pipeline-Phase in die Registerbank zurück geschrieben wird. Durch einen *Bypass* können Werte aus der *Execute*-Phase und der *Write-Back*-Phase unter Umgehung der Registerbank direkt in ein *Decode-Register* übernommen werden.

Analyse der Fehlerpunkte

Das in Abbildung 1 (a) angegebene Prozessormodell wurde aus einer VHDL-Beschreibung synthetisiert, um die Größen der einzelnen Komponenten in Transistoräquivalenten zu ermitteln.

Tabelle 1 zeigt die Ergebnisse. Es ist zu erkennen, dass die Ausführungseinheiten (EU) sowie die Leseports der Registerbank (RP) einen sehr großen Anteil an der Gesamtfläche ausmachen. Für die Fehleranalyse wurde ein einfaches Haftfehlermodell verwendet. Die Auswirkung eines Haftfehlers kann auf Grund der Verdrahtung im VLIW-Modell analysiert werden.

Komponente	Ctrl	FE-Reg	DE-Reg	Bypass	EU	WB-Reg	RP	WP
Transistoren	800	2300	1300	1400	7000	500	6900	34800
Instanzen	1	1	4	8	4	4	8	1

Tabelle 1: Größe der Komponenten des VLIW-Prozessors aus Abbildung 1 (a) in Transistoräquivalenten.

Wird die Nutzung eines kompletten Slots vermieden, dann können damit Defekte in den Komponenten *Bypass*, *DE-Reg*, *EU*, *WB-Reg* und *RP* dieses Slots umgangen werden. Das setzt allerdings voraus, dass der Slot kein Lade-Signal für den PC und kein Ladesignal für die Registerbank generiert, wenn er mit NOP-Operationen versorgt wird. Es empfiehlt sich daher diese Signale in der *Decode*-Phase aus dem Opcode zu generieren, und dieses Signal dann durch die einzelnen Pipelinestufen zu propagieren. Damit müssen nur das *FE-Register* sowie die Flip-Flops zum Speichern und Propagieren der Lade-Signale immer fehlerfrei arbeiten.

Die Nutzung eines kompletten Slots muss zwingend vermieden werden, wenn Fehler in seinem *Decode*-Register oder *Writeback*-Register auftreten. Fehler in den übrigen Komponenten des Slots führen nicht immer zwingend zu seinem Totalausfall. Dies ist der Fall bei Fehlern in den Ausführungseinheiten, den Leseports und dem *Bypass*. Eine detaillierte Lokalisierung der Fehlerpunkte in diesen Komponenten kann eine feingranulare Reparatur ermöglichen.

Ausführungseinheiten: Abhängig vom Aufbau der Ausführungseinheiten, wirken sich Fehler in der Ausführungseinheit möglicherweise nur auf eine bestimmte Klasse von Operationen aus. Operationen einer anderen Art können dann trotzdem noch fehlerfrei in diesem Slot ausgeführt werden.

Leseport: Ein Leseport der Registerbank besteht bei einem 16 Bit Datenpfad aus 16 Multiplexerbäumen, wie es in Abbildung 1 (b) dargestellt ist. Jeder Multiplexerbaum wiederum ist ein Binärbaum, der aus 2:1-Multiplexern aufgebaut ist, wie es in Abbildung 1 (c) zu sehen ist. Ein Haftfehler in einem 2:1-Multiplexer in diesem Baum kann durch einen nachfolgenden Multiplexer maskiert werden. Allerdings können dann die Werte der Register nicht mehr ausgelesen werden, deren Werte über einen Pfad, der durch den fehlerhaften Multiplexer führt, ausgelesen werden würden. Die Wahrscheinlichkeit, dass ein Fehler in einer unteren Ebene des Leseports auftritt ist relativ hoch, da sich die überwiegende Anzahl der 2:1-Multiplexer in den unteren Ebenen des Multiplexerbaums befinden. Ein Fehler in einem Leseport kann somit behandelt werden, wenn über diesen Port die betroffenen Register, deren Werte nicht mehr korrekt ausgelesen werden können, nicht mehr angesprochen werden.

Bypass: Ein *Bypass* ist, wie der Leseport auch, eine Multiplexerstruktur. Diese wählt entweder den durch die Registerbank bereitgestellten Wert oder einen der vier Werte aus der *EX*-Phase oder einen der vier Werte aus der *WB*-Phase aus. Ein *Bypass* kann damit wie ein Leseport für eine Registerbank mit neun Registern aufgefasst werden. Durch einen Haftfehler kann entweder nicht mehr korrekt aus der Registerbank, oder aus einer der zwei Pipelinestufen gelesen werden. Fehler dieser Art können umgangen werden, indem nur noch Werte aus der Registerbank oder nur noch Werte aus den Pipelinephasen EX oder WB in dem zugehörigen Slot verwendet werden.

Im Folgenden soll gezeigt werden, wie die eben beschriebenen Fehlerpunkte durch eine Änderung des Binärcodes umgangen werden können. Die Operationen im Binärcode werden dafür durch einen einfachen Planungsalgorithmus umgeordnet. Abhängig von den dabei betrachteten Fehlerpunkten, kann die Reparatur mit unterschiedlicher Genauigkeit erfolgen.

Behandlung von Fehlern in den Ausführungseinheiten

Der Defektzustand einer Ausführungseinheit EU k kann durch einen Fehlervektor $F_k \in \{0,1\}^n$ charakterisiert werden, wobei $F_k(i)$ den Zustand des Operators i in der Ausführungseinheit k kennzeichnet. Falls $F_k(i) = 0$ für alle Operatoren i, dann ist die gesamte Ausführungseinheit defekt. Äquivalent dazu sind Fehler im *Decode*-Register oder dem *Writeback*-Register desselben Slots. In allen drei Fällen kann der gesamte Slot nicht mehr verwendet werden. Eine sehr schnelle software-basierte Reparatur solcher Defekte ist beispielsweise durch eine Permutation der Operationen innerhalb einer Instruktion möglich [11]. Eine weitere Alternative stellt die Änderung der Ablaufplanung innerhalb eines Basisblocks dar. Diese Methode soll in diesem Beitrag ausführlicher beschrieben werden und schrittweise zur Behandlung von Defekten im *Bypass* und den Lesports erweitert werden. Die Ablaufplanung eines Basisblocks kann im Feld effizient durch eine Implementierung des Scoreboarding-Verfahrens [6] in Software erreicht werden. Der grundlegende Algorithmus ist in Abbildung 2 angegeben.

```
while not all operations are scheduled do
  Create a new empty instruction currInstr
  for slot := 1 to 4 do                  // Alle Slots der Instruktion bearbeiten
    if EU[slot] = 0 then                 // Führt die EU noch eine frühere Operation aus?
      for reg := 0 to 64 do              // Nein, dann Initialisierung von Def und Use
        Def[reg] := 0;
        Use[reg] := 0;
      od
      for each op in priority list do in priority
        if op.time = currInstr then
          Def[op.dst] := 1;             // Generierter Wert kann nicht in derselben Instruktion
        fi                              // verwendet werden.

        if op.state != scheduled then
          if F_slot(op) && CanBeScheduled(op) then
              EU[i] := lat(op)
              instr[slot] := op
              op.time := currInstr;
              lastDef[op.dst] := currInstr;
              op.state := scheduled;
              break;
          else
            Use[op.Src1] := 1
            Use[op.Src2] := 1
            Def[op.Dst] := 1
          fi
        fi
      od
    fi
  od
  if EU[1] > 0 then EU[1]--;
  if EU[2] > 0 then EU[2]--;
  if EU[3] > 0 then EU[3]--;
  if EU[4] > 0 then EU[4]--;
od
```

Abbildung 2: Algorithmus zur Änderung der Ablaufplanung.

Die Operationen des Basisblocks, der bereits aus dem Programmspeicher in den Datenspeicher transferiert wurde, sind in einer Prioritätsliste gegeben. Die Reihenfolge der Operationen in der Prioritätsliste entspricht der Reihenfolge im ursprünglichen Programm. Abhängigkeiten zwischen den Operationen werden durch die zwei Felder `Def` und `Use` während der Erzeugung eines neuen Ablaufplans modelliert. Der neue Ablaufplan wird Instruktionsweise erzeugt und für jede neu hinzugefügte Instruktion (`currInst`) werden planbare Operationen für die vier Slots in der Instruktion gesucht. Um eine planbare Operation zu finden, werden die Operationen in der Prioritätsliste der Reihe nach abgearbeitet. Mit der Bedingung `CanBeScheduled(op)` wird überprüft, ob die aktuell in der Prioritätsliste betrachtete Operation planbar ist. Diese Bedingung muss die Form

```
Def[op.Src1] == 0 && Def[op.Src2] == 0 &&
Def[op.Dst] == 0 &&
Use[op.Dst] == 0
```

haben, damit alle Abhängigkeiten eingehalten werden. Mit der ersten Zeile wird die Einhaltung normaler Flussabhängigkeiten abgesichert. Die Planung von `op` verletzt keine Flussabhängigkeiten, wenn es in der Prioritätsliste keine andere ungeplante Operation vor `op` gibt, die `Src1` bzw. `Src2` als Zielregister hat. Diese Operation wäre nämlich vor `op` auf ihre Planbarkeit überprüft worden. Da sie aber nicht verplant wurde, wäre die, dem Zielregister *d* entsprechende Position im Feld `Def` (`Def[d]`), markiert worden. Analog dazu erzwingt die zweite Zeile die Beachtung von Ausgabeabhängigkeiten. Mit der dritten Zeile wird die Beachtung von Antiabhängigkeiten erzwungen. `op` kann nur dann geplant werden, wenn es keine frühere Operation in der Prioritätsliste gibt, die noch nicht geplant ist und die das Zielregister von `op` liest.

Behandlung von Fehlern im Bypass

Die Ablaufplanung bietet auch die Möglichkeiten Fehler im Bypass des Prozessors und Totalausfälle der Leseports zu behandeln. Bei einem Totalausfall des linken bzw. rechten Leseports kann die EU im zugehörigen Slot weiterhin mit Daten über den entsprechenden Bypass versorgt werden. Eine Operation `op` ist dann nur noch in diesen Slot planbar, falls der Wert des linken bzw. rechten Operanden innerhalb der letzten zwei Takte erzeugt wurde. Falls der linke bzw. rechte Bypass aus der *EX*-Phase defekt ist, dann dürfen in diesen Slot nur Operationen geplant werden, deren linker bzw. rechter Operand nicht durch eine Operation im vorangegangenen Takt erzeugt wurde. Falls der Bypass aus der *WB*-Phase defekt ist, dann darf der Wert des linken bzw. rechten Operanden von `op` nicht durch eine Operation erzeugt worden sein, die zwei Takte zuvor ausgeführt wurde. Diese Bedingungen lassen sich leicht in die Ablaufplanung aus Abbildung 2 integrieren, indem die Bedingung `CanBeScheduled(op)` definiert wird als:

```
Def[op.Src1] == 0 && Def[op.Src2] == 0 &&
Def[op.Dst] == 0 && Use[op.Dst] == 0 &&
(currInstr - lastDef[op.src1] = 1 && (bypassState[2*slot] & 1) ||
 currInstr - lastDef[op.src1] = 2 && (bypassState[2*slot] & 2) ||
 currInstr - lastDef[op.src1] > 2 && (bypassState[2*slot] & 4) &&
(currInstr - lastDef[op.src2] = 1 && (bypassState[2*slot+1] & 1) ||
 currInstr - lastDef[op.src2] = 2 && (bypassState[2*slot+1] & 2) ||
 currInstr - lastDef[op.src2] > 2 && (bypassState[2*slot+1])
```

Das Feld `lastDef` speichert an Position *i* die Nummer der letzten Instruktion, die das Register *i* beschrieben hat. `currInstr` ist die Nummer der Instruktion, in die die Operation `op` geplant werden soll. Der Fehlerzustand der Komponente *Bypass* ist im Feld `bypassState` kodiert. An Position `2*slot` bzw. `2*slot+1` befindet sich die Defektzustand des linken bzw. rechten Bypasses im Slot `slot`. Bit 1 kodiert einen Fehler im Bypass der EX-Phase, Bit 2 einen Fehler im Bypass der WB-Phase und Bit 3 einen Totalausfall des Leseports.

Behandlung von Fehlern im Leseport

Mit den eben beschrieben Erweiterungen lassen sich Totalausfälle von Leseports behandeln. Da ein Leseport aber ein großer Multiplexerbaum ist, ist die Wahrscheinlichkeit, dass ein Defekt in einer unteren Ebene auftritt, relativ hoch. Auf Register, die über einen Pfad ausgelesen werden, der nicht durch den defekten 2:1-Multiplexer führt, kann weiterhin zugegriffen werden. Es muss also nur sicher gestellt werden, dass über den betroffenen Port die nicht erreichbaren Register nicht mehr gelesen werden. Durch eine Anpassung der Bedingung `CanBeScheduled(op)` ist das wieder leicht möglich:

```
Def[op.Src1] == 0 && Def[op.Src2] == 0 &&
Def[op.Dst] == 0 && Use[op.Dst] == 0 &&
(currInstr - lastDef[op.src1] = 1 && (bypassState[2*slot] & 1) ||
 currInstr - lastDef[op.src1] = 2 && (bypassState[2*slot] & 2) ||
 currInstr - lastDef[op.src1] > 2 && regState[2*slot][op.src1]) &&
(currInstr - lastDef[op.src2] = 1 && (bypassState[2*slot+1] & 1) ||
 currInstr - lastDef[op.src2] = 2 && (bypassState[2*slot+1] & 2) ||
 currInstr - lastDef[op.src2] > 2 && regState[2*slot+1][op.src2])
```

Ob das Register *r* über den linken bzw. rechten Leseport im Slot `slot` noch fehlerfrei ausgelesen werden kann, wird im Feld `regState` an Position `[2*slot][r]` bzw. `[2*slot+1][r]` gespeichert. Dieser Fehlerzustand ersetzt den Defektzustand des Leseports, der im Feld `bypassState` abgelegt war. Dadurch lässt sich auch ein vollständig defekter Leseport beschreiben, indem kein Register über diesen Port erreichbar ist.

Ergebnisse und Zusammenfassung

In den Datenpfad des VLIW-Prozessors wurde gleichzeitig ein Fehler in den Addierer von Slot 2, in den Bypass von Slot 1 und in den Leseport von Slot 4 injiziert. In der Tabelle 2 ist die Entwicklung der Ablaufplanlängen von drei kleinen Benchmarkprogrammen nach einer Umplanung der Operationen bei unterschiedlicher Reparaturgenauigkeit dargestellt. Unterstützt die Ablaufplanung nur eine Behandlung von Fehlern auf der Ebene von Slots, dann führen die angegebenen Fehler dazu, dass nur Slot 3 verwendet werden kann. Können Fehler auf der Ebene von Slots und Operatoren behandelt werden, dann kann Slot 3 uneingeschränkt und Slot 2 eingeschränkt verwendet werden. Können auch Fehler im Bypass bzw. Leseport behandelt werden, dann können auch die Slots 1 bzw. 4 eingeschränkt verwendet werden.

Benchmark	Genauigkeit bei der Reparatur von Fehlern			
	Slots	Slots & Operatoren	Slots & Operatoren & Bypass	Slots & Operatoren & Bypass & Leseports
ARF	28	16	10	9
FFT	38	27	16	12
EWF	34	27	18	15

Tabelle 2: Längen der umgeplanten Ablaufpläne bei einer Reparatur mit unterschiedlicher Genauigkeit.

Die Ergebnisse zeigen, dass durch eine feingranulare Reparatur die Erhöhung der Ausführungszeit deutlich verringert werden kann, weil auch die Ausführungseinheiten in solchen Slots genutzt werden können, die Fehler im Bypass oder Leseport haben. Die feingranulare Behandlung von Defekten ist durch eine Software-basierte Umplanung der Operation leicht möglich. Diese bietet einen gravierenden Vorteil gegenüber Hardware-basierten Verfahren: Es wird keine zusätzliche Hardware zu Administrationszwecken benötigt. Weiterhin wurden in diesem Beitrag auch kritische Fehlerpunkte im Datenpfad aufgezeigt, die sich mit den beschriebenen Techniken nicht reparieren lassen. Aus den aufgezeigten Reparaturmöglichkeiten lassen sich nun sinnvolle Anforderungen an einen diagnostischen Selbsttest ableiten.

Literatur

[1] R. Baumann: *Soft Errors in Advanced Computer Systems*. IEEE Design & Test of Computers, 22(3), S. 258-266, 2005.

[2] Y.-Y. Chen, H. Shi-Jinn und L. Hung-Chuan: *An Integrated Fault-Tolerant Design Framework for VLIW Processors*. 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03), S. 555-562, 2003.

[3] P. E. Dodd und L. W. Massengill: *Basic Mechanisms and Modeling of Single-Event Upsets in Digital Microelectronics*. IEEE Transactions on Nuclear Science, 50(3), S. 583-602, 2003.

[4] M. Franklin: *A Study of Time Redundant Fault Tolerance Techniques for Superscalar Processors*. International IEEE Workshop on Defect and Fault Tolerance in VLSI Systems (DFT'95), S. 207-215, 1995.

[5] L. Guerra, M. Potkonjak und J. M. Rabaey: *Behavioral-Level Synthesis of Heterogenous BISR reconfigurable ASIC's*. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 6(1), S. 158-167. 1998.

[6] J. L. Hennessy und D. A. Patterson: *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, 2003.

[7] R. Karri, K. Kim und M. Potkonjak: *Computer Aided Design of Fault-Tolerant Application Specific Programmable Processors*. IEEE Transactions on Computers, 49(11), S. 1272-1284, 2000.

[8] T. Koal und H. T. Vierhaus: *A software-based self-test and hardware reconfiguration solution for VLIW processors*. Proc. of the 13th IEEE International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS'10), 2010.

[9] S. Mitra, W.-J. Huang, N. R. Saxena, et. al.: *Reconfigurable Architecture for Autonomous Self-Repair*. IEEE Design & Test of Computers, 23(3), S. 228-240, 2004.

[10] S. P. Park und K. K. K. Roy: *Reliability Implications of Bias-Temperature Instability in Digital ICs*. IEEE Design & Test of Computers, 26(6), S. 8-17, 2009.

[11] M. Schölzel: *Software-Based Self-Repair of Statically Scheduled Superscalar Data Paths*. Proc. of the 13th IEEE International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS'10), 2010.

[12] S. Sirisantana, B. C. Paul und K. Roy: *Enhancing Yield at the End of the Technology Roadmap*. IEEE Design & Test of Computers, 21(6), S. 563-571, 2004.

Eine Konzeption zu Energieeffizienz- und Performanceanalysen mittels Debug-Werkzeugen für eingebettete System-on-a-Chip

Andreas Gajda, Dr. Steffen Köhler
{Andreas.Gajda, Steffen.Koehler}@tu-dresden.de
Technische Universität Dresden

Kurzfassung

Die Anforderungen an eingebettete "System-on-a-Chip" hinsichtlich Energieeffizienz, Sicherheit, und Zuverlässigkeit steigen stetig an. Sind Methoden wie z.B. der Einsatz von Multi-Core-Plattformen, Compiler-Optimierung, oder statische Programmanalysen ausgeschöpft, oder können gar nicht erst eingesetzt werden, ist eine bessere Ausnutzung der On-Chip–Ressourcen mit Daten des Laufzeitverhaltens dennoch möglich. Viele Entwicklungswerkzeuge unterstützen Profiling, mit dem unter anderem solche Optimierungen möglich werden. Jedoch erfolgt Profiling meist über eine Instrumentierung des Programmes und verändert damit sein Laufzeitverhalten. Für eine Vielzahl von eingebetteten Systemen ist dies jedoch nicht akzeptabel.

Es soll ein Konzept vorgestellt werden, das vorhandene Hardware-Trace-Technologie für Systemanalysen nutzbar machen soll. Der Ausgangspunkt des Konzeptes ist die formale Beschreibung einer Analyseaufgabe, aus der automatisch eine Trace-Parametrierung und die abschließende Trace-Daten-Auswertung abgeleitet werden. So wird eine Sicht auf das System als Ganzes erschlossen und eröffnet neue Möglichkeiten der optimierten Ressourcennutzung in der gewünschten Richtung.

Einleitung

Neue umfangreichere „System-on-a-Chip" (SoC) und deren potenzierte Möglichkeiten zur Interaktion und gegenseitigen Beeinflussung der Komponenten verlangen vom Entwickler zunehmend die Sicht von einzelnen Komponenten zu lösen. Durch die stetig steigenden Anforderungen an eingebettete Systeme z.B. in Richtung Energieeffizienz, Sicherheit und Zuverlässigkeit wird der Entwicklungsprozess solcher Systeme immer komplexer und schwerer beherrschbar. Das vorgestellte Konzept soll einen Teil der Komplexität der Systemoptimierung übernehmen und so für den Entwickler reduzieren.

Je nach Optimierungsrichtung können die einzelnen Schritte einer Analyse des aktuellen Systemzustandes variieren. Bei der Verwendung von Trace-Hardware sind jedoch allen Analysen die Teilschritte der analyseorientierten Trace-Parametrierung, Trace-Durchführung und Auswertung gemeinsam. Diese Schritte sollen durch eine formale Beschreibung der Trace-Spezifikation und Auswertung zu einer Analyseaufgabe zusammengefasst werden. Die Teilschritte werden dann durch ein Entwicklungswerkzeug automatisch durchgeführt.

Voraussetzungen

Zunehmend werden in kommerziellen SoC Trace-Hardware-Einheiten und Debug-Schnittstellen integriert und sind damit einem breiten Entwicklerkreis verfügbar. Der größte Mangel der Trace-Schnittstellen und On-Chip Speicher ist jedoch ihre Dimensionierung, die einen zeitnahen Transfer der zwangsläufig großen Trace-Datenmengen in den Trace-Host verhindert. Der logische Schluss ist die Begrenzung des zu erzeugenden Datenaufkommens durch intelligente Trace-Parametrierung. Dazu wird die Trace-Parametrierung um Kontext-Daten reduziert, die aus Trace-Daten rekonstruierbar sind. Aufgrund der begrenzten Konfigurationsmöglichkeiten der Trace-Hardware wird aber jede Parametrierung eine Abwägung zwischen möglicherweise redundanten Daten und dem Verlust notwendiger Daten.

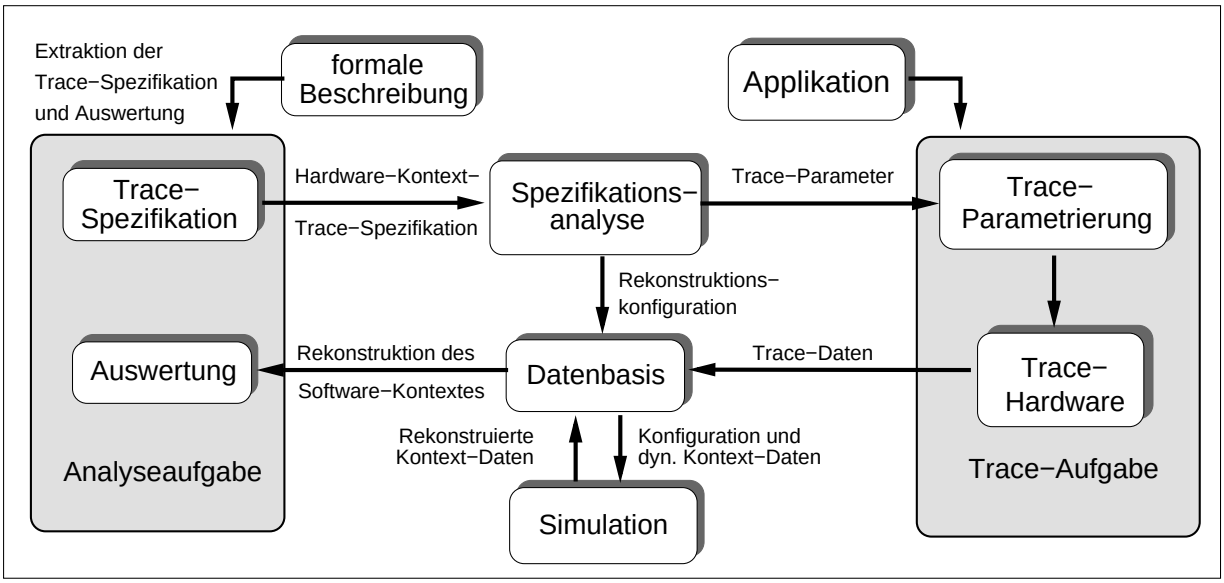


Abbildung : Schematische Darstellung des Prozesskonzepts

Wichtigste Voraussetzung ist hier natürlich, dass die Kontext-Informationen in großem Umfang sicher zu rekonstruieren sind. Einfluss haben darauf unter anderem die durchschnittliche Lebensdauer eines Registerinhalts, die Lokalität von Berechnungen und die Sichtbarkeit von Speicherzugriffen im Trace. Voraussetzung ist auch, dass ein großer Anteil der Hardware-Anweisungen deterministisch und frei von unberechenbaren Nebeneffekten ist.

Solch eine Parametrierung ist offensichtlich abhängig von untersuchten Software, der Analyseaufgabe, der verwendeten Trace-Hardware und der Architektur des Ziel-Prozessors. Flexibilität bei der Implementierung solch einer Technologie wird eine Voraussetzung für ihre praktische Verwendbarkeit sein.

Konzept

Betrachtet man den Entwicklungsprozess eines Systems, so fallen viele Routineaufgaben mit einem hohen manuellen Aufwand an. Einige dieser speziellen Aufgaben können jedoch auch automatisch durchgeführt werden, wie z.B. die Trace-Parametrierung und die Rekonstruktion des Kontextes aus den Trace-Daten. In beiden Fällen wird eine Übersetzung zwischen Hardware-Kontext und einem abstrakteren Niveau, dem Software-Kontext, vorgenommen. Diese Abstraktionsarbeit soll in Zukunft ein Entwicklungswerkzeug leisten.

Die Kernidee des Konzeptes (Abb. 1) ist die Überführung einzelner Iterationsstufen der Entwicklung in eine formale beschriebene Aufgabe, die anschließend durch ein Entwicklungswerkzeug bearbeitet wird. Eine Analyseaufgabe enthält zwei Komponenten. Eine ist die Spezifikation des Trace-Anteils auf Software-Ebene. Die Zweite bestimmt die Auswertung der Kontext-Daten in Form von Funktionen. Diese beiden Teile der Analyseaufgabe werden aus ihrer formalen Beschreibung durch einen speziellen Parser (Abb. 2) extrahiert. Die Trace-Spezifikation bildet den Anfangspunkt und die Auswertung den Endpunkt des nachfolgend beschriebenen Prozesses.

Nach der Extraktion der Trace-Spezifikation wird der Software-Kontext in den Hardware-Kontext überführt. Eine Spezifikationsanalyse untersucht nun, welche Anteile des Kontextes aus Trace-Daten rekonstruiert werden können. Können Kontext-Daten nicht rekonstruiert werden, so müssen sie mit Hardware-Trace ermittelt werden. Die Ergebnisse der Analyse werden in

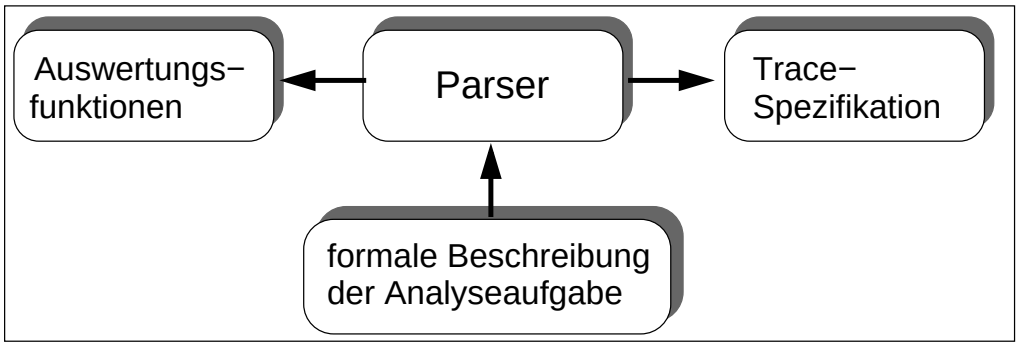


Abbildung : Extraktion der Trace-Spezifikation und Auswertungsfunktionen aus der formal beschriebenen Analyseaufgabe

einer Datenbasis für spätere Schritte festgehalten.[1] Die zu ermittelnden Kontext-Daten werden im nächsten Schritt in eine Parametrierung der Trace-Hardware übersetzt.

Bei der Erzeugung der Trace-Daten kann man mit einem hohen Datenaufkommen rechnen, da eine Erzeugung redundanter Trace-Daten nicht immer vermeidbar ist. Darum soll bereits bei der Erfassung eine Vorauswahl der Daten durch einen programmierbaren Adapter für Trace-Schnittstellen erfolgen. Für den Adapter werden Selektions- und Kompressionsalgorithmen entwickelt, die eine hohe Transferrate zwischen Trace-Host und Trace-Target erlauben. Die so reduzierten und komprimierten Trace-Daten werden beim Host in die Datenbasis eingefügt. Es ist möglich diesen Prozess der (Re-)Konfiguration, Erzeugung und Speicherung wiederholt durchzuführen, bis alle notwendigen Daten erfasst wurden.

Zur Rekonstruktion der statischen Daten wird ein Simulator verwendet, der als Eingabe die Trace-Daten aus der Datenbasis verwendet und die rekonstruierten Daten wieder in die Datenbasis einfügt. So kann Schritt für Schritt der Kontext aus dem erfassten Trace-Zeitraum rekonstruiert werden. Da sich die für Analysator und Simulator notwendigen Informationen überschneiden, werden beide Einheiten aus einer gemeinsamen Hardware-Beschreibung generiert. Diese Beschreibung und zwei Templates dienen dann als Grundlage für die Erzeugung der beiden Einheiten (s. Abb. 3). Dies ist eine wichtige Voraussetzung für die zeitnahe Anpassung des Entwicklungswerkzeuges an neue und experimentelle Architekturen.

Die formal beschriebenen Analyseaufgaben enthalten auch die Auswertefunktionen für die Kontext-Daten. Da die Auswertefunktionen dem Optimierungszweck dienen, braucht man verschiedene Implementierungsvarianten. Grundlegend kann die Auswertung über Wahrheitswerte erfolgen, so z.B. als Laufzeit-Zusicherungen für Variablen-Werte. Aber auch eine komplexere Logik oder Funktionen einer Bibliothek für Programm-Visualisierungen sind möglich. So könnten Trace-Daten z.B. als Kontrollfluss-Baum, Histogramm oder Programm-Überdeckung dargestellt werden.

Anwendungen

Am Anfang sind Anwendungen aus den Bereichen der Performance- und Energieeffizienz-analyse oder der Programm-Verifikation geplant. Die Infrastruktur der Parametrierung und Rekonstruktion ermöglicht aber auch weitere Anwendungen, wie zum Beispiel das Debugging nicht-unterbrechbarer Systeme oder stochastische Analysen der Trace- und Kontext-Daten.

So werden bei Performance- und Energieeffizienzanalysen Eigenschaften wie Häufigkeiten für Sprünge, Cache-Hit/Miss-Raten pro Programm-Zeile usw. ermittelt, mit denen die Auslastung der Ressourcen und die Qualität der Optimierung für jede Quellcode-Zeile überprüft werden

[1] Es werden zwei unterschiedlich strukturierte Datenbasen für die Rekonstruktionsdaten und die Trace-Daten benötigt. Für den Zweck der Beschreibung werden sie hier jedoch als eine abstrakte Datenbasis zusammengefasst.

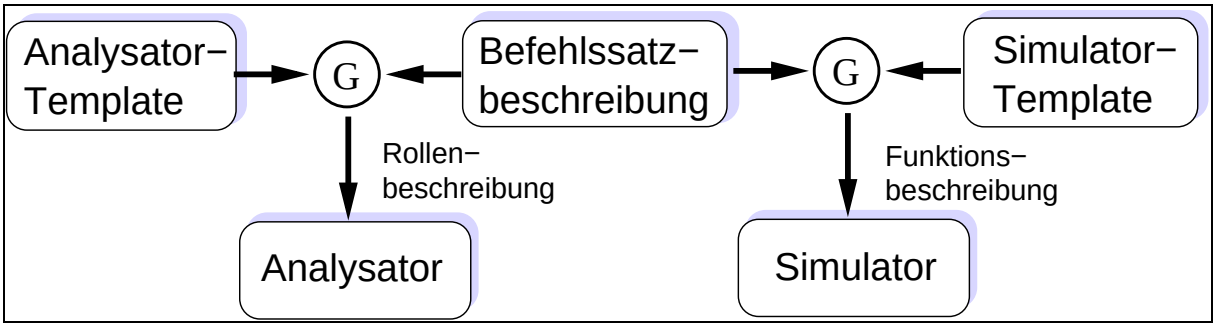


Abbildung : Erzeugung von Simulator und Analysator aus einer Befehlssatzbeschreibung und Templates

können. Der Entwickler erhält damit wichtige Kenngrößen für Entscheidungen über die Programm-Struktur und den Erfolg von Compiler-Optimierungen.

Für die Programm-Verifikation können Informationen wie die Ausführungsdauer eines Interrupt-Handlers, die Signal-Antwort-Zeit einer Applikation, die maximale Stack-Größe, die Einhaltung von Echtzeitbedingungen oder Aussagen über Programm-Invariante ermittelt werden.

Die Rekonstruktion ermöglicht bereits die Navigation innerhalb des lokal ermittelten Kontexts mit normalen Debug-Funktionen. Mit Hilfe des integrierten Simulators, können auf diese Weise zusätzlich Einzelschritte rückwärts, sowie Speicher- und Register-Manipulation auf dem Host simuliert werden. Dies ist eine wichtige Anwendung vor allem für nicht-unterbrechbare Systeme.

Die gesammelten Trace-Daten und die rekonstruierten Kontext-Daten in der Datenbasis bieten eine interessante Grundlage für automatisches Debugging. Der Vergleich von verschieden Trace-Ergebnissen einer Analyseaufgabe kann z.B. Informationen zur Fehlereingrenzung liefern oder Auskunft über Programm-inhärente Fehler geben.

Zusammenfassung

Der vorgestellte Arbeitsablauf beginnt mit einer formellen Beschreibung einer Analyseaufgabe. Aus der Beschreibung wird durch Analyse ermittelt, welche Informationen per Hardware-Trace gewonnen werden müssen, und welche Informationen durch Rekonstruktion gewonnen werden können. Das Ergebnis der Analyse ist eine weitgehend optimale Parametrierung der Trace-Hardware. Aus den erhaltenen Trace-Daten wird der Software-Kontext rekonstruiert auf dem die Analyse durchgeführt wird. Die Ergebnisse der Analyse werden dann je nach Aufgabenstellung visualisiert. Die erzeugte Datenbasis ist aber auch für weitere Analysen nutzbar.

Dieses, zugegeben, ambitionierte Konzept soll das Potenzial von Hardware-Trace in eingebetteten Systemen einem breiten Entwicklerkreis zugänglich machen. Sein Ziel ist die Automatisierung von Routineaufgaben und komplexen Analysen im Bereich des Hardware-Trace. Die Komponenten sind für die Einbindung in einen iterativen Entwicklungsprozess entworfen. Diese Technik ist geeignet um sowohl in frühen Phasen des Systementwurfs, als auch im Reife-Prozess des Systems eingesetzt zu werden.

Literatur

[1] WEIßE, STEFAN; GAJDA, ANDREAS; *„Increasing the System Performance by Code Optimization using On-Chip Emulator“*. In: Proceedings of the Embedded World Conference; März 2010;

[2] DITU, BOGDAN; *„Automatic Optimizations based on Profile Information“* . In: Proceedings of the Embedded World Conference; März 2010;

[3] GUANGQUAN, ZHANG; MEI, RONG: *„An Approach of Concurrent Object-oriented Program Slicing Based on LTL Property“*. In: Proceedings of the International Conference on Computer Engineering and Software Engineering; Wuhan; 2008

[4] GRAF, PHILLIP; MÜLLER-GLASER, KLAUS D., REICHMANN, CLEMENS; *„Nonintrusive Black- and White-Box Testing of embedded Systems Software against UML-Models“*. In: 18th International Workshop on Rapid System Prototyping (RSP'07);

[5] JIANG, LINGXIAO; SU, ZHENDONG; *„Context-aware statistical debugging: from bug predictors to faulty control flow paths“*; In: ASE `07 Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering; Atlanta 2007; S.184-193

[6] HUANG, MICHAEL; RENAU, JOSE, TORELLAS, JOSEP; *„Profile-based Energy-Reduction for High-Performance Processors“*, 2001

[7] SMITH, MICHAEL D.; *„Overcoming the Challenges to Feedback-Directed Optimization“*; In: Proceedings of the ACM SIGPLAN Workshop on Dynamics and Adaptive Compilation and Optimization (Dynamo‘00); Boston; 2000

Fehleranalyse mittels Trace-basierter Rekonstruktion von Prozessorzuständen

Steffen Köhler, Andreas Gajda, Rainer G. Spallek

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

Kurzfassung

Zur erweiterten Systemanalyse in komplexen, eingebetteten Prozessoren ist es oft unumgänglich, Programmzustandsinformationen über bereits ausgeführte Codeabschnitte zu ermitteln, um mit deren Hilfe Laufzeitverhalten, Energieffizienz und Fehlerszenarien besser rekonstruieren zu können. Diese Rekonstruktion basiert dabei auf einer Aufzeichnung wesentlicher Merkmale des abgearbeiteten Befehls- und Datenstromes, die mittels spezieller On-Chip Trace-Hardware erfasst wurden. Da die anfallenden Trace-Daten bei hohen Taktraten aufgrund der begrenzten Verarbeitungs- und Ausgabekapazität der Trace-Hardware nicht vollständig erfassbar sind, andererseits deren Vollständigkeit durch Nutzung von Regularität des Befehls- und Datenstromes auch nicht zwingend notwendig ist, müssen fehlende Trace-Daten unter Beachtung der im Befehlsstrom enthaltenden Zusatzinformationen ergänzt und komplettiert werden. Unter *Back-Trace-Analyse* wird in diesem Zusammenhang die Rekonstruktion des relevanten Prozessorzustandes verstanden, der zu einem vergangenen Zeitpunkt der Programmausführung vorgelegen hat. Im Gegensatz zu bisherigen Ansätzen, die lediglich die Ermittlung von Stack-Informationen unterstützen, betrachten die die hier dargestellten Methoden und Techniken den kompletten Prozessorkontext. Die Vorstellung der wesentliche Komponenten eines solchen Systems zur Rekonstruktion von Prozessorzuständen ist Gegenstand dieses Beitrags. Dabei werden sowohl analytische Betrachtungen zu relevanten Daten- und Befehlsflüssen als auch die eigentliche iterativ simulative Rekonstruktion der Prozessorkontexte dargestellt.

1 Einleitung

Laufzeitanalysen an Hardware-/Software-Systemen sind von entscheidender Bedeutung, wenn Fehler- und Effizienzanalysen in eingebetteten Systemen untersucht werden sollen. Die Nachvollziehbarkeit von Interaktionen zwischen einer großen Anzahl an Komponenten innerhalb komplexer System-on-a-Chip ist ohne die Nutzung und Rekonstruktion zur Laufzeit erfasster relevanter Systemzustände kaum möglich. Der vorliegende Beitrag befasst sich mit dem Teilproblem der Rekonstruktion von Zuständen eingebetteter Prozessorkerne, deren zunehmende Anzahl auf einem Chip eine wachsende Herausforderung für den Softwareentwicklungsprozess darstellt. Die Zeitstellung besteht darin, Programmzustandsinformationen über ausgeführte Codeabschnitte zu ermitteln, um mit deren Hilfe Laufzeitverhalten und Fehlerszenarien rekonstruieren zu können. Diese Rekonstruktion basiert dabei auf einer Aufzeichnung wesentlicher Merkmale des abgearbeiteten Befehls- und Datenstromes, die mittels Trace-Hardware [1] aufgezeichnet werden. Da

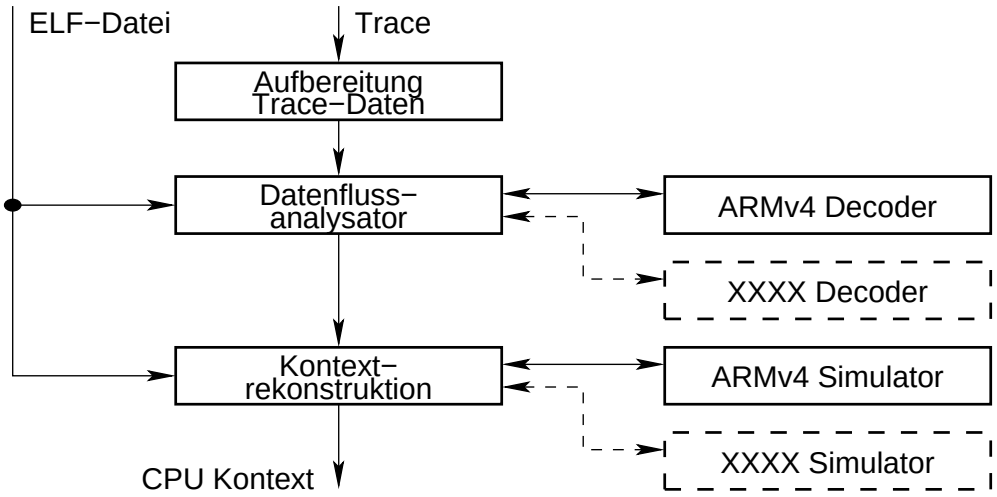


Abbildung 1: Überblick zur Back-Trace-Analyse

die Trace-Daten einerseits aufgrund der begrenzten Kapazität der Trace-Hardware im Bezug auf die Menge der Prozessorzustandsinformationen nicht vollständig erfassbar sind, andererseits deren Vollständigkeit durch Nutzung von Regularität des Befehls- und Datenstromes auch nicht zwingend notwendig ist, müssen fehlende Trace-Daten unter Beachtung der im Befehlsstrom enthaltenden Zusatzinformationen ergänzt und komplettiert werden. In einem zweiten Schritt sind die Trace-Daten dann wieder in einen Kontext mit dem ausgeführten Programmquelltext zu setzen, um das Ausführungsverhalten zu rekapitulieren. Unter *Back-Trace-Analyse* wird in diesem Zusammenhang die Rekonstruktion des relevanten Prozessorzustandes (Register, Speicher) verstanden, der zu einem vergangenen Zeitpunkt der Programmausführung vorgelegen hat. Das angewandte Rekonstruktionsprinzip kann dabei als eine Art Trace-basierte Simulation angesehen werden. Der vorliegende Beitrag beschreibt wesentliche Komponenten, die zur schrittweisen Umsetzung dieses Prinzips notwendig sind und die prototypisch realisiert wurden:

- Trace-Aufbereitung
- Befehlsdecoder
- Datenflussanalysator
- Befehlssatz-Simulator

Die strukturelle Anordnung dieser Komponenten ist in Abbildung 1 dargestellt. Das dargestellte Framework ist - mit Ausnahme des Befehlssatz-Simulators - in der Programmiersprache C++ verfasst.

2 Trace-Aufbereitung

Je nach Verfügbarkeit verschiedener Trace-Spezifikationsmöglichkeiten der Trace-Hardware können folgende Daten für die Rekonstruktion von Kontextdaten herangezogen werden:

IP-Trace bezeichnet den realen Programmfluss anhand des Befehlszählers (Instruction Pointer)

LS-Trace umfasst alle Load/Store Operationen inklusive der zugehörigen Adressen und Daten

Obwohl das Vorliegen des vollständigen Programmflusses eine notwendige Voraussetzung zur Rekonstruktion von Kontextdaten darstellt, kann die damit assoziierte Datenmenge auf die ausgeführten Sprungbefehle beschränkt werden. Da der zwischen Sprungbefehlen ausgeführte Programmcode linear abgearbeitet wird (Basisblöcke), ist dieser Code für den Programmfluss nicht

relevant und muss nicht aufgezeichnet werden.
In einer ersten Vorverarbeitungsstufe wird daher der IP-Trace auf die relevanten Basisblockinformationen reduziert und aus Gründen eines schnellen sequentiellen Zugriffes als Graph abgespeichert. Darüber hinaus sind Informationen über nicht ausgeführte Transfer- und Arithmetikoperationen ebenfalls Bestandteil der Basisblöcke.
Die Vorverarbeitung des IP-Trace kann durch gezielte Parametrierung der Trace-Hardware erleichtert werden, indem nur Sprungbefehle und nicht ausgeführte Transfer- und Arithmetikoperationen aufgezeichnet werden. Da die Mehrheit der ausgeführten Sprungbefehle ein festes, statisch analysierbares Sprungziel besitzt, kann bei deren Aufzeichnung das Sprungziel ignoriert werden, womit eine weitere Reduktion des Trace-Datenstromes erreicht werden kann. Lediglich nicht statisch analysierbare Sprungziele einiger relativ selten ausgeführter Sprungbefehle (z.B. *return*, *switch*) müssen inklusive Adressdaten aufgezeichnet werden.
Im Gegensatz zum IP-Trace ist die Notwendigkeit des Vorliegens eines LS-Trace für die Kontextrekonstruktion stark applikationsspezifisch. Das Fehlen von LS-Trace-Informationen ist dabei gleichbedeutend mit der Unvollständigkeit von Kontextinformationen. Prinzipiell sind Reduktionstechniken wie das Ignorieren von Ladeoperationen von Konstanten aus dem Speicher zwar möglich, jedoch treten diese im dynamischen Programmfluss relativ selten auf (typisch $< 5\%$). Als weiteres Problem muss die Notwendigkeit zur Aufzeichnung von Zugriffsadressen und Daten beim LS-Trace genannt werden, deren Datenvolumen das des IP-Trace selbst bei Kompression der Adressinformation deutlich übersteigt. Das folgende Beispiel soll die Menge der zu erwartenden Trace-Daten bei einer durchschnittlichen Programmlast (EEMBC Industrial/Automotive Benchmark [2]) verdeutlichen:

- Bei einer Sprunghäufigkeit von $10\ldots15\%$ und einer komprimierten Kodierung von 1 Trace-Byte pro Sprungbefehle ist beim IP-Trace mit einem durchschnittlichen Trace-Datenvolumen von $0.1\ldots0.15$ Bytes pro Prozessortakt zu rechnen.

- Bei einer Ausführungshäufigkeit von $10\ldots15\%$ und einer nur minimal komprimierbaren Kodierung von 7 Byte pro LS-Operation (1 Byte Header, 2 Bytes Adresse, 4 Bytes Daten) kann beim LS-Trace von einer durchschnittlichen Trace-Datenrate von $0.7\ldots1.0$ Bytes pro Prozessortakt ausgegangen werden.

- Für einen eingebetteten Prozessor mit 100 MHz Taktfrequenz bedeutet dies ein Trace-Volumen von mehr als 100 MByte pro Sekunde. Soll der Trace-Strom über eine im Gegensatz zu dedizierten Trace-Ports vergleichsweise schmalbandige JTAG-Schnittstelle [4] (3 MBytes pro Sekunde) ausgegeben werden und besitzt der auf dem Chip integrierte Trace-Buffer eine Kapazität von 8 KByte, so beträgt das Aufzeichnungsintervall ca. 80 μs. Nur die innerhalb dieser Zeitspanne aufgezeichneten Daten stehen dabei für die Rekonstruktion aller relevanten Kontextinformationen zur Verfügung, frühere/spätere Trace-Daten können nicht herangezogen werden.

Da wie bereits erwähnt die Rekonstruktion der Kontextinformationen durch Simulation durchgeführt werden soll, müssen die zu diskreten Zeitpunkten erfassten Trace-Daten auf deren Lebensspannen (Gültigkeitsintervalle) expandiert werden. Anschließendes Ziel der Simulation ist es, die fehlenden Daten der Gültigkeitintervalle basierend auf den gegebenen Lebensspannen durch erneute Ausführung des Programmcodes im Simulator zu ergänzen. Die Ermittlung der Lebensspannen erfolgt dabei durch den Datenflussanalysator, der Gegenstand des folgenden Abschnitts ist.

```
R00: 4-5,7-71,49232-49233,49233-49241,49241-49249,49249-49257
R01: 31-41,41-51,51-57,57-59,49237-49239,49245-49247
R02: 11-12,12-13,13-14,19-20,20-21,37-44,46-48,48-54,54-55
R03: 8-9,9-10,15-16,16-17,17-18,22-23,23-24,24-25,26-27
...
R12: 39-42,42-44,49231-71071,71088-84184,84201-93553
R13: 2-6,6-62,62-66,66-2682880,2682880-2682881
R14: 3-6,33-36,36-38,49223-71078,71078-84191,84191-93560
FLAGS: 84-85,90-91,96-97,102-103,108-109,114-115,120-121
```

Abbildung 2: Intervallvektoren der Datenflussanalyse

3 Datenflussanalysator

Der eingesetzte Datenflussanalysator führt eine Low-Level Analyse der Lebensspannen von Speicher-/Registerinhalten durch. Die eingesetzten Prinzipien orientieren sich dabei an den typischerweise in Compiler-Backends eingesetzten Techniken [3].
Ein Speicher-/Registerinhalt *r* ist dabei lebendig an einem Programmzeitpunkt *p*, wenn es auf einem Programmpfad nach *p* eine Setzung von *r* gibt und ein Pfad von *p* zu einer Benutzung von *r*, auf dem *r* nicht gesetzt wird. Die Lebensspanne eines Speicher-/Registerinhaltes *r* ist die Menge der Programmzeitpunkte, an denen *r* lebendig ist. Ein Speicher-/Registerinhalt ist also an einem Programmpunkt lebendig, wenn sein dortiger Inhalt noch benötigt werden kann.
Zur detaillierten Analyse der Lebensspannen müssen die vom Trace-Strom abgedeckten Befehle bzgl. deren Datenfluss (Quell- und Zieloperanden) untersucht werden. Dazu ist ein spezieller Befehlsdecoder (Abschnitt 4) erforderlich, der die Menge der datenflussrelevanten Operanden in separaten Listen verwaltet. Um die Lebensspannen der Register- und Speicherinhalte zu ermitteln, wird der Programmfluss schrittweise in einer Bottom-Up-Strategie durchlaufen. Die Lebensspanne von Register- und Speicherinhalte werden damit invers, also vom Punkt ihrer letzten Benutzung bis zu ihrer Setzung verfolgt.
Die Speicherung der Lebensspannen erfolgt in adressspezifischen, sortierten Intervallvektoren, da diese eine wahlfreie Zugriffsmöglichkeit auf Einzelelemente in konstanter Zeit $O(1)$ ermöglichen. Das Auffinden von Register- und Speicherbelegungen in Intervallvektoren kann somit als binäre Suche in $O(\log n)$ Schritten effizient ausgeführt werden. Dieser Fakt ist von großer Bedeutung, da typische Trace-Muster bis zu mehreren Millionen Lebensspannen mit bis zu einigen Millionen Taktzyklen enthalten können. Abbildung 2 verdeutlicht die Verwaltung von Lebensspannen in Form von Intervallvektoren.
Als Basis für die Ermittlung der Intervalle fungiert die mit 64 Bit Auflösung durchgeführte Enumeration des Befehlssquenz ausgehend vom Wert 1 für den ersten Befehl der Trace-Aufzeichnung. Der Intervallstartwert 0 stellt als Sonderfall das Lesen von Register- und Speicherinhalten dar, die zu Beginn der Trace-Aufzeichnung nicht gesetzt sind. Dies ist häufig bei konservativer Registerrettung zum Programmstart der Fall.

4 Befehlsdecoder

Der Befehlsdecoder wird vom Datenflussanalysator zur Ermittlung von Quell- und Zieloperanden benötigt. Die Trennung von Datenflussanalysator und Befehlsdecoder erfolgte mit der Zielstellung, die Back-Trace-Analyse auf verschiedenen Prozessorarchitekturen anwenden zu können. Es ist geplant, zukünftig den Befehlsdecoder aus der Befehlssatzbeschreibung des Befehlssatz-Simulators automatisch zu generieren.

```
380:   e5c35000        strb    r5, [r3]
384:   e1c040b2        strh    r4, [r0, #2]
390:   e249903e        sub     r9, r9, #62
398:   e1a0b005        mov     fp, r5
3a0:   e7c3b009        strb    fp, [r3, r9]
```

Abbildung 3: Beispielcode zur inversen Rekonstruktion

5 Befehlssatz-Simulator

Da die Expansion der LS-Trace-Daten nur für einen kleinen Teil der Lebensspannen mit direktem LS-Bezug möglich ist, müssen die restlichen Daten (insbesondere Registerinhalte, die nicht mittels Trace auslesbar sind) durch Simulation rekonstruiert werden. Die simulative Vervollständigung der Gültigkeitsintervalle geht dabei iterativ vor, indem zunächst alle Zieloperanden der Befehle errechnet werden, deren Quelloperanden bereits bekannt sind. Zu Beginn der Rekonstruktion sind dies vor allem direkt auf LS-Trace-Daten basierende oder konstante Quelloperanden (Immediates oder im Datenspeicher abgelegte Konstanten). Die erhaltenen Zieloperanden werden zur Komplettierung der Intervalldaten herangezogen und sind für nachfolgende Simulationsschritte nutzbar. Dieser Vorgang wird so lange wiederholt, bis keine neuen Intervalldaten mehr errechnet werden können.
Die iterative simulative Rekonstruktion ist besonders vorteilhaft auf Lebensspannen anwendbar, deren Länge wesentlich kürzer als die Länge der Trace-Aufzeichnung ist. Übersteigt die Länge von Lebensspannen die Länge der Trace-Aufzeichnung (z.B. Schleifenzähler, Adressregister), so sind zur Errechnung relevanter Daten aus den innerhalb der Trace-Aufzeichnung erfassten Schreib-/Lesezugriffe zusätzlich inverse Rekonstruktionstechniken unumgänglich.
Die rückwärtsgerichtete Berechnung von Intervalldaten ist für einfache Operationen wie Register-Kopieroperationen, Additionen und Subtraktionen sowie eine große Anzahl von Adressrechnungen problemlos möglich. Beispielhaft soll hier die inverse Adressrechnung genannt werden, die durch den in Abbildung 3 gezeigten Programmcode illustriert wird. Das Adressregister $r0$ lässt sich beispielsweise aus dem LS-Address-Trace an Adresse 384 durch Subtraktion des Offset von 2 ermitteln, während der Wert von $r9$ ab Adresse 394 aus den LS-Adress-Traces an den Adressen 380 und 3a0 ebenfalls bestimmbar ist. Der Wert von $r9$ vor dem Durchlaufen der Adresse 390 kann durch Invertierung der Subtraktion an Adresse 390 ebenfalls ermittelt werden. Die Häufigkeit der inversen Rekonstruktion ist entsprechend den in Abbildung 4 exemplarisch gezeigten Lebensspannen relativ niedrig, da nur wenige lange Lebensspannen existieren.
Die Ausführung der inversen Operationen ist als ein Teil des iterativen Rekonstruktionszyklus anzusehen, wobei Vorwärts- und Rückwärtsrekonstruktionen solange durchzuführen sind, bis keine neuen Intervalldaten mehr bestimmt werden können.
Als Simulator wird das *JAHRIS* Framework [5] verwendet, welches über die JNI-Schnittstelle angesteuert wird.

6 Zusammenfassung

Die vorgestellten Komponenten zur Realisierung einer Back-Trace-Analyse sind als experimentelle Prototyplösung für die spätere Integration in die Debug-Umgebung UDE [6] entwickelt worden. Bisherige Untersuchungen wurden anhand der ARM v4/v5 Architektur durchgeführt, für die sowohl Decoder- als auch Simulationsmodelle erstellt wurden. Die bisher für die Konzeptevaluation herangezogenen Trace-Daten wurden ausschließlich durch Simulationen generiert, da passende ARM Prozessoren mit entsprechender Trace-Hardware (ETM v2 oder neuer bzw. Core-

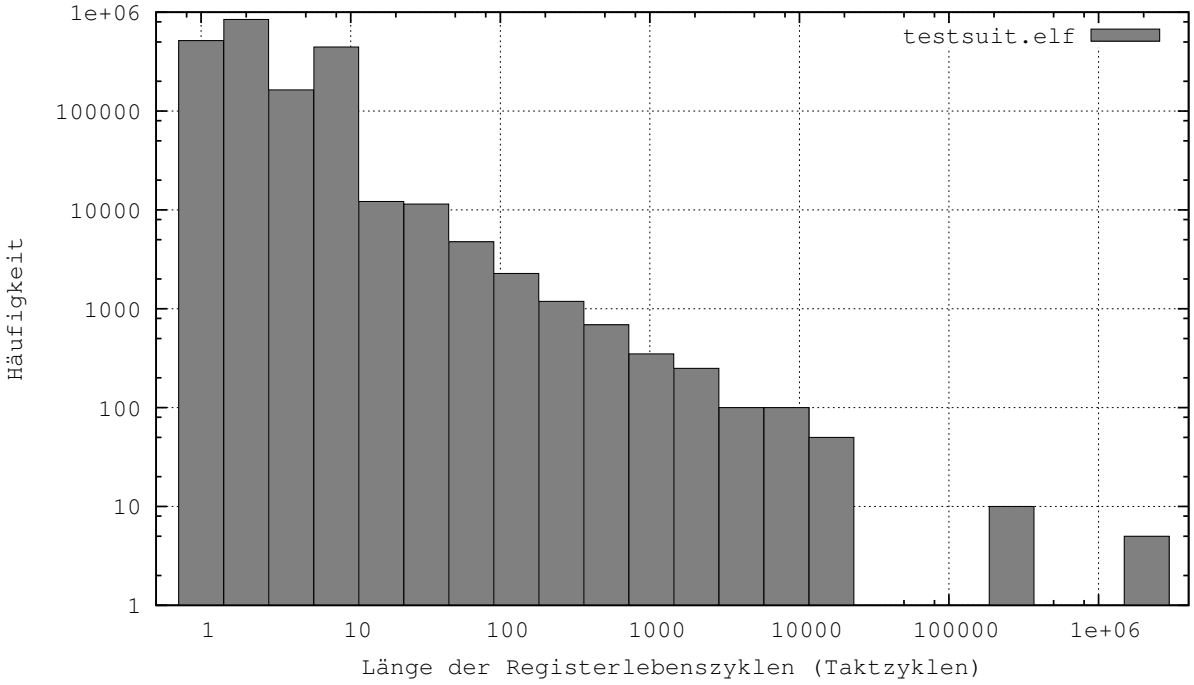


Abbildung 4: Häufigkeit von Lebensspannen in Abhängigkeit ihrer Länge

Sight) nicht zur Verfügung standen. Es wurden jedoch Konverter entwickelt, die die Nutzung der von UDE erfassten Trace-Daten durch die Prototypumgebung ermöglichen.
Der Rekonstruktionsgrad der Kontextdaten ist generell sehr applikationsspezifisch und hängt von der Verfügbarkeit verwertbarer LS-Trace-Profile - insbesondere der Speicherzugriffsadressen - ab. Repräsentative Aussagen über die Rekonstruierbarkeit von Kontextdaten relevanter Applikationen unter Zuhilfenahme realer Trace-Ströme sind Gegenstand noch durchzuführender Untersuchungen.

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] Dieter Maurer, Reinhard Wilhelm. *Übersetzerbau - Theorie, Konstruktion, Generierung*. Springer Verlag, 1992.

[4] IEEE Computer Society, Test Technology Technical Committee. *Standard Test Access Port and Boundary-Scan Architecture*, Januar 1990.

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

[6] Stefan Weiße. Flexible Plattform für applikationsspezifische Entwicklungsumgebungen. 2000.

VLSI Implementation of a Heap Sorting Algorithm for Time Based Event Communication

Stefan Scholze, Stephan Henker, Johannes Partzsch, Christian Mayr, Rene Schüffny
{scholze, henker, partzsch, mayr, schueffn}@iee.et.tu-dresden.de
Institut für Grundlagen der Elektrotechnik und Elektronik, TU Dresden

Abstract

Large-scale hardware systems for spiking neural networks require high-speed communication for transmitting so-called spike events between units of the neural network. Usually, a packet network is used for this spike transmission, which efficiently exploits the limited bandwidth of the single transmission channels, while in addition offering flexible link configuration. The drawback of such a network is the unpredictable delay of individual packets, contrasting with the constant delay between units usually required in neural networks. To overcome this, we have developed a timestamp-based sorting and buffering algorithm, ordering the packets in a tree-like sorted heap structure. The measured performance of the heap sort achieves a competitive compromise between the different solutions in literature, combining low latency and high throughput required for the pulse routing application with the power and size reduction needed for a high integration density of the complete system.

1 Introduction

Hardware implementations of spiking neural networks benefit from accelerated operation, which inreases the integration density. However, this approach requires high-speed communication of spike events between neuron circuits. Such an accelerated, large-scale hardware system for spiking neural networks is currently implemented in the FACETS project [1]. This system employs waferscale integration to achieve a high integration density [2]. For realising spike communication between wafer modules, a packet network has been implemented, which efficiently copes with a large number of sources and targets. This so-called layer 2 network consists of custom-designed Digital Network Chips (DNC) that realise the communication with the High Input Count Analog Neural Network (HICANN) units on the wafer and high-performance Field Programmable Gate Arrays (FPGA) for implementing the inter-wafer communication [1].

For realising spike connections with a constant delay (required for spiking neural networks [3]), spike packets have to be buffered and sorted at the target network node, which is the DNC in the FACETS system. We have developed a high-performance, combined sorting and buffering unit for this purpose. Despite its special application, it compares favourably with current more general VLSI sorting implementations. In section 2, we derive the constraints for the sorting algorithm, which is then introduced in section 3. Section 4 describes the hardware implementation of the algorithm and compares it to state-of-the-art VLSI sorting implementations.

2 Network requirements

In an accelerated system such as the FACETS system, transmission delays are not negligible. Biological delays for long-range connections between spiking neurons are in the order of 5-10ms [4] and assumed to have a relatively low jitter. Taking into account the acceleration factor of 10^4 for the FACETS hardware, this corresponds to a connection delay of 500ns to 1μs. For comparison, the pure transmission delay for a connection between wafers (HICANN-DNC-FPGA-FPGA-DNC-HICANN) is approx. 530ns. Thus, processing of spike events at the network nodes requires a latency as small as possible in order to reproduce biologically realistic delays.

As a second requirement, the network communication has to provide a very high bandwidth. Each HICANN may receive pulses from 512 neurons transmitted over the layer 2 communication network. If each of these neurons pulses at 10 Hz, which is a typical pulse rate in biological

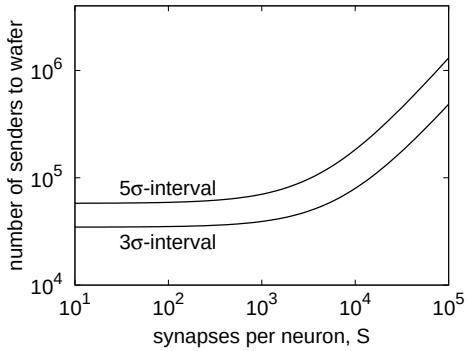


Figure 1: Number of external spike sources to a single wafer for a locally coupled network. The 5σ-interval curve represents a more complete mapping of the network to the hardware.

time, the interface has to provide a throughput of 51.2Mpulses/s. In the current implementation, 2 pulses are transmitted in a 64bit packet, resulting in a data rate of 1.6Gbit/s. This is close to the maximum data rate of 2Gbit/s provided by the DNC-HICANN LVDS channel.

With 512 external neurons per HICANN and 400 HICANNs on a wafer, 200k neurons can send to a wafer. This number is actually required for implementing realistic network models, as we show in the following for locally coupled networks [5]. In such networks, neurons are arranged in a two-dimensional space and the probability p of a connection between two neurons decreases with their Euklidean distance d:

$$p(d) = \mathrm{e}^{-\frac{d^2}{2\sigma^2}} \ , \tag{1}$$

where σ determines the spread of the connectivity. These networks are scalable on a two-dimensional system such as the FACETS hardware [6]. Thus, the bandwidth requirements for pulse communication between wafers are only determined by σ, which is directly proportional to the number of synapses S per neuron. Figure 1 shows the number of external spike sources with respect to S for a fully utilized wafer. As is shown in the plot, the available number of external spike sources is well utilized for realistic S values (10^3 to 10^4). Other state-of-the-art network models may even be more demanding in terms of inter-wafer communication, see [6].

For maximally utilizing the available bandwidth, each pulse packet is transmitted as fast as possible over the routing network. With the target time of the pulse event stored in the packet, pulses can be delayed at the final network node (DNC) to ensure a constant delay. This approach is advantageous compared to establishing connections with a-priori constant delay, because these have to reserve bandwidth for each spiking neuron [3]. Due to the irregular, unpredictable spiking of neurons (see e.g. [7]), this would lead to low bandwidth utilization, which is prohibitive for our application as discussed above.

While delaying at the target node (DNC) is advantageous for bandwidth utilization, it introduces a source of congestion: An event with a late target time may block other events with earlier target times. Thus, events have to be sorted by their target time to ensure the earliest event is released next. This sorting is very demanding, because it has to meet the throughput requirements of the DNC-HICANN channel. Furthermore, it should operate with low latency, as discussed above. Finally, it should be implementable with moderate hardware resources. We have developed a sorting and buffering unit that fulfills these requirements. We describe its algorithm and implementation in the next sections.

3 Heap Sorting Algorithm

As described in the previous section, a sorting algorithm is needed in the Digital Network Chip (DNC) of the FACETS system that reorders and buffers incoming pulse event packets. The ordering is done according to the upper 10 bit of a 15 bit timestamp contained in the pulse event packet

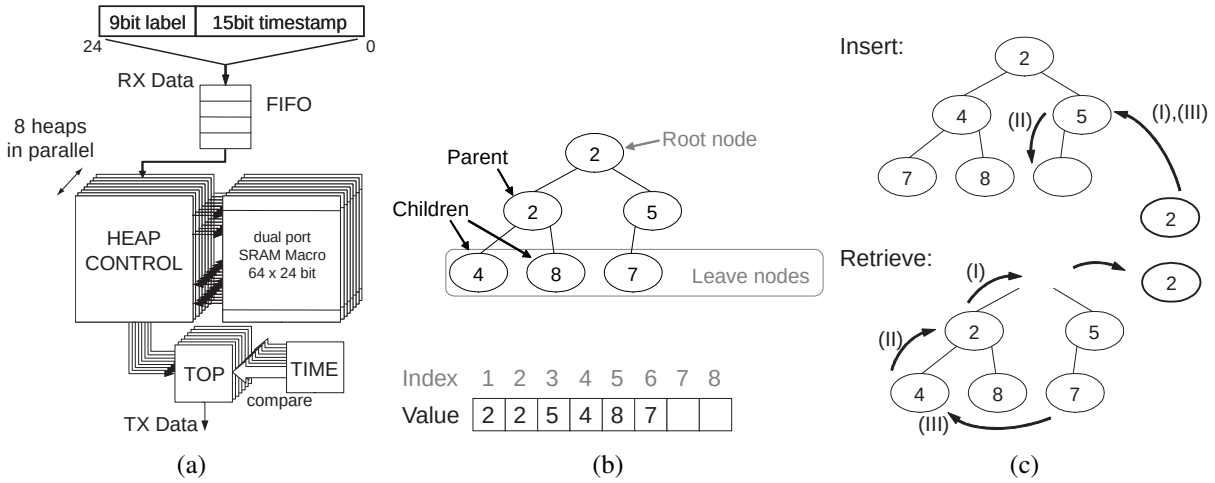


Figure 2: (a) block diagram of one channel of the system using the heap, (b) binary heap tree and the projection to an array, (c) standard insert and retrieve operations on a heap

(see Fig. 2a), determining the time at which the packet should be sent to the target HICANN (the lower 5 bit are processed on the HICANN).

A simplified data flow is shown in Fig. 2a. A single input FIFO is used to buffer incoming event packets. This is required to handle input burst access. As the output interfaces work autonomously, each output implements an own instance of the sorting module, basically consisting of a memory and a heap control. The next pending event of an interface is buffered in a separate register (TOP) to allow instantaneous access. This is necessary, as this packet needs to be transmitted immediately when it's delay expires and accessing the heap is not always possible within one clock cycle (i.e. the heap control is blocked while executing an insert operation). When the heap control takes an element from the FIFO, it is first compared to the top element. When the top element has a larger timestamp, the new element and the current top element are exchanged. The remaining element is then inserted to the heap. Once the top element is sent, the heap control extracts the next pending element from the memory.

A heap structure (Fig. 2b) was chosen because it offers interesting properties that can be efficiently exploited in hardware: First, a heap is only weakly ordered, i.e. the key (timestamp) of a parent node is ensured to be smaller or equal only compared to each of it's child nodes. In contrast, entries in different branches of the underlying tree may exist in arbitrary order. This property simplifies insertion and retrieval of elements. In general, insertion of a new and the removal of the top element require logarithmic time at constant resource needs. The second advantage is the possibility to use SRAM macros. Implemented as binary heap, where each parent node has a maximum of two child nodes, insertion and removal operation need access to at most two array elements in parallel, thus implying the usage of dual-port SRAM which is available as highly integrated memory macro. The heap is implemented as min heap with smallest element of the tree as root node. The tree itself is projected to a memory array $M(1:n)$, with address range $j \in [1,n]$. The root element is $M(1)$, and the two children $K_1(j)$ and $K_2(j)$ of an element $M(j)$ in a heap are $M(2j)$ and $M(2j+1)$. The heap property constraints ensure direct access to the smallest element in the heap and a very efficient hardware for address calculation as the calculation of both child node addresses can be implemented simply by using a shifter and an incrementer. There are two main operations to be implemented in hardware: the insertion of an element at the correct position and the removal of the top element with the subsequent reordering of heap tree (Fig. 2c).

Insert Element Fig. 3a presents the algorithm to insert an element to the heap. Initially, when there is no element in the memory, the counter c is equal to one and pointing to the first free position in M. The algorithm sets the variable j to c and then runs the search loop. In the empty

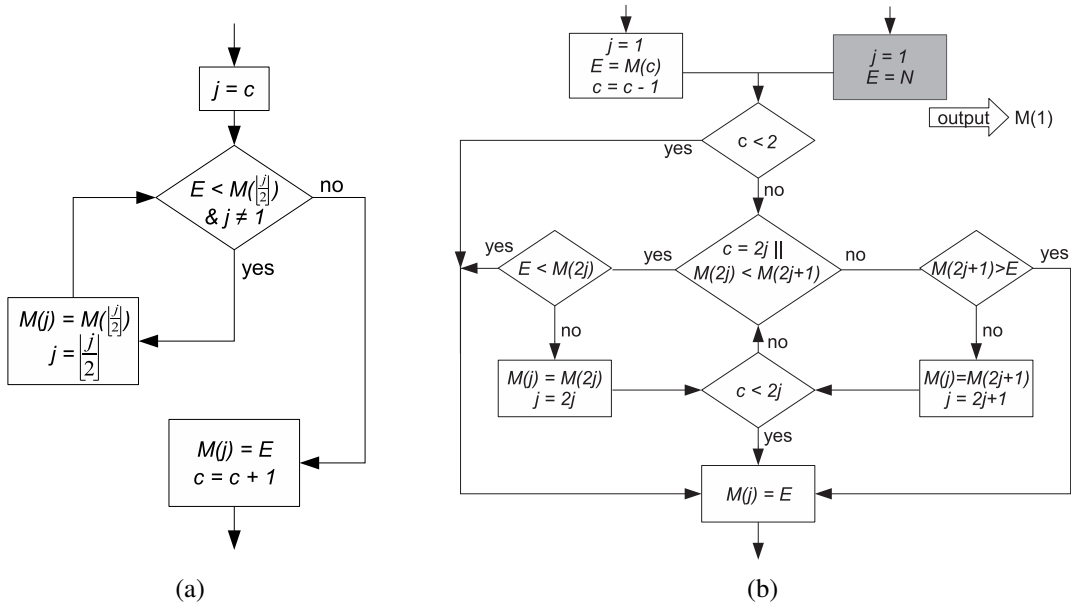


Figure 3: algorithms for inserting (a) and retrieving (b) an element to/from the heap (the grey entry combines both operations, i.e. an element is also inserted during retrieval)

case, where $j = 1$, the loop is immediately terminated and the new entry E is stored at position 1 and c is incremented. In all other cases, the parent element $M(j/2)$ of the current position j is compared with the input element. If it is smaller or equal than the new element, the loop terminates and the new element is stored at the current position j. Otherwise the parent element is larger than the new element, meaning the new element has to be inserted before it, thus the parent element is moved down in the heap by storing it at position j, then j is set to the parent index and the loop is restarted. As in general newly arriving events in the target application have a larger timestamp than previously received events, this inserting scheme requires less than the worst case number of ld(c) steps.

Retrieve Element The removal of the root node of the heap memory basically requires to rebuild the heap. One of the two child nodes of the removed root node has to be chosen as new root and this decision has to be repeated until the leaves. When reaching a leaf node, the heap may be incomplete because the reached leaf might not have been the last element and thus, the hole is left inside the array. In this case, the last element of the array is placed into the hole and percolated upwards like in the insert operation. This is the standard heap algorithm as found in e.g. the standard template library [8].

In Fig. 3b, a modified algorithm for removing an element and rebuilding the heap is shown. In case of an empty heap the algorithm is bypassed. When there are elements in the heap, the algorithm returns the root element, stores the last element (E) for later use and decrements the element counter c. The algorithm terminates immediately if the heap is now empty. If c indicates that there is only one element left in the heap, E is stored at the top node and the algorithm finishes. Otherwise in each iteration of the algorithm, the smallest element of both children of the current node j is selected and then compared with the element E. If the child node is smaller than E, then the child is placed at the current position j and j is set to the selected child position. The loop terminates if either j has reached a leaf node or when E is smaller than the selected child node. In all these cases, E is inserted at the position j where the loop has terminated. One additional check is necessary for the case that a parent only has one child, which is the case when $2 \cdot j$ is equal to the element count c. In this case the algorithm skips selecting the smallest child and directly compares E with the left child.

Special case: Insert and Retrieve Element During full operation of the DNC, the main action is a repeated insertion and retrieval operation, i.e. an event is released at its target time and

Table 1: Time consumption of the heap operations in clock cycles

Operation	Min Time	Max Time
Insert	3	14
Retrieve	2	14
Insert & Retrieve	2	14

Table 2: Comparison of the proposed heap sort with current state-of-the-art

Reference	Technology	Gate Count	Power	f_{clock}	Throughput	N_{cycles}	Keysize	Number of Keys
[9]	FPGA	2784 Slices	1.8W	133MHz	128keys/0.96μs	128	8 Bit	128
[10]	–	428	–	–	–	ca. 4000	16 Bit	64
[11]	130nm	1604261	71mW	143MHz	35.8Mkeys/s	–	12 Bit	4096
this design	180nm	593	1.3mW	250MHz	31Mkeys/s	2-14	10 Bit	64

a new event arrives at the FIFO. The subsequent application of retrieval and insert is not optimal as both operations need logarithmic time. An optimisation is possible by basically doing an insert and remove operation simultaneously. This can be implemented using an extended version of the retrieve algorithm, as presented in Fig. 3b (grey). For the special insert-retrieve operation, not the last element from the array but the new element is taken as E and the element counter is not decremented since the size of the heap is unchanged. The rest of the algorithm is the same for both cases. With this minor extension in the initialisation, the effort needed for the combined insert-retrieve process is reduced roughly by factor 2 compared to the sequential execution of retrieval and insert.

4 Hardware Implementation

The described algorithm is implemented in VLSI hardware using a UMC 180nm technology. The applied SRAM memory blocks offer 64 entries of 24 bit each and have a dual port interface. The dual port significantly reduce the clock cycles needed per algorithm loop. For example, the core of the retrieve algorithm requires three comparisons with three memory accesses. Using single port memory, it would require at least three clock cycles where first the two children are read subsequently from the memory, then compared and the smaller one written back to the parent position. In contrast, the dual port interface allows two read/write or a read and a write operation in parallel, thus when selecting the smaller of the two children, both can be read in parallel and compared, while in the next clock cycle the smaller child is read on one memory port and written to the memory on the other port. This operation additionally saves extra registers that would be necessary when using single port memory. Hence the core loop of the retrieve operation has been implemented using only two clock cycles. Table 1 presents the operations and their general time consumption in clock cycles.

The circuit was synthesised and has been constrained for 250MHz, which is the maximum possible clock frequency of the applied standard cell library. Fig. 4 shows a die photograph of the complete communication ASIC. Eight of the channels depicted in Fig. 2a were implemented, with each channel having eight of the heaps operating in parallel, i.e. a single SRAM block and its surrounding digital circuitry corresponds to one $64 * 24$ Bit heap and its control. The parameters of the final realisation are depicted in table 2.

To our knowledge, there is no related implementation of a sorting algorithm using SRAM memory. A comparison with three other sorting implementations is also given in table 2. When assuming that a single FPGA slice is at least equivalent to 5-7 Gates, our implementation is significantly less complex than [9]. When adjusting the throughput of [9] for the higher keysize and architectural differences of our implementation, an equivalent throughput of approx. 50Mkeys/sec can be estimated. The serial architecture of [11], with key size and number of keys identical, is slightly better in terms of complexity (i.e. number of gates), but is on average a factor of 800 slower. The large-scale parallel implementation of [11] has to be adjusted for the smaller number

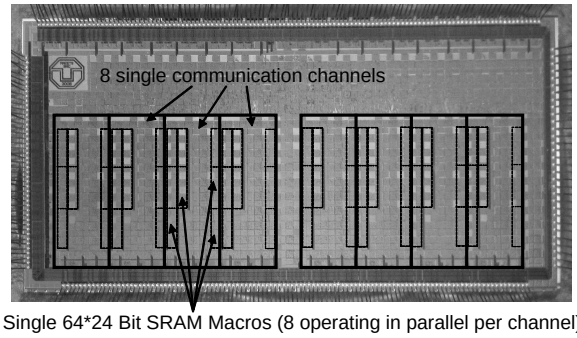


Figure 4: Chip photograph of the binary heap implementation as part of the Digital Network Chip [1]. The DNC contains eight of the channels depicted in Fig. 2a

of keys but higher key size of our design to enable a comparison. After adjustment, [11] would have ca. 50% more throughput rate and gate count 64 times less than stated in table 2. However, the gate count is still a factor of 42 higher than our design. The power consumption of [11] in the adjusted version should be similar to our implementation.

5 Conclusion

We present a semi-custom solution of a priority queue for sorting and buffering of events in a continuously operating communication system applying a binary heap and SRAM full-custom macros. The presented sorting implementation has a similar complexity to serial sort implementations [10], but is much faster. It is somewhat worse in terms of throughput compared to serial/parallel [9] or tree-parallel [11] algorithms, but exhibits a factor 20-40 less gate count. Overall, it achieves a competitive compromise between the different solutions in literature, combining low latency and high throughput required for the pulse routing application with the power and size reduction needed for a high integration density of the complete system.

References

[1] S. Scholze, M. Ehrlich, and R. Schüffny. Modellierung eines wafer-scale systems fuer pulsgekoppelte neuronale netze. In *Dresdner Arbeitstagung Schaltungs- und Systementwurf*, pages 61–66, 2007.

[2] J. Schemmel, J. Fieres, and K. Meier. Wafer-scale integration of analog neural networks. In *IJCNN*, pages 431–438, 2008.

[3] S. Philipp, J. Schemmel, and K. Meier. A QoS network architecture to interconnect large-scale VLSI neural networks. In *IJCNN*, pages 2525–2532, 2009.

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

[5] C. Mehring, U. Hehl, M. Kubo, M. Diesmann, and A. Aertsen. Activity dynamics and propagation of synchronous spiking in locally connected random networks. *Biol. Cybernetics*, 88:395–408, 2003.

[6] J. Partzsch and R. Schüffny. On the routing complexity of neural network models - rent's rule revisited. In *ESANN*, pages 595–600, 2009.

[7] E.M. Izhikevich. which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15(5):1063–1070, 2004.

[8] Standard template library.

[9] K. Ratnayake and A. Amer. An FPGA architecture of stable-sorting on a large data volume: Application to video signals. In *Conf. on Information Sciences and Systems*, pages 431–436, 2007.

[10] E. Mumolo, G. Capello, and M. Nolich. VHDL design of a scalable VLSI sorting device based on pipelined computation. *Journal of Computing and Information Technology*, 12:1–14, 2004.

[11] K. McLaughlin, S. Sezer, H. Blume, X. Yang, F. Kupzog, and T. Noll. A scalable packet sorting circuit for high-speed wfq packet scheduling. *IEEE Trans. on VLSI Systems*, 16:781–791, 2008.

Sophisticated nvSRAM based Transponder Architecture provides novel Features for Field Programming and Safety

Andreas Scade, Stefan Günther
scade@anvo-systems-dresden.com
Anvo-Systems Dresden GmbH, Dresden, Germany
Hagen Grätz, Hans-Jürgen Holland
hagen.graetz@ipms.fraunhofer.de
Fraunhofer Institute for Photonic Microsystems, Dresden, Germany

Abstract

A new Transponder architecture offers the ability to reprogram the transponder in the field. The current implementation of flash based memories for code and/or parameter storage impede this feature due to significant current consumption over a long time for store operation.

The proposed sophisticated approach uses the low current consumption of embedded nvSRAM memories. The nvSRAM embedded memory combines the fast unlimited access of a SRAM bitcell backed up by a SONOS based storage device. The store and erase procedure are based on Fowler-Nordheim erase and programming of the SONOS device. This field emission tunneling consumes very small current and provides the capability of programming all memory cells at the same time. The necessary charge is stored on a capacitor that was charged up very smoothly during power on by controlled current source technique to avoid the feedback on transponder operation. Based on this concept the code and/or parameters could be stored by hardware control of the transponders microcontroller, by software control insight and outsight of the system or even in case of emergency system power down.

Beside the field programming capability a proprietary store procedure control allows to store additional information of the store procedure itself to flag the successful store operation.

This capability provides the new powerful feature of logging the store procedure in a very low current consuming environment and checks the validity after re-powering the system.

This paper gives an introduction in different approaches for transponder architectures by using nvSRAM memory architecture, presents the advantage and introduces new safety features in very low current consuming systems.

Index Terms- Transponder, nvSRAM, SRAM, Store, SONOS, CMOS.

I. INTRODUCTION

Transponders with the ability to store data are a rapidly growing market segment. These transponders require especially low power consumption and high data security. Both issues will be addressed by the proposed new architecture approach.

Transponder with memory function usually contains RAM-, ROM-, FLASH- and/or EEPROM-memory as well as a RF-building block for power supply and data communication with read/write interface. For controlling the data transfer between the RF – unit and the memory an integrated state machine is used. The configuration of the transponder is already determined during the IC design phase and cannot be modified after the circuit is processed in the wafer FAB. The modification of transponders behavior and function needs a circuit redesign with all the disadvantages, risks and cost for verification, mask cost, wafer processing and assembling time.

The simplified block diagram of a hard wired transponder by design is shown in Fig. 1.

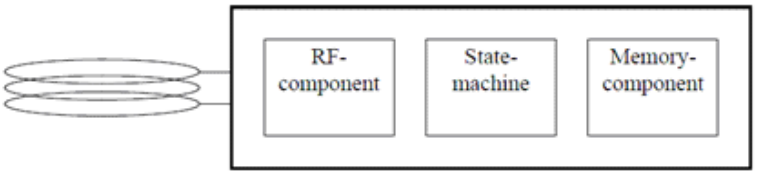


Fig. 1. Simplified block diagram of a hard wired by design transponder

If the state machine in the Transponder is substituted by a programmable microcontroller solution the circuit can be configured after the wafer process and adopted by modification of the firmware very cost efficient to other applications. The block diagram of a configurable transponder is shown in Fig. 2.

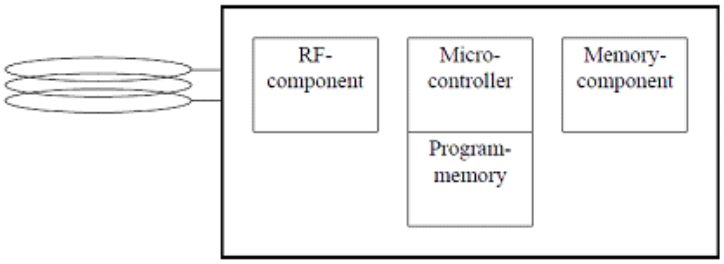


Fig. 2. Block diagram of a configurable transponder

The microcontroller's nonvolatile Program Memory has to have a programming interface available to provide the necessary data and power supply voltage especially the high voltage for programming and erase operation.

Currently FLASH – memories are dominating the embedded market for code storing of microcontrollers.

The advantage of FLASH – memories is beside the availability in nearly all wafer process technology their area efficiency.

State of the Art embedded FLASH – memory building blocks have implemented the sub circuits for necessary programming and erase voltage generation for nonvolatile write operation and supports reprogramming of the microcontroller via transponders data communication interface without applying additional external voltages.

In addition the FLASH – memory can be used for Data storing. The disadvantages are the necessary programming time and the high power consumption for nonvolatile write operations. Furthermore the slow read access time of the FLASH – memory limits significant the overall system performance due to low system clock frequency of the microcontroller.

Using nvSRAM - storage technology helps to overcome these disadvantages.

User data and the microcontroller program data may be used for one building block of the transponder circuit. Special safety features and permission control of the communication protocol prevent an unintended overwriting of the microcontroller program. The necessary Initial programming procedure of the transponders EEPROM is executed during wafer sort via test interface.

Table 1 gives a summary about the most important operation features required for transponder operation.

Supply Voltage	3.3V
System operation current	1-2 mA
Time to POR	10 ms
System clock frequency	16 MHz
EEPROM size	512 Byte
Flash size	16 Kbyte
RAM size	8 Kbyte

Table 1: Basic system requirements and operation parameter

EEPROM and Flash can be easily substituted by nvSRAM with significant improvement of read/write operation.

According to the known operation parameter and range of nvSRAM stand alone memories [2] the voltage operation range of 2.7V to 3.6V and an autostore switching level of 2.6V, wide temperature operation range between -40 and 85 DegC and an internal read write access time of less than 15 ns support system frequencies up to more than 66 MHz. All these parameters fit very well to the requirements of the transponder circuit.

II. nvSRAM TECHNOLOGY

The nvSRAM technology provides significant differences in non-volatile data storing compared with most Flash like technologies. The nvSRAM technology was developed by SIMTEK in the late 1980th. This technology was applied and improved by SIMTEK and ZMD. Both companies provided standalone nvSRAM memories in the range between 16kbit and 1Mbit [2]. For several reasons it's interesting to make this technology available for embedded applications. The main differentiator between EEPROM or Flash and nvSRAM storage architecture is the approach to use for a physical mechanism with very low current consumption for erase and store. This is requested for two reasons:

1. The memory provides the capability of storing the data in case of system power loss. In this case the system is using its own power supply based on the charge stored on a small capacitor during the operation time.

2. To achieve reasonable store time and power consumption the entire memory array have to store at the same time.

A nvSRAM Bitcell

The nvSRAM bitcell combines a common 6-transistor SRAM bitcell and adds 6 transistors for backup array consisting of common n-channel transistors and the storage devices, two SONOS transistors. The circuit schematic of the nvSRAM bitcell is shown in Fig. 3.

During volatile read/write operation the memory access to the nvSRAM is the same like a SRAM access. The SONOS backup devices are not used for volatile SRAM access.

The nonvolatile Backup memory is used basically on power up when a recall operation is performed. During this operation the SRAM bitcell operates as a differential sense amp and read the differential threshold voltage of the erased and programmed SONOS backup cell. This differential read gives an excellent read performance of the nonvolatile data storing, much better compared to Flash or EEPROM read. The second nonvolatile operation mode is the bitcell store operation. The store operation is a very complex procedure. During a first step the SONOS

devices have to be erased. During this step the threshold voltage of all SONOS devices is shifted to a negative threshold voltage. In a second step - the so called programming - the SRAM bitcell is connected to the SONOS transistors.

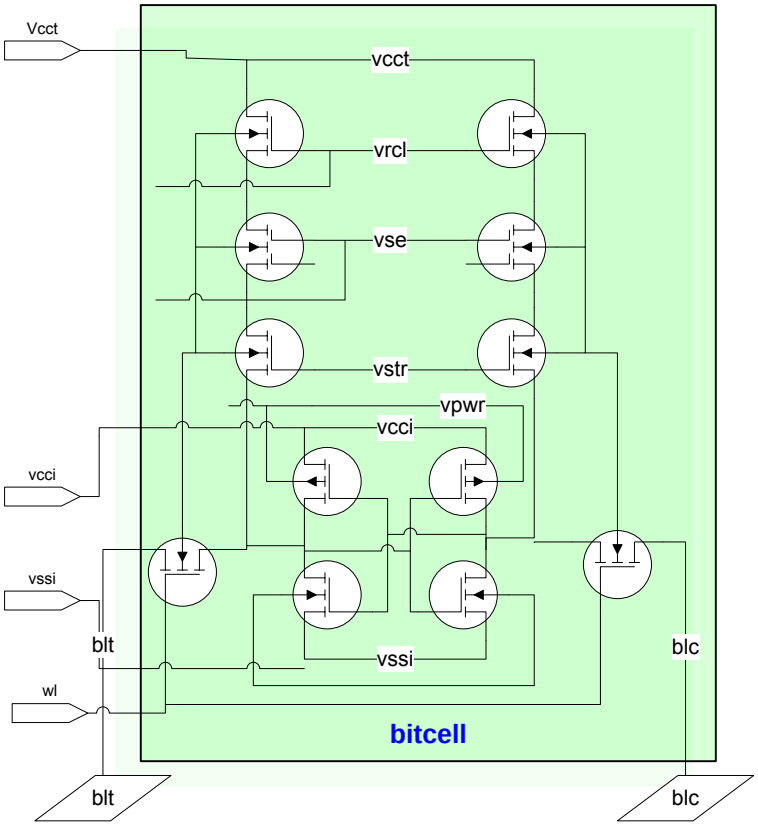


Fig. 3. Circuit schematics of the nvSRAM bitcell

Erase and Store based on modified Fowler-Nordheim tunneling. Fowler-Nordheim tunneling requires an accurate erase control in terms of voltage and erase time to prevent over erase but the advantage of this mechanism is that it nearly consumes no current and thus it's feasible to use it for very low power applications [1].

III. EMBEDDED MEMORY CIRCUIT DESIGN

T The implementation of the nvSRAM cell as an embedded system on the interface between transponder and nvSRAM is very simple because the nvSRAM operates like a SRAM.

To get the embedded memory working it needs some analog and control circuitry beside the bitcell. The entire embedded block consists of the SRAM control circuitry, a control block that provides all the signals necessary for store and recall control, a state machine or micro control block, a programmable oszillator providing the time base for store and recall procedure, a band gap reference and a programmable high voltage generator for generating the negative voltage for erase (Vneg ~ 12V) and the positive voltage for programming (Vpos ~ 12V). Sequencing the Vpos and Vneg voltage and time and controlling the signals of the bitcell provides a broad range of different store procedures to support different applications. The block diagram of the embedded nvSRAM is shown in Fig. 4.

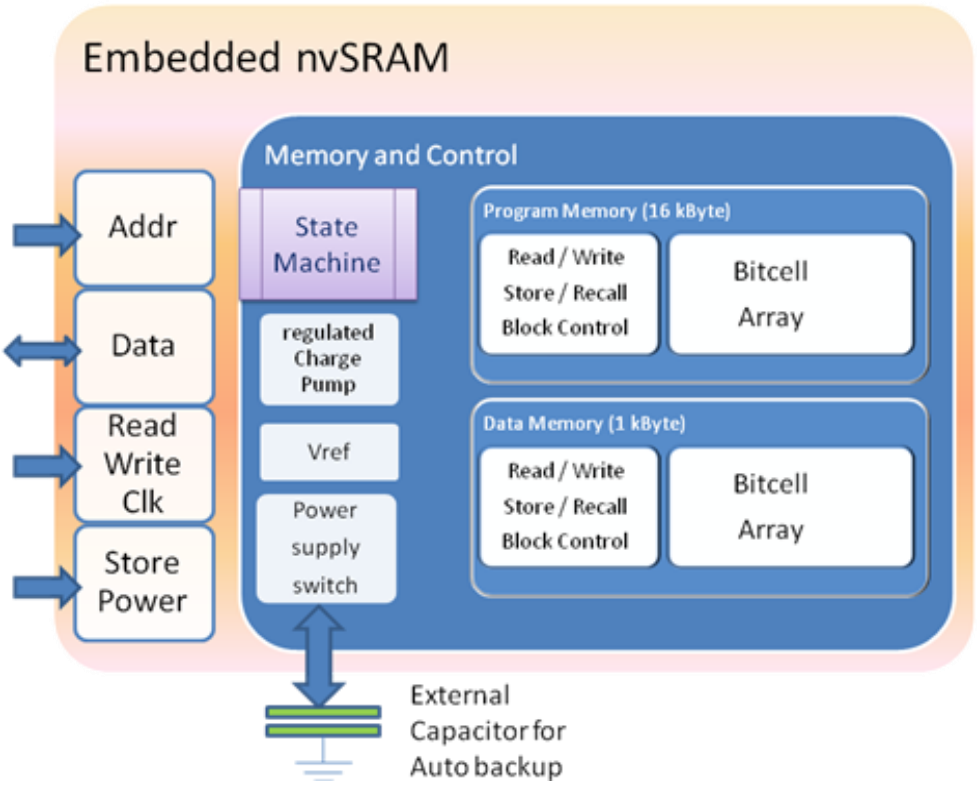


Fig. 4. Block diagram of the embedded nvSRAM macro

IV. NEW FEATURES BY USING nvSRAM

The nvSRAM approach adds new system features to the transponder system. One of the most important advantages of the stand alone nvSRAM's is the ability to auto backup in case of emergency power lost the content of the SRAM to the nonvolatile backup cell. This procedure is automatically executed when the power control of the nvSRAM detects a voltage loss. A switch disconnects the power supply of the embedded memory array and connects it to a capacitor Cstore. This capacitor has been charged during the normal operation time and its charge is now available for auto backup operation. This operation might be helpful to increase data consistency by auto backup of additional measurements, parameter and time and date information.

Another powerful feature could be used for self checking of the auto backup operation. Data memory and program memory are being erased in parallel. By using a dedicated control schema the program memory is stored and the voltage level of the programming pulse and programming duration is checked. During this time the data memory is kept in erase state. If both parameters are in expected limits a flag in the data memory is set and the data memory and program memory is stored again. After powering up the system again and checking the flag it's possible to get information about the success of last store procedure. It's not possible to reconstruct the data but in pattern sensitive environment the customer gets the possibility to decide about additional strategies for data validation.

V. RESULTS

The size of a nvSRAM memory array consisting of 1kByte Data Memory and 16kByte Program Memory in a 0.18 µm CMOS wafer process is approximately 0.3 mm². It includes the state machine for the store and recall control of the nvSRAM content. The charge pump for on chip voltage generation consumes 0.15 mm².

Measurements were performed on-wafer using a Nextest Maverick test system. A new methodology for store current measurement was developed to overcome the synchronization issue during integration of small but strongly varying currents as observed during the store procedure. Fig. 5 plots the normalized current for 1kByte nvSRAM memory size. Although there is no significant channel currents the overall programming current depends on the memory size due to the increase of the size of the capacitance of the SONOS store gates.

The measurements predict a store current for the 16 kByte Program and 1 kByte Data nvSRAM memory array of less than Istore < 100 µA in case of 3.3V supply voltage. This current consumption fits well into the overall current consumption of the transponder circuit. Even more, assuming a current consumption of 100 uA and a supply voltage drop of 0.3 V from Vcc voltage for an auto backup store the needed size of an external capacitor would be less than 5 µF for a store time of 10 ms.

Another advantage of using nvSRAM is to resolve the different functional memory blocks. The transponder architecture doesn't request for a separation in code, data and RAM memory. The nvSRAM approach combines all advantages in one memory and simplifies the transponder architecture.

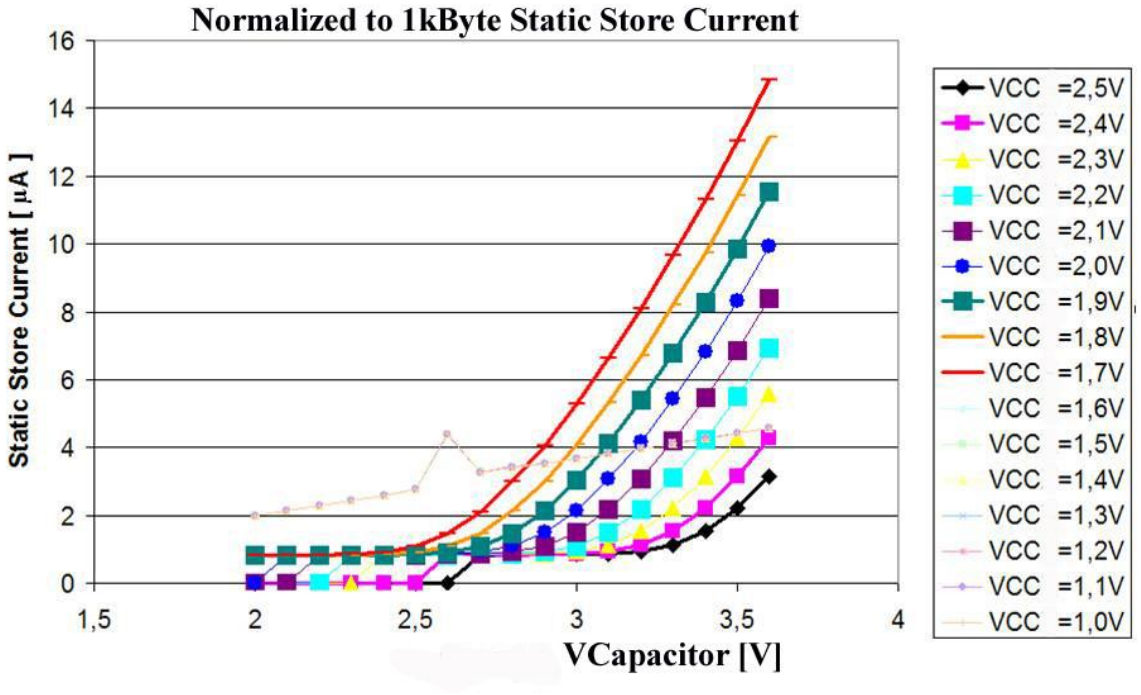


Fig. 5. Normalized to 1kByte nvSRAM array static store current

VI. CONCLUSION

A concept of a configurable embedded nvSRAM based transponder has been presented in this paper. Due to the extreme low power consumption during store operation it is possible to reprogram the transponder in the field. This allows the use of the transponder in extreme difficult environments where removing of the transponder for reprogramming is impossible. Furthermore the wafer process for transponder is a standard CMOS process based on SONOS programming devices. Thus, the transponder is well suited for commercial applications operating as well as highly secure applications or transplants.

The performance of the nvSRAM memory for all significant parameters makes the nvSRAM an excellent choice to expand the transponder applications and targeting completely new applications.

ACKNOWLEDGEMENTS

The author would like to thank Th. Petzold, Anvo-Systems Dresden, for his contribution in development of methodology for store current measurements and providing measurement results.

REFERENCES

[1] William D. Brown, Joe E. Brewer, Nonvolatile Semiconductor Memory Technology, IEEE PRESS, 1998.

[2] A. Scade et al., “1MnvSRAM a sophisticate SoC Solution as a Challenge for Design Architecture, Wafer Process and Design Verification”, DASS 2006, Dresden, 2006.

[3] Datasheet UL62H1616, ZMD AG, Dresden, 2004

Hardware-Implementierung eines Partikelfilters zur Positionsbestimmung

Daniel Froß, Jan Langer, André Froß, Ulrich Heinkel
{daniel.fross, jan.langer, andre.fross, ulrich.heinkel}@etit.tu-chemnitz.de
Technische Universität Chemnitz

Kurzfassung

In drahtlosen Sensornetzen werden Sensordaten in der Regel erst in Verbindung mit deren Aufnahmeort auswertbar. Sind die Positionen der sensortragenden Knoten unbekannt und Positionierungssysteme wie GPS nicht oder nur eingeschränkt nutzbar, müssen die Knotenpositionen aus netzinternen Feldstärke-, Winkel- oder Laufzeitmessungen abgeleitet werden. Aufgrund der Speicher- und Rechenkomplexität robuster Positionsschätzverfahren ist eine Software-Implementierung auf Low-Power Mikrokontrollern meist nicht umsetzbar. Wir präsentieren die Hardware-Implementierung eines Partikelfilters zur dreidimensionalen Positionsschätzung von Netzwerkknoten unter Verwendung laufzeitbasierter Distanzinformationen. Diese kann als Co-Prozessor eines Low-Power Mikrokontrollers eingesetzt werden. Die VHDL-Designbeschreibung wurde auf Basis eines vollständigen Satzes formaler Eigenschaften erstellt und für eine FPGA-Prototypenplattform synthetisiert.

1. Einleitung

Die Positionskenntnis von Teilnehmern drahtloser Sensornetze gewinnt zunehmend an Bedeutung. In vielen praktischen Anwendungen ergeben Sensordaten erst in Verbindung mit deren Aufnahmeort ein aussagekräftiges Bild. Geographische Routing-Protokolle leiten Nachrichten zu Knoten in einer bestimmten geographischen Richtung weiter und müssen hierfür deren Positionen kennen. Ohne den Einsatz zusätzlicher Lokalisierungshardware, bspw. GPS oder Ultraschall, muss die Position der Knoten aus netzwerkinternen Feldstärke-, Laufzeit- oder Winkelmessungen der Funksignale geschlossen werden. Aufgrund der inhärenten Unsicherheit solcher Messungen, bedingt durch das Messsystem, Mehrwegeausbreitung etc., ist es sinnvoll probabilistische Verfahren einzusetzen, die in der Lage sind, solche Effekte zu berücksichtigen. Eine weit verbreitete Methode zur Schätzung eines unbekannten Zustandes (Position) unter Einbeziehung von Systembeobachtungen (Messungen) sind Bayes-Filter, bei denen Zustand und Beobachtung allgemein als Wahrscheinlichkeitsverteilungen modelliert werden. Kalman-Filter sind eine mögliche und besonders effiziente Implementierungsvariante von Bayes-Filtern. Da sowohl Zustand als auch Beobachtung als Gauß-Verteilungen modelliert werden, lassen sich Filterupdates vollständig über einen Satz von Aktualisierungsgleichungen für Mittelwert und Kovarianz beschreiben. Nachteilig wirkt sich aus, dass sich mit Kalman-Filtern nur Systeme mit (nahezu) Gauß'scher Fehlerverteilung und linearen System- und Beobachtungsmodellen beschreiben lassen. Erweiterungen für nichtlineare Systeme, wie der Extended oder Unscented Kalman Filter [1] [2] linearisieren entweder das Zustands- und/oder Beobachtungsmodell des zugrunde liegenden Systems oder verwenden charakteristische Stützstellen zur Transformation. Die Restriktion auf Gauß'sche Fehlerverteilungen bleibt jedoch bestehen. Eine weitere ebenfalls sehr populäre Implementierungsvariante des Bayes-Filter Algorithmus sind Partikelfilter [3] . Im Gegensatz zu Kalman-Filtern können hier beliebige Wahrscheinlichkeitsverteilungen über einen Satz diskreter Zufallsvariablen (Partikel) der zugrunde liegenden Verteilung genähert werden. Aus Gründen der Approximationsgenauigkeit sollte die Partikelanzahl möglichst groß sein. Die dadurch bedingte Speicher- und Rechenkomplexität ist jedoch in der Regel nicht umsetzbar auf Low-Power Mikrocontrollern, die in der Mehrzahl in drahtlosen Sensorknoten eingesetzt werden. In solchen Fällen ist der Einsatz einer dedizierten und damit effizienten Hardwarekomponente wünschens-

wert. Wir präsentieren die prototypische Implementierung eines Partikelfilters zur Positionsschätzung als separate Hardwarekomponente, die über eine serielle Schnittstelle an bestehende Netzwerkknoten angekoppelt werden kann. Der Rest des Papers ist folgendermaßen strukturiert: In Abschnitt 2 wird die Algorithmik des Partikelfilters zur Positionsschätzung, in Abschnitt 3 dessen Implementierung mit Operationseigenschaften vorgestellt. Die Abschnitte 4 und 5 schließen das Paper mit den erreichten Ergebnissen und einer kurzen Zusammenfassung ab.

2. Partikelfilter zur Positionsbestimmung

Bayes-Filter repräsentieren den Zustand eines Systems allgemein als Wahrscheinlichkeitsdichtefunktion (pdf, probability density function) über alle möglichen Systemzustände. Partikelfilter sind eine konkrete Implementierungsvariante des Bayes-Filter-Algorithmus, bei der diese pdf über einen Satz diskreter Stützstellen, sog. Partikel, angenähert wird. Die Partikel konzentrieren sich dabei in Bereichen mit hoher Wahrscheinlichkeitsdichte. In unserem Fall stellt die gesuchte dreidimensionale Position *(x, y, z)* des Netzwerkknotens den Zustand des Systems dar. Das System lässt sich nur indirekt über Distanzmessungen r_a zu Ankerknoten bekannter Position (x_a, y_a, z_a) beobachten. Zur eindeutigen Positionsbestimmung im dreidimensionalen Fall sind Distanzmessungen zu mindestens vier nicht in einer Ebene liegenden Ankerknoten notwendig. Um jeden Anker ergibt sich eine Kugel mit einem der Distanzmessung entsprechenden Radius. Alle auf der Kugeloberfläche befindlichen Positionen genügen dabei der Distanzmessung. Der Schnittpunkt aller Kugeln ergibt schließlich die gesuchte Position. Mathematisch lässt sich dieser Sachverhalt durch folgendes nichtlineares Gleichungssystem darstellen.

$$(x - x_a)^2 + (y - y_a)^2 + (z - z_a)^2 = r_a^2, \quad a = 1..N > 3$$

Im betrachteten Anwendungsfall werden die Distanzinformationen auf Basis von Laufzeitmessungen der Funksignale abgeleitet. Unter dem Einfluss von Mehrwegeeffekten im Signalpfad sowie der begrenzten Genauigkeit des zugrunde liegenden Messsystems unterliegen die Distanzmessungen zum Teil erheblichen Störeinflüssen. Dies führt dazu, dass sich trotz statischer Positionen zweier Knoten variierende Distanzmessungen dazwischen ergeben. Partikelfilter bieten die Möglichkeit solche Einflüsse im Beobachtungsmodell zu berücksichtigen. Die angenommene zeitliche Entwicklung der Knotenposition wird über das Zustandsvorhersagemodell abgebildet. Ausgehend von der alten Position werden Hypothesen einer neuen Position aufgestellt. Diese werden anschließend durch die Beobachtung (Distanzmessung) entweder validiert oder gelöscht. Eine Aktualisierung des Filters beinhaltet folgende Schritte:

1. Vorhersage einer hypothetischen Position für jedes Partikel ausgehend von seiner letzten Position. In unserem Fall wird die Position des Netzwerkknotens als statisch angenommen. Die modellseitige Unsicherheit dieser Annahme wird durch ein Verrauschen der Partikel berücksichtigt. Dabei wird eine erwartungswertfreie Normalverteilung einstellbarer Varianz zugrunde gelegt. Bei entsprechender Wahl der Varianz können somit auch bewegliche Netzwerkknoten modelliert werden. In der Implementierung werden die normalverteilten Zufallswerte auf Basis des zentralen Grenzwertsatzes aus jeweils 12 gleichverteilten Zufallswerten abgeleitet.
2. Bestimmung eines Gewichts (importance factor) für jedes Partikel. Das Gewicht wird unter Einbezug der Messung, der zugehörigen Ankerposition sowie eines spezifischen Messmodells bestimmt. Das Modell beschreibt die Wahrscheinlichkeit, die gegebene Distanz zum Anker an einer beliebigen Position im Raum zu messen. In unserer Implementierung werden Unsicherheiten bezüglich des physikalischen Messsystems und Mehrwegeeffekten berücksichtigt. Folgende Funktion wird zur Gewichtsberechnung verwendet (Vgl. Abb. 1):

$$w = \frac{k^2}{k^2 + \Delta d^2}, \quad \Delta d = r_a - \sqrt{(x - x_a)^2 + (y - y_a)^2 + (z - z_a)^2}, \quad k^2 = \begin{cases} k_1^2, \, wenn \; \Delta d < 0 \\ (k_1 + k_2)^2, \, sonst \end{cases}$$

Darin entspricht Δd der Differenz zwischen der realen Messung r_a und der für den betrachteten Partikel erwarteten Messung zum Anker. Der verwendete Stauchungsparameter k kann in Abhängigkeit des Vorzeichens von Δd die Werte k_1 oder k_1+k_2 annehmen. Unsicherheiten bezüglich des verwendeten Messverfahrens werden als symmetrisch zur erwarteten Messung angenommen und durch k_1 berücksichtigt. Längere Messungen können zusätzlich durch Mehrwegeeffekte entstehen und werden durch k_2 berücksichtigt.

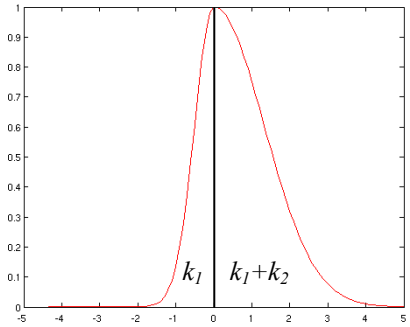


Abb. 1: Gewichtsfunktion w in Abhängigkeit von Δd

3. Generierung des finalen Partikelsatzes durch eine Resamplingprozedur des hypothetischen Partikelsatzes aus (1). Die Wahrscheinlichkeit der Übernahme eines Partikels in den finalen Satz ist proportional zu dessen Gewicht. Auf diese Weise werden stark gewichtete Partikel dupliziert wohingegen schwach gewichtete wegfallen. Die resultierenden Partikel fokussieren sich somit auf Regionen mit hoher Wahrscheinlichkeit. In unserer Implementierung wird ein so genannter Low-Variance-Resampler eingesetzt.
4. Extraktion der durch den resultierenden Partikelsatz angenäherten Wahrscheinlichkeitsdichteverteilung. In unserer Implementierung werden in diesem Schritt der Mittelwert und die Kovarianzen in x,y und z-Richtung über alle Partikel ermittelt. Diese Kenngrößen entsprechen den ersten beiden statistischen Momenten.

Soll das Filter ohne Messung aktualisiert werden, entfallen die Schritte 2-4. Die neue Position wird dann ausschließlich auf Basis des eingesetzten Bewegungsmodells vorhergesagt. Das Filter benötigt zur eindeutigen Positionsbestimmung im dreidimensionalen Raum Distanzmessungen zu mindestens vier nicht auf einer Ebene liegenden Ankerknoten. Bei nur zwei unbekannten Dimensionen reichen drei Anker aus. Anderenfalls können sich als Resultat der Mehrdeutigkeit Partikelverteilungen ohne globales Maximum ergeben. Abb. 2 zeigt Partikelverteilungen, die sich durch Einbeziehung unterschiedlicher Distanzmessungen ergeben.

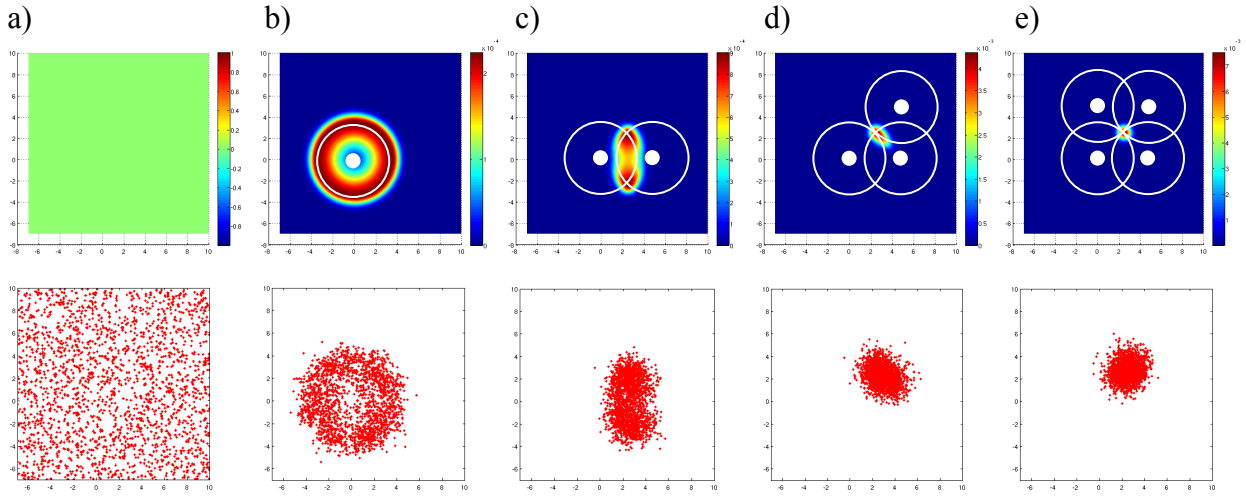


Abb. 2: Darstellung der Wahrscheinlichkeitsverteilung einer 2D-Positionsschätzung (obere Reihe) a) inital ohne Messung, b-e) nach Messung zu 1-4 Ankerknoten. Die Punkte stellen die Ankerpositionen dar, die Kreisradien entsprechen den gemessenen Distanzen. Es wird die in Abb. 1 vorgestellte Messfunktion mit k_1^2=0.5 und $(k_1+k_2)^2$= 3 Längeneinheiten^2 verwendet. In der unteren Reihe ist das Ergebnis des Partikelfilters dargestellt.

3. Filterimplementierung mit Operationseigenschaften

Die kommerzielle Verifikationssoftware 360MV™ von OneSpin Solutions [4] verwendet eine so genannte *Gap Free Verification* Methodik, welche Operationseigenschaften verwendet, um die Vollständigkeit einer Eigenschaftsmenge zu beweisen. Die zur Beschreibung der Eigenschaften verwendete Sprache nennt sich InTerval Language (ITL). Sie bietet spezielle Sprachkonstrukte, die es dem 360MV Tool erlauben, zu beweisen, dass durch die Abfolge der Eigenschaften für jede gültige Eingangsfolge des Systems exakt eine Ausgabesequenz festgelegt ist [5] . Die genutzte Beweisengine ist mächtig genug, um auch für größere Designblöcke diese Vollständigkeit der Eigenschaften nachzuweisen, und zusätzlich zu prüfen, ob die einzelnen Eigenschaften für das Design erfüllt sind. Ersterer Nachweis kann dabei gänzlich ohne ein bestehendes Design durchgeführt werden.

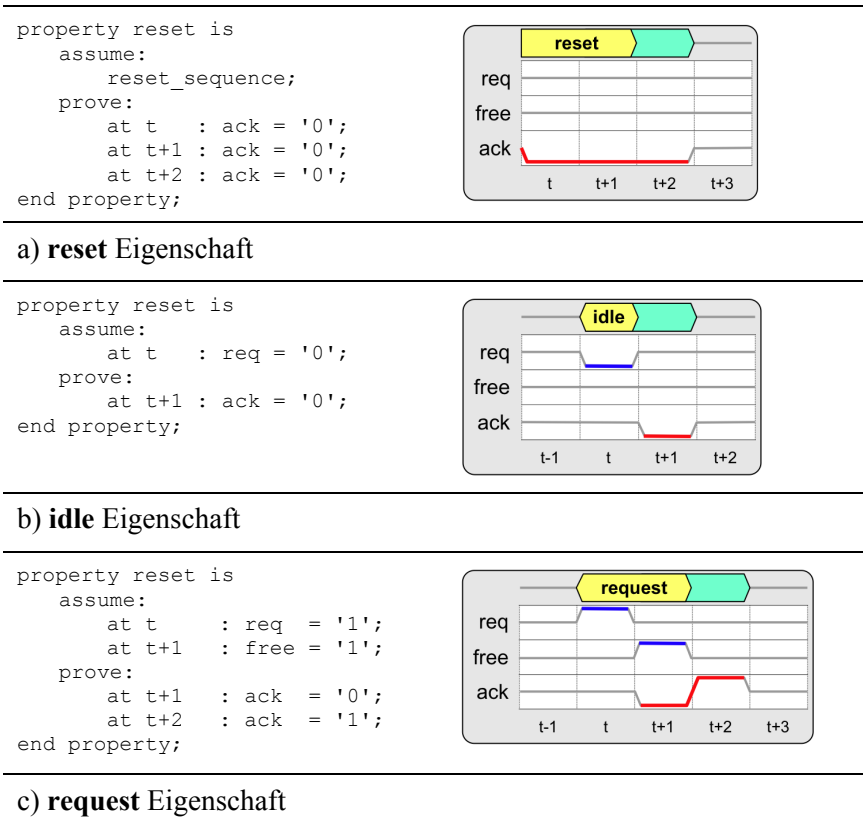
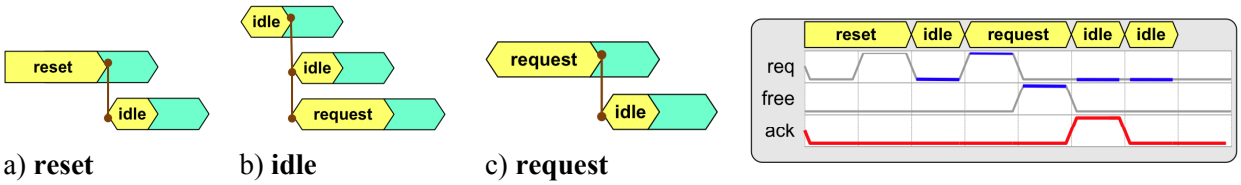


Abb. 3: Eigenschaftssatz für das stark gekürzte Interrupt Controller Beispiel.

Operationseigenschaften können am besten mit Hilfe eines kleinen Beispiels erklärt werden. Es handelt sich dabei um einen minimalen Interrupt Controller, der in Abb. 3 gezeigt wird. Er nutzt drei Operationseigenschaften um das Verhalten abzubilden. Die Eigenschaft reset legt die Reaktion der Komponente unmittelbar nach dem Rücksetzen des Reset-Signals fest. Dabei wird lediglich das Ausgangssignal ack für drei Taktzyklen auf low gesetzt. Die idle Eigenschaft hingegen tritt im Falle eines nicht angelegten req Signals ein und setzt das Signal ack ebenfalls auf low. Die Eigenschaft request ist wiederum für den Fall eines eingetretenen Requests zuständig, d.h. sie wird bei gesetztem req und darauf folgendem free aktiviert. In den gezeigten Darstellungen sind blaue Kennzeichnungen die Aktivierungsbedingungen (Assumptions) einer Eigenschaft, während die Ausgangssignale (Commitments) rot markiert sind. Weiterhin muss der Verifikationsingenieur einen Eigenschaftsgraphen bereitstellen, der die Übergänge zwischen den Eigenschaften festlegt, d.h. er gibt alle möglichen Nachfolger jeder Eigenschaft vor.



a) **reset** Eigenschaft b) **idle** Eigenschaft c) **request** Eigenschaft d) Abfolge von Eigenschaften

Abb. 4: Eigenschaftsgraph des Interrupt Controller Beispiels.

In Abb. 4a-c) werden alle Übergänge des Interrupt Controller Beispiels gezeigt. Wenn nun eine gegebene Eingangsfolge auf Eigenschaften abgebildet wird, ergibt sich eine Eigenschaftssequenz wie in Abb. 4d). Die Eigenschaften dürfen dabei nicht überlappen und 360MV hilft beim Finden möglicher Lücken in der Eigenschaftsstruktur.

In [6] werden Operationseigenschaften als alternative Entwurfsmethodik zwischen Register-Transfer-Ebene und höheren Abstraktionsebenen vorgeschlagen, die insbesondere in denjenigen Anwendungsbereichen die Entwurfsproduktivität erhöht, in denen das Verhalten einer Komponente als zeitlich aufeinander folgende Operationen definiert werden kann. Zu diesem Zweck wurde das Entwurfswerkzeug *VHISyn* vorgestellt. Es kann aus einem vollständigen Eigenschaftssatz zusammen mit dem zugehörigen Eigenschaftsgraphen ein taktgenaues VHDL-Modell auf Register-Transfer-Ebene generieren.

Der in Abschnitt 2 vorgestellte Partikelfilter wurde mit Operationseigenschaften beschrieben, die denen aus obigem Beispiel ähneln. Abb. 5 zeigt die Komponenten des Designs. Die vier Blöcke *prediction, weight calculation, resampling* und *statistics* entsprechen funktional den vier Update-Schritten aus Abschnitt 2. Die Berechnung erfolgt in Festkommaarithmetik, Koordinatenangaben sind mit 14 Bit per Dimension, Distanzen mit 15 Bit und die Gewichte mit 32 Bit kodiert. Da eine initiale Unwissenheit des Filters angenommen wird, startet es mit einem Satz von zufällig verteilten Partikeln sowie Einheitsgewichten. Der Resampling-Block benötigt die kumulative Summe aller Gewichte und kann erst starten, wenn diese feststeht. Aus diesem Grund ist es notwendig den kompletten Satz von M=8192 Partikeln inklusive der zugehörigen Gewichte in zwei FIFOs am Eingang des Resampling-Blocks zwischenzuspeichern. Nach Durchlaufen der Resampling-Stufe werden die neu generierten Partikel den nachfolgenden Stufen zur weiteren Prozessierung übergeben. Die durch VHISyn generierte VHDL-Beschreibung des Designs wurde unter Verwendung von Xilinx ISE 10.1 für eine vorliegende Virtex-II Pro (xc2vp30) FPGA Prototypenplattform synthetisiert. Das Design ist über den PLB Bus mit einem der beiden eingebetteten PowerPCs verbunden. Der Prozessor dient als Schnittstellenkomponente, die dem Design eingehende Messungen sowie Modellparameter (k_1,k_2, …) zuführt und die Statistikdaten ausliest.

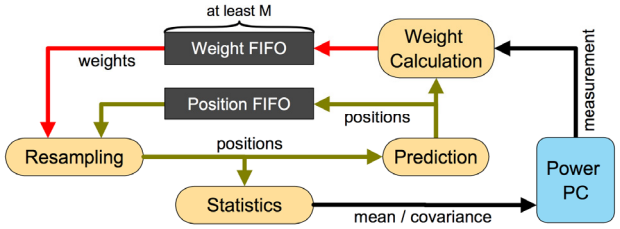


Abb. 5: Top-Level Komponenten des Partikelfilter Designs

4. Ergebnisse

Abb. 6 zeigt das Ergebnis der Positionsschätzung auf Basis einer simulierten Zufallsbewegung mit Geschwindigkeitskomponenten in *x*- ,*y*- und *z*-Richtung, denen während der Simulation normalverteilter Zufall aufaddiert wurde. An den Begrenzungsflächen wurden die entsprechenden Geschwindigkeitskomponenten invertiert. Die der simulierten Position zugehörigen Distanzen zu den einbezogenen Ankern wurden entsprechend dem in Abschnitt 2 eingeführten Messmodell verrauscht (k_1=0.5m, k_2=1m) und dem Filter über eine serielle Verbindung als Mes-

sungen zugeführt. Sowohl simulierte Position als auch die vom Filter zurückgelieferten Mittel- und Kovarianzwerte der geschätzten Position sind in Abb. 6 als *x*-*y*-Projektion dargestellt. Die abgebildeten Kovarianzellipsen stellen den Unsicherheitsbereich innerhalb der einfachen Standardabweichung dar. Alle Längeneinheiten entsprechen Zentimetern. Die korrekte Funktionsweise des Partikelfilters konnte auch unter realen Messbedingungen gezeigt werden. Dazu wurde die verwendete Prototypenplattform in ein bestehendes Drahtlosnetzwerk [7] integriert. Es basiert auf dem IEEE 802.15.4a Standard für drahtlose Nahbereichsnetze und bietet die Möglichkeit Funksignallaufzeiten und damit Distanzen zwischen Knoten zu messen. Der seriell an die Plattform angekoppelte Netzwerkknoten misst zyklisch Distanzen zu seinen umgebenden Ankerknoten und führt diese dem Filter zu. Die geschätzte Position wird anschließend ausgelesen und per Funk an das Netzwerkmanagement übermittelt und visualisiert. Aus der Synthese des Designs ergibt sich eine maximale Taktfrequenz von ca. 25 MHz. Bei einer Anzahl von 8192 Partikeln und einer Verarbeitungszeit von 3 Takten pro Partikel ergibt sich somit die Dauer eines Filterupdates zu ca. 1 Millisekunde. Das verwendete Xilinx Virtex-II Pro FPGA wird vom Design mit 6011 von 13696 Slices zu 43% ausgelastet.

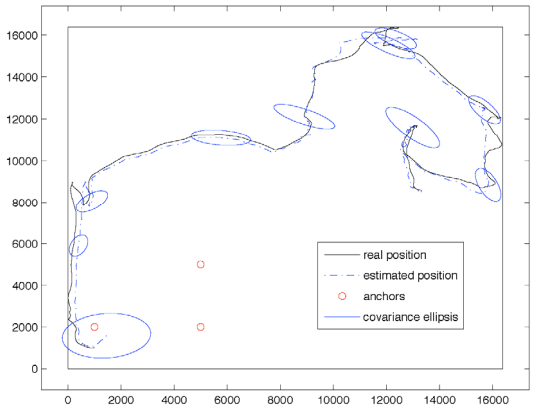


Abb. 6: Ergebnis der Positionsschätzung mit generierten Messdaten (x-y-Projektion)

5. Zusammenfassung

Es wurde die Hardware-Implementierung eines Partikelfilters zur dreidimensionalen Positionsschätzung unter Verwendung laufzeitbasierter Distanzinformationen vorgestellt. Die VHDL-Designbeschreibung wurde auf Basis formaler Operationseigenschaften erstellt und für eine FPGA-Prototypenplattform synthetisiert. Die korrekte Funktionsweise des Filters konnte mit simulierten und realen Messdaten gezeigt werden.

Danksagung

Diese Arbeit wurde gefördert durch das Bundesministerium für Bildung und Forschung (BMBF) im Rahmen der InnoProfile Initiative “GPSV - Generalisierte Plattform zur Sensordatenverarbeitung” (03 IP 505) und dem Projekt “HERKULES” (01 M 3082).

Literatur

[1] G. Welsh, G. Bishop, *“An Introduction to the Kalman Filter“*, Technical Report, University of North Carolina, Chapel Hill, 1995.

[2] S. J. Julier, J. K. Uhlmann, *„A New Extension of the Kalman Filter to Nonlinear Systems“*, in 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, 1997.

[3] S. Thrun, W. Burgard, D. Fox, *„Probabilistic Robotics“*, MIT Press, 2005, Kap. 4.3, S. 96–113.

[4] Onespin Solutions GmbH, *OneSpin 360MV*, www.onespin-solutions.com.

[5] J. Bormann, *„Vollständige funktionale Verifikation“*, Dissertation, Universität Kaiserslautern, 2009.

[6] J. Langer, U. Heinkel, *„High level synthesis using operational properties“*, in Forum on Specification & Design Languages (FDL), 2009.

[7] D. Froß, A. Froß, C. Roßberg, M. Rößler, U. Heinkel, *„IEEE 802.15.4a basiertes Sensor- und Lokalisierungsnetz“*, in 9. Chemnitzer Fachtagung Mikromechanik & Mikroelektronik, 2009.

A Multibit Continuous-Time Delta-Sigma ADC for Direct Conversion of CdZnTe Detector Arrays

Matthias Völker, Haiyan Zhou, Johann Hauer
matthias.völker@, haiyan.zhou@, johann.hauer@iis.fraunhofer.de
Fraunhofer Institut für Integrierte Schaltungen , Am Wolfsmantel 33 91058 Erlangen

Abstract

Beside the traditional signal processing channel for radiation detector arrays based on multiple analog filter stages, a new direct analogue to digital conversion strategy is presented in this paper. The main building block of the new channel topology is a continuous-time analog to digital converter. This paper presents the design and implementation of a 78 dB SNR multibit continuous-time delta-sigma ADC. Such ADC is dedicated to the readout of CdZnTe detector arrays at the bandwidth of 625 kHz. This ADC offers the possibility to move most signal processing steps like pulse shaping, energy detection and timing analysis into the digital domain. The die area and power consumption can be reduced by using digital filtering and advanced signal processing algorithmen. The delta-sigma ADC has been designed in an 180nm 1P6M CMOS technology. The circuit consumes 3.8 mW from 1.8 Volt supply.

I. System description

Large-area gamma-ray detection is a base technology for different imaging methods in medical applications. Single photon emission computed tomography (SPECT) and positron emission tomography (PET) are only two examples. The development of sensors for these applications is an ongoing process. Cadmium zinc telluride (CdZnTe) detectors are promising candidates for room temperature operated radiation detection sensor arrays. Beside the sensor itself the signal processing strategies have to be improved and optimized for multi-channel applications. Signal processing for nuclear imaging applications has been developed over several decades [1]. The use of large detector arrays in combination with integrated signal processing offers the possibility to transfer new signal acquisition and processing concepts from other fields. During recent years many traditional analog processing concepts have been replaced by direct analog to digital conversion in combination with digital signal processing. The main reason for this strategy is to reduce power consumption and die area by exploiting the great advancements in digital processing. The model of an 8x8 pixel CdZnTe detector and the detector crystal unit are shown in Fig. 1. [7]

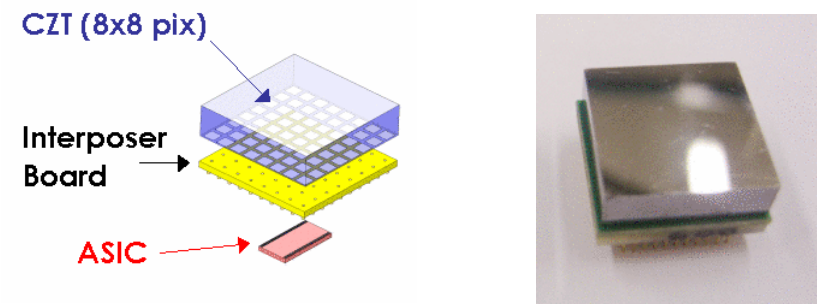


Fig. 1. Model of CdZnTe detector and detector crystal unit (from NASA)

This paper presents a continuous-time delta sigma ADC especially designed for direct analog to digital conversion of CdZnTe detector arrays. In Section II, the proposed concept for direct analog to digital conversion is described. Section III presents the behavioural modeling of the delta-sigma modulator and the circuit level implementation of each building block. Section IV shows simulation results and layout of ADC. The paper closes with a summary in Section V.

II. Direct conversion concept for CdZnTe detector arrays

Readout circuits for CdZnTe detector arrays measure charges caused by electron-hole pairs generation due to ionizing events. These circuits are mainly based on semi-gaussian shaping of the input signal followed by peak height detection and timing analysis in the analog domain [2]. The semi-gaussian shaping has been developed over many years [1] and is up to now used in many detector circuits. A typical detector channel as used in current developments [3] is depicted in Fig. 2. The charge pulse emitted by the sensor is integrated by a charge sensitive amplifier and filtered by shaping filters. Peak sampling or timing analysis can be done on the shaped pulses depending on the selected shaping time.

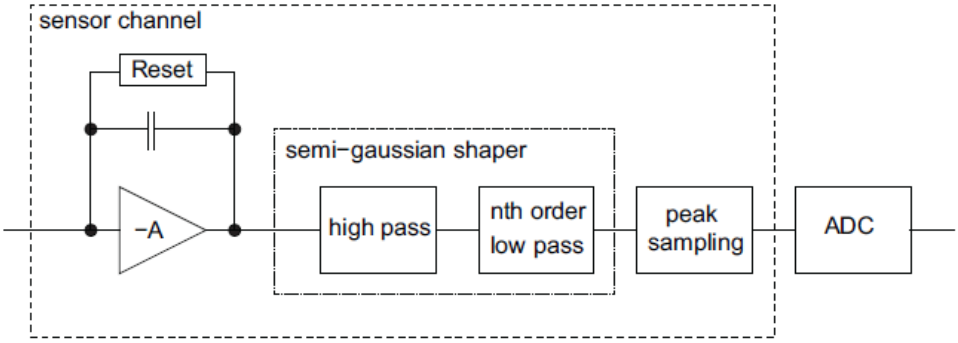


Fig. 2. Conventional channel architecture (the ADC is shared by multiple channels)

Fig. 3 depicts the described signal processing chain. The analog shaping filters in front of the sampling circuit are required to avoid aliasing during the sampling process and to suppress high frequency noise. Relaxed requirements in terms of anti-alias filtering would offer the possibility to move pulse shaping, peak detection and peak sampling into the digital domain. Continuous-time (CT) delta-sigma ADCs fit very well into these requirements:

- The shaping filters can be moved to the digital domain due to the implicit anti-alias filtering of these converters.
- The resolution scales with the data rate which makes it a perfect solution for high data rate timing analysis as well as high resolution peak sampling.
- The digital processing can be easily shared between several pixels.

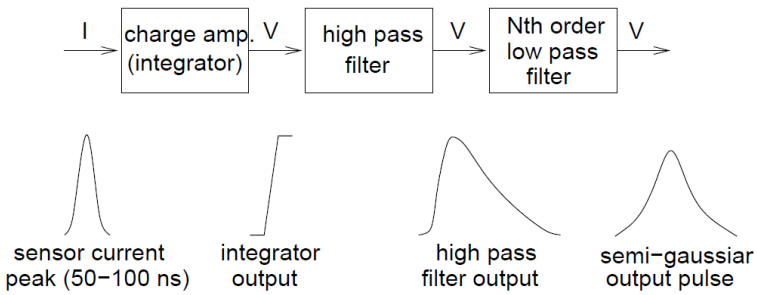


Fig. 3. Conventional signal processing chain

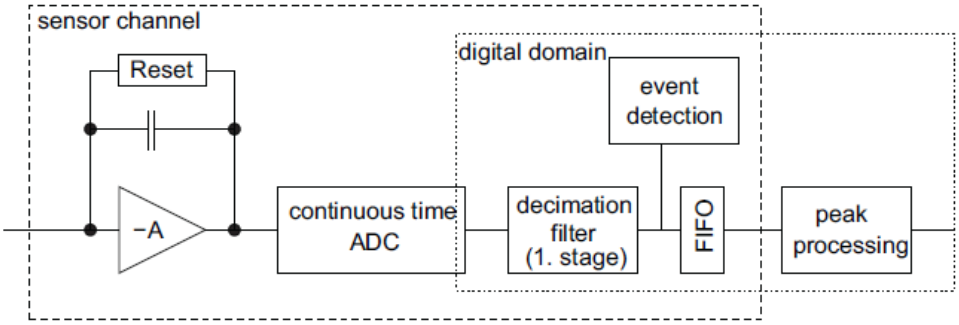


Fig. 4. New proposed channel architecture

This paper presents a new architectural approach for an early analog to digital signal conversion in a multi channel gamma ray detection system. A continuous-time delta-sigma ADC was especially designed for direct analog to digital conversion of CdZnTe detector arrays. Fig. 4 shows a block diagram of the channel architecture of the new approach.

III. Behavioral modeling and circuit level implementation

There are various architectural choices in the design of a CT delta-sigma modulator. The performance of the ADC can be optimized by appropriate selection of the loop filter transfer functions, the modulator order and the number of bits of the quantizer.

A. Topology Selection

The signal processing requirements for CdZnTe detector arrays define the specification for the ADC. The design aim of the peak resolution is 12 bit which results in a minimum SNR of 74 dB. Due to the non-linearity correction in the post process steps the THD can be relaxed to -50 dB. A signal band of 625 kHz is selected to allow every shaping time above 200 ns. Lower shaping times can be achieved with reduced resolution. The generation and distribution of very low jittered clocks on large readout ICs is a challenging task. A multi-bit implementation with non return-to-zero feedback pulses offers the possibility to be tolerant against clock jitter. It also allows a second order loop in combination with an oversampling ratio of 50 which reduces the number of operational amplifiers and the requirement of gain-bandwidth as well. Fig. 5 without the dashed part illustrates the idealized model of such a modulator. A 2nd order integrator and a 4 bit quantizer compose the feedforward path while the feedback path consists of a 4 bit DAC. The gain coefficients k_1 and k_2 amplify the output signal and feed it back to the input of both integrators respectively. Behavioral simulations in Matlab proved that high resolution can be obtained, when k_1 and k_2 equal 1 and 1.5 respectively. The model in Fig. 5 assumes zero delay inside the loop. In reality there is a non-zero delay known as excess loop delay which has to be addressed. In case the excess loop delay t_d reaches the same order of magnitude as half of the clock period T_s, the CT modulator becomes unstable. The delay of the quantizer also depends on the input signals and is therefore strongly signal dependent which could introduce harmonics into the loop. To solve this problem, an additional feedback path is introduced and the delay is forced to a fixed value, as illustrated in the dashed part in Fig. 5 [4]. This path is applied directly to the quantizer input and an explicit time delay t_d is added. Assuming that $t_d = 0.5T_s$, the feedback coefficients are:

$$a_1 = 1$$
$$a_2 = 2$$
$$a_3 = 0.875$$

The implementation of gain coefficients based on RC time constants strongly depends on process variations. Extensive behavioral simulations show that the modulator remains stable up to 20 % variation of the coefficients. Digital controlled trimming is used to attenuate the process variations down to this limit.

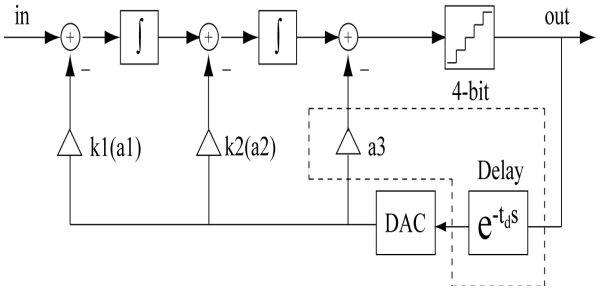


Fig. 5. Modulator model

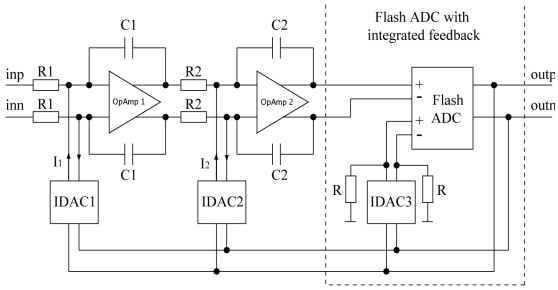


Fig. 6. Modulator block diagram

B. Circuit Implementation

After determing the feedback coefficients a_1 to a_3, the modulator model should be transformed into circuit level. The modulator block diagram is illustrated in Fig. 6. Two RC integrators are implemented for the 2nd order loop filter. The 4 bit quantizer is realized as a flash ADC. Three current-mode DACs (IDACs) feed back the quantizer output to the inputs of the respective building blocks.

Flash ADC with Integrated Feedback: Since the quantizer includes a direct feedback, it is necessary to implement the flash ADC together with the 3rd DAC. In this case a critical point is to implement the summing operation. In order to avoid the complexity of summing two voltage signals at the input stage of the quantizer, the summing operation is shifted to the reference stage. The resulting quantizer exhibits the same signal transfer as achieved by summing at the input nodes. A fixed value is added to the reference voltage by shifting the whole reference resistor string depending on the feedback DAC.

Current mode DACs: The current mode DACs in the feedback loop are located between the flash ADC and the respective integrator. Using a complementary structure and cascoded current source devices, the IDAC cell can precede positive and negative output currents which exhibit complementary values at outp and outn [5]. By combining these currents to the same nodes, the sum of each output current can be obtained for outp and outn respectively.

Loop Filter: Two RC integrators are implemented for the 2nd order loop filter, where the integrator gains are mapped into resistor-capacitor products. To ease the design of the modulator the same amplifier is used in both integrators. Due to the fact that the performance is mainly limited by the first one, the second amplifier can be sized down to reduce power consumption. The OpAmps employ two stage design using Miller compensation. For the first amplifier stage a local CMFB is used. It provides additional gain boosting and increases the slew rate as well as GBW of the amplifier. In the second stage, a SC-CMFB circuit is employed.

IV. Simulation results and layout of ADC

Architectural block level simulation and optimization were done with HDL-Models. Finally a transient simulation for the overall delta-sigma modulator was performed on circuit level. The clock was set to 50 MHz with a duty cycle of 50 %. A sine signal with 0.6 V amplitude (-4.4 dBFS) and 500 kHz frequency was applied. The output data of the modulator were exported to Matlab to compute the spectrum using a 4096-point Hanning windowed FFT. The spectrum of the output data is depicted in Fig. 7. Further transient simulations were performed with sweeping the input signal amplitudes. The resulting SNR versus the normalized input amplitude is shown in Fig. 8. Maximum SNR is simulated at 0.7 V input amplitude (-3 dBFS) and equals 78.3 dB. The calculated Dynamic Range (DR) is 81 dB.

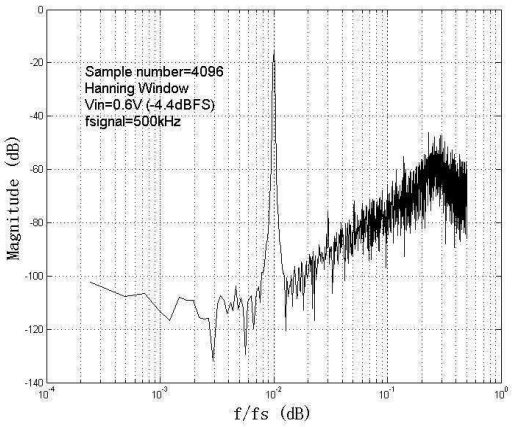


Fig. 7. Power spectrum density

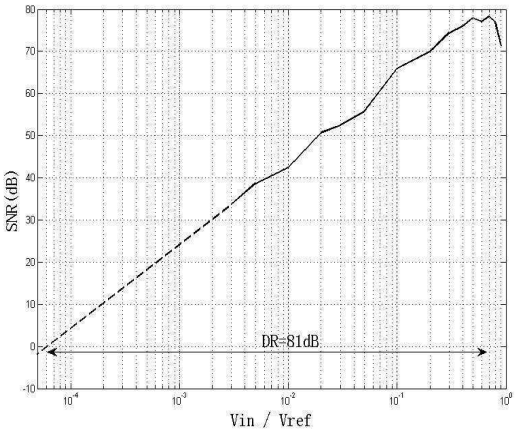


Fig. 8. SNR versus input amplitude

The main performance parameters of the delta-sigma modulator are summarized in Table I.

signal bandwidth	625 kHz
sampling frequency	50 MHz
oversampling ratio	50
dynamic range	81 dB
SNR	78.3 dB
SNDR	57.5 dB
supply voltage	1.8 V
power consumption	3.78 mW
Technology	0.18 μm CMOS

Table I. Performance summary

In the next step a layout of the whole modulator was accomplished and it provides a good matching performance within a 1.5x1.5 mm^2 die area. Fig. 9 illustrates the layout of the continuous time delta-sigma ADC. The prototype chip is currently in fabrication.

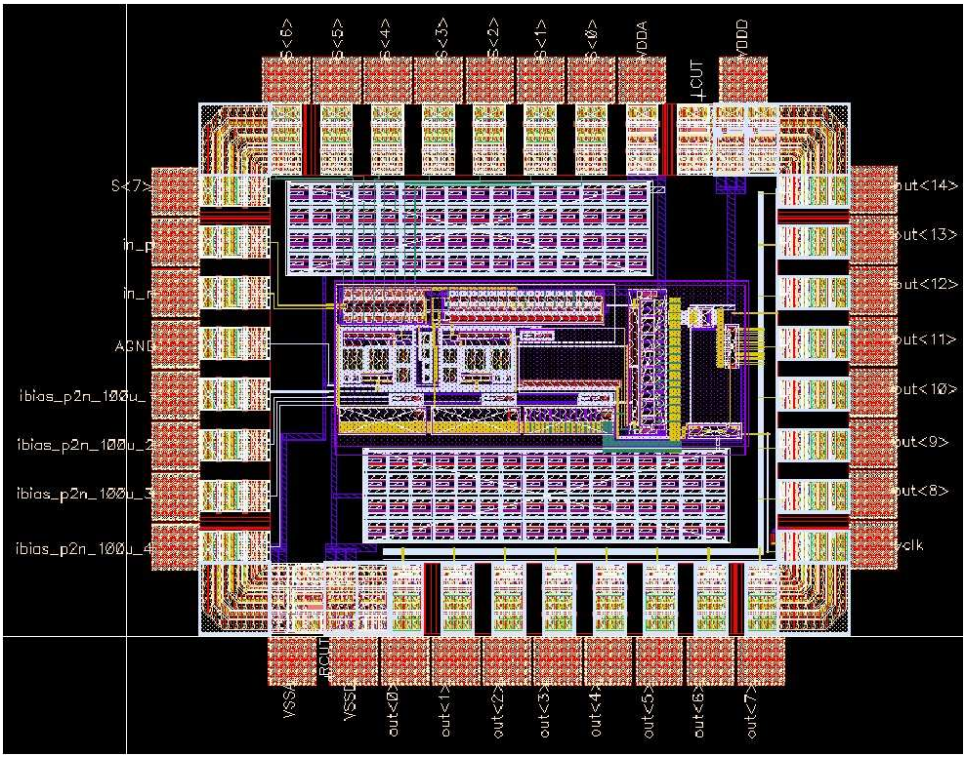


Fig. 9. Layout of the continuous time delta-sigma ADC

V. Summary

This paper covers the design and implementation of a continuous time delta-sigma ADC for a new readout strategy of CdZnTe detector arrays. A 2nd order loop filter associated with a 4 bit quantizer and current-mode feedback DAC were chosen. In behavioral modeling, the critical excess loop delay problem was addressed by means of a local feedback loop. The quantizer was based on a flash-ADC and integrates the local feedback loop. HDL-modeling simulation was used to verify the architectural function. Through circuit level simulations, the output signals were obtained and processed in Matlab. The SNR of the overall delta-sigma modulator is 78 dB and the DR is 81 dB within 625 kHz bandwidth. Clocked at 50 MHz, the modulator consumes 3.8 mW power from 1.8 Volt supply.

Literatur

[1] C. H. Mosher, “Pseudo-gaussian transfer functions with superlative baseline recovery,” *Nuclear Science, IEEE Transactions on*, vol. 23, no. 1, pp. 226–228, 1976.

[2] G. De Geronimo, P. O’Connor, and J. Grosholz, “A generation of cmos readout asics for czt detectors,” *Nuclear Science, IEEE Transactions on*, vol. 47, no. 6, pp. 1857–1867, 2000.

[3] G. De Geronimo, E. Vernon, K. Ackley, A. Dragone, J. Fried, P. O’Connor, Z. He, C. Herman, and F. Zhang, “Readout asic for 3d position-sensitive detectors,” in *IEEE Nuclear Science Symposium Conference Record NSS ’07*, E. Vernon, Ed., vol. 1, 2007, pp. 32–41.

[4] M. Ortmanns and F. Gerfers, *Continuous-Time Sigma-Delta A/D Conversion*. Springer, 2005.

[5] R. Schreier and G. Temes, *Understanding Delta-Sigma Data Converters*. Jone Wiley & Sons, Inc., Publication, 2005.

[6] J. Uhlig, R. Schüffny, H. Neubauer, J. Hauer, J. Haase „A Low-Power Continous-Time Incremental 2nd-Order-Mash SD-Modulator for a CMOS Imager” *Electronic Circuits and Systems, IEEE International Conference on*, Medina, 2009

[7] http://exist.gsfc.nasa.gov/design/het/czt/

Rapid Prototyping mit FAUmachine am Beispiel einer VHDL-PCI-Soundkarte

Stefan Potyra, Matthias Sand, Volkmar Sieh, Dietmar Fey
Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl für Rechnerarchitektur
Martensstr. 3, 91058 Erlangen
Email: {Stefan.Potyra, Matthias.Sand, Volkmar.Sieh, Dietmar.Fey}@informatik.uni-erlangen.de

Zusammenfassung

Das Koppeln der virtuellen Maschine FAUmachine mit einem VHDL-Interface ermöglicht eine hochperformante Kosimulation. Dies unterstützt den Systementwurf mittels Rapid Prototyping, z.B. für die Entwicklung von kompletten Baugruppen für einen PC. Das vorgestellte Verfahren bietet durch Abstraktion eine hohe Simulationsperformanz, ist jedoch dennoch detailliert genug, um Aussagen über konkrete Hardware-Modelle treffen zu können.

1. Eigene Vorarbeiten und verwandte Arbeiten

Zur Simulation von Rechensystemen existieren lediglich für bestimmte Teilaspekte analytische Modelle, wie beispielsweise für Netzwerkinteraktionen [22, 26, 3, 13] oder Speicherhierarchien [16, 12, 5]. Diesen Modellen ist gemeinsam, dass sie jeweils auf die zu evaluierenden Aspekte reduziert und von der Nachbildung realer Hardware entsprechend abstrahiert sind. Ein solcher Ansatz scheidet daher zur detaillierten Simulation eines Rechensystems aus.

Neben analytischen Modellen können auch konkrete Hardwarebeschreibungen in geeigneten Sprachen (Hardware Description Languages – HDL) basierend auf diskreter, ereignisorientierter Simulation evaluiert werden. Hierfür existieren sowohl kommerzielle Werkzeuge, wie beispielsweise ModelSim [15, 7], als auch frei verfügbare Implementationen, beginnend bei HDL-Analysewerkzeugen [25] bis hin zu auf Time Warp basierenden Simulationsumgebungen [19]. Die Hauptprobleme bei der Verwendung solcher Simulatoren sind aber, dass ganze Systeme nur mit sehr großem konzeptionellem Aufwand modelliert werden können, insbesondere aber, dass die Simulation extrem ineffizient und daher kaum praktikabel ist.

Für virtuelle Maschinen stellt die Abstraktion des Detailgrades, zum Beispiel der Simulation von gesamten Bustransaktionen anstelle einer zyklengenauen Simulation eine notwendige Voraussetzung dar, um eine hohe Simulationsgeschwindigkeit zu erreichen. Das sogenannte *Transaction Level Modeling* bietet ähnliche Ansätze im Bereich der Hardware-Beschreibungssprache *SystemC* [17]. Die Steigerung der Performanz, aber auch das Entkoppeln von Hardware-Komponenten von der tatsächlichen Busimplementierung sind hierbei die Ziele.

Im Bereich der virtuellen Umgebungen gibt es neben FAUmachine [6, 20] bereits eine Vielzahl weiterer Emulatoren (QEMU [4], Bochs [21], z.T. VMware [24], etc.) und Virtualisierungslösungen (Xen [2], User-Mode-Linux [23], z.T. VMware [24], VirtualBox [11], KVM [14], etc.), die – mehr oder weniger – vollständige Rechensysteme effizient simulieren können. Keiner der hier genannten Vertreter ist jedoch bisher dazu in der Lage, eine deterministische Simulation bereitzustellen. Ebenfalls fehlt den genannten Vertretern die Fähigkeit, automatisierte Testläufe durchzuführen. Lediglich KVM stellt hierfür eine rudimentäre Implementierung bereit.

FAUmachine ist eine quelloffene Implementierung einer virtuellen Maschine, welche gebräuchliche PC-Hardware simulieren kann. FAUmachine kann eine Vielzahl unterschiedlicher Betriebssysteme in unmodifizierter Form ablaufen lassen. Durch den Einsatz eines Just-in-Time-Compilers [8] kann eine hohe Simulationsgeschwindigkeit erreicht werden. Im Gegensatz zu anderen Virtualisierungslösungen sind die Komponenten von FAUmachine stark an der zugrunde liegenden, realen Hardware orientiert. FAUmachine kann interaktiv benutzt werden, bietet aber ebenfalls

die Möglichkeit, automatisierte Systemtests deterministisch durchzuführen [18]. Die Möglichkeit, Fehler zu injizieren [9], unterstützt das Entwickeln von fehlertoleranten Systemen.

2. Modellierung innerhalb von FAUmachine

FAUmachine verfolgt den Ansatz der strikten Trennung von Komponenten, um das Simulationssystem modular zu halten: Keine Komponente verfügt über Wissen von anderen Komponenten und gleichermaßen ist die Struktur des gesamten zu simulierenden Systems keiner Komponente bekannt. Signale oder Signalbündel bilden die Schnittstelle zwischen Komponenten. Die Änderung eines Signals in einer Komponente hat das Aufrufen einer Callback-Funktion in jeder verbundenen Komponente zur Folge. Dies geschieht unmittelbar; es verstreicht hierbei keine Simulationszeit.
Manche Komponenten erfordern jedoch zwingend zeitliche Verzögerungen, wie beispielsweise der *Programmable Interval Timer*. Hierfür stellt der Scheduler von FAUmachine eine Funktion zur Verfügung, mit der eine Callback-Funktion zeitlich verzögert aufgerufen werden kann. Somit kann diskrete, ereignisorientierte Simulation innerhalb von FAUmachine nachgeahmt werden.
Um nebenläufige, sequentielle Vorgänge leichter zu modellieren, existieren logische Prozesse. Logische Prozesse werden nach dem Prinzip von Koroutinen realisiert: Ein Prozess ist eine Funktion innerhalb einer Komponente, welche die Kontrolle abgeben kann, und zu einem späteren Zeitpunkt wieder als lauffähig markiert werden kann. Der Scheduler führt sämtliche lauffähigen logischen Prozesse der Reihe nach aus.
Die Komponenten innerhalb von FAUmachine werden hierarchisch modelliert: „Architectures“ bilden die unterste Ebene. Sie sind ähnlich zu Architectures der Beschreibungssprache VHDL: Eine Architecture kann von einer Komponente ein- oder mehrmals instanziert werden. Architectures sind in C realisiert. Der C-Präprozessor wird zum Instanzieren genutzt. Ein Architecture kann somit in mehreren Komponenten wiederverwendet werden.
„Chips“ bilden die nächste Hierarchiestufe. Chips werden ebenfalls in C implementiert. Im Gegensatz zu Architectures können Chips auch zur Laufzeit instanziert werden. Die Schnittstellen eines Chips, also die Eingangs-, Ausgangs- und bidirektionalen Signale, werden in einem vereinfachten Format beschrieben. Aus diesem Format wird automatisch der passende C-Header sowie eine VHDL-Schnittstellenbeschreibung generiert.
„Komponenten“ entsprechen handelsüblichen PC-Teilen, wie beispielsweise einer CPU, einem Mainboard oder einem Festplattenlaufwerk. Komponenten setzen sich aus Chips zusammen. Die Modellierung erfolgt mit dem EDA-Werkzeug `gschem` aus der gEDA-Suite [10]. Aus den graphisch modellierten Schaltplänen wird eine VHDL-Beschreibung generiert. Auch die Struktur des gesamten zu simulierenden Systems wird in VHDL beschrieben.

3. Kopplung der VHDL-Simulation mit einer virtuellen Maschine

Die Hardware-Komponenten, die FAUmachine simulieren kann, sind in C modelliert. Die Struktur des zu simulierenden Systems wird in VHDL beschrieben. Der VHDL-Compiler von FAUmachine übersetzt diese Beschreibung in einen schnell interpretierbaren Zwischencode. Der VHDL-Interpreter wertet diesen aus. Für *Entities* ermittelt der VHDL-Compiler, ob diese als in C modellierte Komponenten vorliegen. Hierfür wird die im vorherigen Abschnitt erwähnte VHDL-Schnittstellenbeschreibung verwendet. Handelt es sich um solche Komponente, so nutzt der Interpreter eine Zwischenschicht zur Instanzierung.
Der VHDL-Interpreter verwendet eine diskrete, ereignisorientierte Simulation gemäß dem VHDL-Standard [1]. Zur Kopplung teilt der VHDL-Compiler Signale entsprechend ihrer Nutzung in drei Kategorien ein: Zum einen existieren Signale, welche nur im VHDL Kontext benutzt werden. Diese werden alleine durch den Interpreter von FAUmachine behandelt. Signale, die nur von in C implementierten Komponenten genutzt werden, werden über eine Vermittlungsschicht

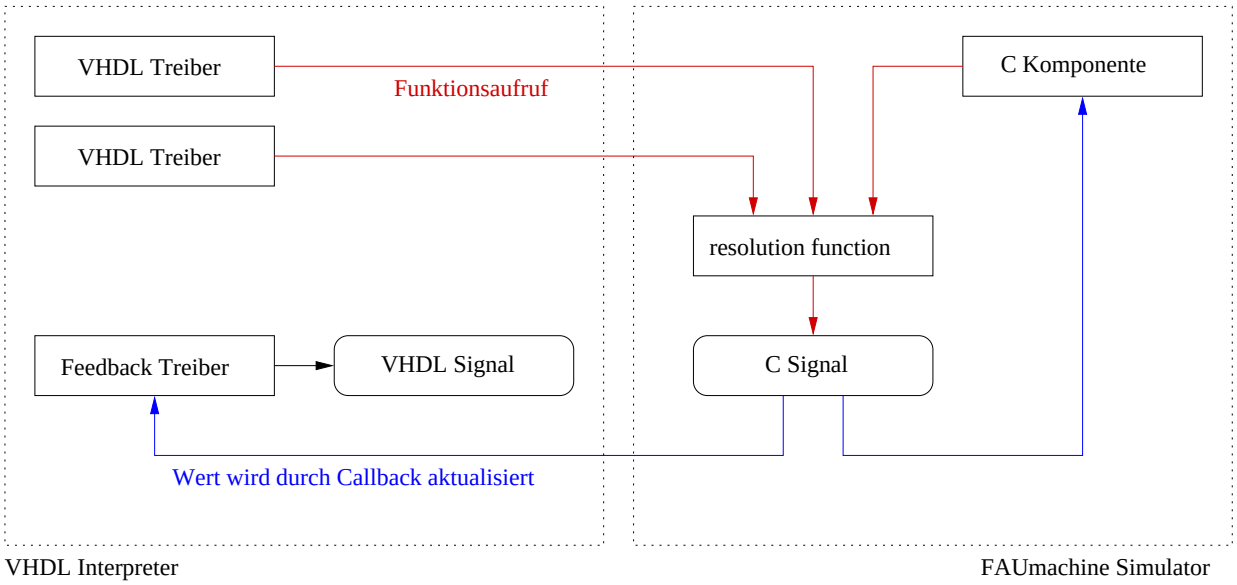


Abbildung 1: Gemeinsam genutzte Signale

erzeugt und verwaltet. Schließlich existieren Signale, welche von in C modellierten Komponenten und aus dem VHDL-Kontext gemeinsam genutzt werden. Diese bedürfen einer besonderen Behandlung, welche in Abbildung 1 dargestellt ist: Für jedes gemeinsam genutzte Signal wird ein VHDL-Signal und ein in C implementiertes Signal erzeugt. Sämtliche VHDL-Treiber geben ihre Änderungen nur an das C-Signal weiter. Die Auflösung des Wertes erfolgt durch das C-Signal. Ändert sich der Wert, so wird dieser an einen speziellen Feedback-Treiber des VHDL-Interpreters übermittelt. Dieser Treiber ist der einzige Treiber, der für den Signalwert des VHDL-Signals verantwortlich ist.

4. Methode zur schnellen Simulation

Signale oder Signalbündel verbinden Komponenten. Die Änderung eines Signals in einer Komponente hat das Aufrufen einer Callback-Funktion in jeder verbundenen Komponente zur Folge.
Die C-Implementierung von FAUmachine kennt nicht nur Signale für einfache Typen (bspw. `boolean` oder `std_logic`), sondern auch Signale, welche komplette Bus-Transaktionen modellieren. Dies führt bereits zu einem enormen Geschwindigkeitsgewinn. Zusätzlich existiert eine weitere Optimierung im Hinblick auf Bustransaktionen: Wird eine Operation durchgeführt, so resultiert dies einmalig in einem Callback in jeder Komponente, die mit dem Bus verbunden ist. Der Rückgabewert des Callbacks gibt an, ob eine Komponente für eine Bustransaktion zuständig ist. Diese Information wird zwischengespeichert. Damit kann ein Busmaster in allen folgenden Buszugriffen, welche dieselbe Zieladdresse besitzen, direkt auf die angesprochene Komponente zugreifen, ohne dass die Callbacks der anderen verbundenen Komponenten dazu aufgerufen werden müssen.
Bei der Anbindung einer VHDL-Komponente an den PCI-Bus stellt die hohe Taktfrequenz ein Problem dar. Folgende Verfahren gewährleisten dennoch eine hohe Performanz: Zum einen erfolgt die Abbildung von Bus-Zyklen nicht mit exakter Zeitauflösung. Im Gegenteil: Die Simulationszeit schreitet während eines Bus-Zykluses gar nicht voran. Lediglich ein Anstoßen des Interpreters nach Änderung des Taktsignals gewährleistet zeitliche Kausalität. Dies stellt jedoch keine Einschränkung bei der Simulation von synthetisierbarem VHDL dar.
Zum anderen wird das Taktsignal lediglich *während* eines Bus-Zugriffs simuliert. Dies steigert die Simulationsgeschwindigkeit gewaltig, führt jedoch zu folgender Einschränkung: Die Zustandsänderungen in der Hardwarebeschreibung müssen in direkter Abhängigkeit von Bus-

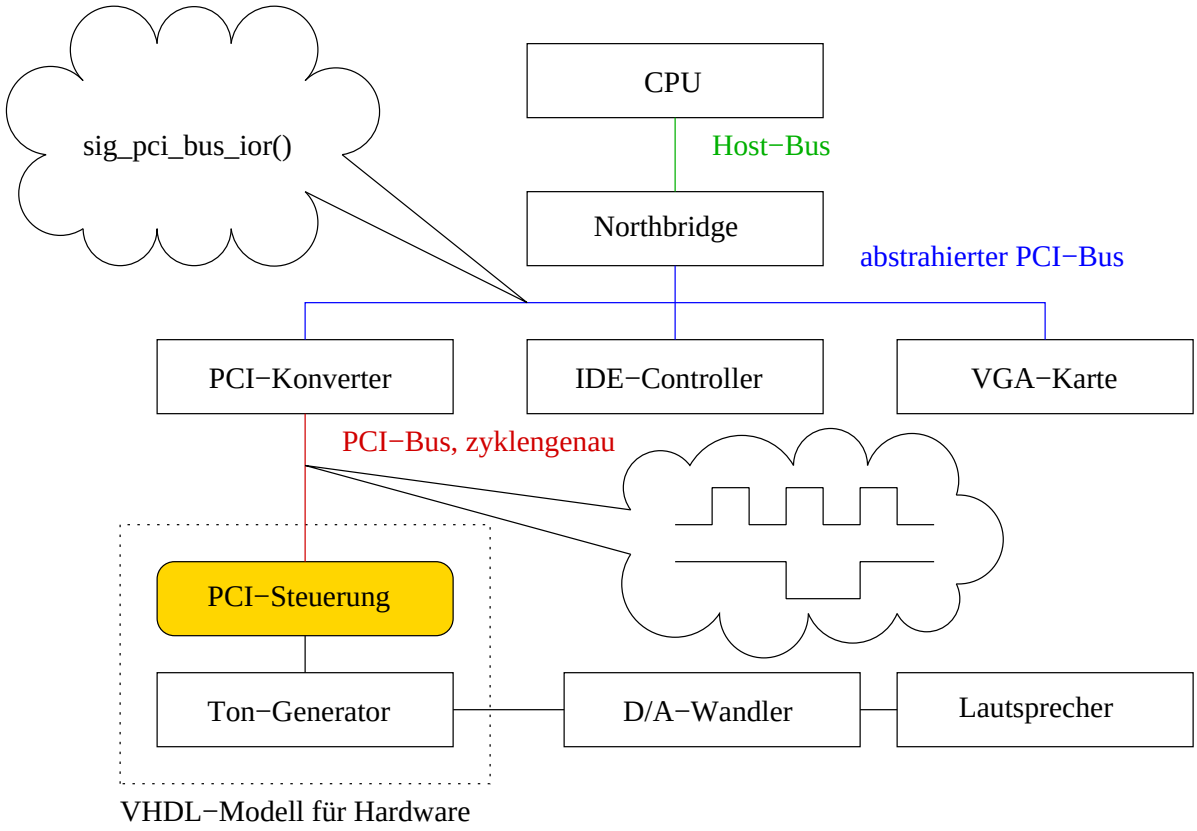


Abbildung 2: Simuliertes Modell

Zugriffen erfolgen. Ein Zähler, welcher als Zählimpuls das Taktsignal verwendet, würde folglich nicht korrekt simuliert.

5. Proof-of-concept

5.1. Hardware-Modell

Um die Brauchbarkeit der vorgestellten Methode zu zeigen, wurde eine PCI-Soundkarte in VHDL modelliert. Die Karte besteht zum einen aus einer PCI-Steuerung, die die Signale des PCI-Busses dekodiert. Darüber hinaus besitzt die PCI-Steuerung die zur Konfiguration notwendigen Register. Ebenfalls sind Register zur Ansteuerung von 16 IO-Ports vorhanden. Die IO-Ports der PCI-Steuerung sind mit einem Ton-Generator verbunden. Dieser kann die Frequenzen für unterschiedliche Noten erzeugen. An den Ton-Generator ist ein Digital-Analog-Wandler angeschlossen. Der Ausgang dieses Wandlers kann mit handelsüblichen Lautsprecherboxen verbunden werden. Sowohl die PCI-Steuerung als auch der Ton-Generator wurden in VHDL modelliert. Die Funktionsfähigkeit des entwickelten Modell wurde auf einer PCI-Prototyp Karte mit einem CPLD des Typs *iM4A3-512/160* der Firma Lattice Semiconductors getestet.

5.2. Simulation im Kontext von FAUmachine

Abbildung 2 zeigt das für FAUmachine verwendete Modell: Innerhalb der virtuellen Maschine werden abstrahierte Signale für den Host- und den PCI-Bus verwendet. Der PCI-Konverter wandelt diese in zyklengenaue Signale des PCI-Busses um. Auf diese Signale kann von VHDL aus zugegriffen werden.
Das entwickelte Hardware-Modell wurde in PCI-Steuerung und Ton-Generator aufgeteilt: Für die PCI-Steuerung wurde **exakt** das zuvor entwickelte VHDL-Modell verwendet. Um zu zeigen, dass auch C-Komponenten aus dem Kontext von VHDL erzeugt und angesteuert werden können,

wurde der Tongenerator in C modelliert. Das Interface des Ton-Generators stimmt hierbei mit dem der VHDL-Beschreibung überein. Zum automatisierten Testen wurde ein Test-Skript erstellt, das einen Selbsttest der PCI-Steuerung durchführt und anschließend ein Lied abspielt.
Durch die vorgestellte Methode, das PCI-Taktsignal nur bei Buszugriffen, welche die Soundkarte betreffen, zu simulieren, wird die Performanz durch die zusätzliche VHDL-Simulation nur unwesentlich beeinträchtigt. Lediglich bei Zugriffen auf die VHDL-Karte muss der abstrahierte Zugriff zyklengenau abgebildet werden. Um den Overhead zu messen, wurde die PCI-Steuerung als C-Komponente erneut implementiert. Der automatisierte Testlauf wurde anschließend sowohl mit der VHDL-Implementierung als auch mit der C-Implementierung ausgeführt und die benötigte Zeit gemessen:

Simulationsmodus	**C**	**VHDL**	**Overhead**
interaktiv	126,9 s	127,4 s	0,4 %
deterministisch	92,0 s	94,3 s	2,4 %

Die VHDL-Simulation erzeugt einen durchschnittlichen Overhead von 0,4-2,4 % im Vergleich zur Simulation durch in C implementierte Komponenten. Eine interaktive Benutzung der virtuellen Maschine ist somit weiterhin möglich.

6. Zusammenfassung und Ausblick

Die vorgestellte Verfahrensweise ist in der Lage, VHDL-Hardware-Modelle im Kontext einer virtuellen Maschine mit äußerst hoher Performanz zu simulieren. Nach unserem Kenntnisstand existiert kein anderes Verfahren, welches eine HDL-Simulation im Kontext eines vollständiges Systems mit ähnlicher Geschwindigkeit durchführen kann. Forschungsbedarf besteht darin, eine Methode zu finden, die auch für taktabhängige Hardwaremodelle, wie beispielsweise Zähler, sowohl hohe Performanz als auch korrekte Simulation gewährleistet.

Literatur

[1] 1076-2002, IEEE STD.: *IEEE Standard VHDL Language Reference Manual*. Product No.: SH94983-TBR, 2002.

[2] BARHAM, PAUL, BORIS DRAGOVIC, KEIR FRASER, STEVEN HAND, TIMOTHY L. HARRIS, ALEX HO, ROLF NEUGEBAUER, IAN PRATT und ANDREW WARFIELD: *Xen and the art of virtualization.* In: *SOSP*, Seiten 164–177, 2003.

[3] BAUER, DAVID, GARRETT YAUN, CHRISTOPHER D. CAROTHERS, MURAT YUKSEL und SHIVKUMAR KALYANARAMAN: *Simulation of large scale networks III: ROSS.Net: optimistic parallel simulation framework for large-scale internet models.* In: *WSC '03: Proceedings of the 35th conference on Winter simulation*, Seiten 703–711. Winter Simulation Conference, 2003.

[4] BELLARD, FABRICE: *QEMU, a fast and portable dynamic translator*. In: *ATEC'05: Proceedings of the USENIX Annual Technical Conference 2005 on USENIX Annual Technical Conference*, Seiten 41–46, Berkeley, CA, USA, 2005. USENIX Association.

[5] FARIAS, CLÉVER RICARDO GUAREIS DE, LUÍS FERREIRA PIRES, WANDERLEY LOPES DE SOUZA und CÉLIO ESTEVAN MORÓN: *Specification and Validation of a Real-Time Parallel Kernel Using LOTOS.* In: *MASCOTS*, Seiten 7–14. IEEE Computer Society, 2001.

[6] FAUMACHINE TEAM: *FAUmachine*. URL: http://www.FAUmachine.org/, 2003–2010.

[7] HATNIK, UWE und SVEN ALTMANN: *Using ModelSim, Matlab/Simulink and NS for Simulation of Distributed Systems*. In: *PARELEC '04: Proceedings of the international conference on Parallel Computing in Electrical Engineering*, Seiten 114–119, Washington, DC, USA, 2004. IEEE Computer Society.

[8] HÖXER, H.-J., M. WAITZ und V. SIEH: *Advanced virtualization techniques for FAUmachine*. In: SPENNEBERG, R. (Herausgeber): *11th International Linux System Technology Conference, Erlangen, Germany, September 7-10, 2004*, Seiten 1–12, 2004.

[9] HÖXER, H.-J., M. WAITZ und V. SIEH: *Fast Simulation of Stuck-At and Coupling Memory Faults Using FAUmachine*. In: *Supplement to Proc. HASE 2005: International Symposium on High Assurance Systems Engineering (Ninth IEEE International Symposium on High Assurance Systems Engineering Heidelberg, Germany 12-14 October 2005)*, Seiten 1–2, Oktober 2005.

[10] HVEZDA, ALES et al.: *gEDA Project*. URL: http://www.gpleda.org/, 2009.

[11] INNOTEK GMBH: *VirtualBox*. URL: http://www.virtualbox.org/, 2008.

[12] IYER, RAVI R.: *On Modeling and Analyzing Cache Hierarchies using CASPER.* In: *MASCOTS*, Seiten 182–187. IEEE Computer Society, 2003.

[13] KAMAL, AHMED E.: *Discrete-time modeling of TCP Reno under background traffic interference with extension to RED-based routers.* Perform. Eval., 58(2-3):109–142, 2004.

[14] KIVITY, AVI, YANIV KAMAY, DOR LAOR, URI LUBLIN und ANTHONY LIGUORI: *kvm: the Linux virtual machine monitor*. In: *OLS '07: The 2007 Ottawa Linux Symposium*, Seiten 225–230, July 2007.

[15] MENTOR GRAPHICS CORP.: *ModelSim – a comprehensive simulation and debug environment for ASIC and FPGA design*. URL: http://www.model.com, 2007.

[16] OVER, ANDREW, PETER STRAZDINS und BILL CLARKE: *Cycle Accurate Memory Modelling: A Case-Study in Validation*. In: *MASCOTS '05: Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Seiten 85–96, Washington, DC, USA, 2005. IEEE Computer Society.

[17] PASRICHA, SUDEEP: *Transaction level modeling of SoC with SystemC 2.0*. In: *Synopsys User Group Conference (SNUG)*, 2002.

[18] POTYRA, S., V. SIEH und M. DAL CIN: *Evaluating fault-tolerant system designs using FAUmachine*. In: *EFTS '07: Proceedings of the 2007 workshop on Engineering fault tolerant systems*, Seite 9, New York, NY, USA, 2007. ACM.

[19] RADHAKRISHNAN, RADHARAMANAN, DALE E. MARTIN, MALOLAN CHETLUR, DHANANJAI MADHAVA RAO und PHILIP A. WILSEY: *An Object-Oriented Time Warp Simulation Kernel*. In: *ISCOPE '98: Proceedings of the Second International Symposium on Computing in Object-Oriented Parallel Environments*, Seiten 13–23, London, UK, 1998. Springer-Verlag.

[20] SAND, MATTHIAS, STEFAN POTYRA und VOLKMAR SIEH: *Deterministic High-Speed Simulation of Complex Systems Including Fault-Injection*. In: *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2009.

[21] SOURCE FORGE: *Bochs IA-32 Emulator Project*. URL: http://bochs.sourceforge.org/, 2001.

[22] SZYMANSKI, BOLESLAW K., YU LIU und RASHIM GUPTA: *Parallel Network Simulation under Distributed Genesis*. In: *PADS '03: Proceedings of the seventeenth workshop on Parallel and distributed simulation*, Seiten 61–68, Washington, DC, USA, 2003. IEEE Computer Society.

[23] USER MODE LINUX CORE TEAM: *User Mode Linux HOWTO*. URL: http://user-mode-linux.sourceforge.net/UserModeLinux-HOWTO.html, 2004.

[24] VMWARE INC.: *VMware*. URL: http://www.vmware.com/, 2001.

[25] WILSEY, P. A., D. E. MARTIN und K. SUBRAMANI: *8.4: SAVANT/TyVIS/WARPED: Components for the Analysis and Simulation of VHDL*. In: *IVC-VIUF '98: Proceedings of the International Verilog HDL Conference and VHDL International Users Forum*, Seiten 195–201, Washington, DC, USA, 1998. IEEE Computer Society.

[26] YAUN, GARRETT, CHRISTOPHER D. CAROTHERS und SHIVKUMAR KALYANARAMAN: *Large-Scale TCP Models Using Optimistic Parallel Simulation.* In: *PADS '03: Proceedings of the seventeenth workshop on Parallel and distributed simulation*, Seiten 153–162, Washington, DC, USA, 2003. IEEE Computer Society.

Ein retargierbarer, hochperformanter ISA-Simulator in Java

Marco Kaufmann, Thomas B. Preußer, Rainer G. Spallek

Institut für Technische Informatik
Technische Universität Dresden
D-01062 Dresden
{marco.kaufmann, thomas.preusser,
rainer.spallek}@tu-dresden.de

Kurzfassung

In dieser Arbeit wird *Jahris* vorgestellt, ein hochperformanter, retargierbarer ISA-Simulator, welcher vollständig in Java realisiert ist. Seine hohe Simulationsgeschwindigkeit wird durch die dynamische Compilierung der Zielapplikation in Java-Bytecode, große Ausführungseinheiten sowie Hot-Spot-Techniken des dynamischen Simulationscompilers erreicht. Trotz großer Ausführungseinheiten erfolgt die Simulation dabei befehlsgenau. Nichtinvasive Breakpoints sowie schrittweise Befehlsausführung werden unterstützt. Jahris verfügt zudem über eine dedizierte ISA-Beschreibungssprache, welche auf die rasche Modellierung von Zielarchitekturen für den Simulator sowie auf hohe Simulationsgeschwindigkeiten abzielt. Sie ist derart flexibel, dass mit ihr ein breites Spektrum von Zielarchitekturen beschrieben werden kann, einschließlich RISC-, CISC- und VLIW-artiger Architekturen. Das hervorstechendste Merkmal von Jahris ist jedoch die Verwendung der Java-Laufzeitumgebung als virtueller Hostplattform für die Simulation. Dies macht es zu einem dynamisch compilierenden und dennoch plattformunabhängigen Simulator. Zudem profitiert der Just-in-Time - Simulationscompiler direkt von den hochentwickelten Optimierungstechniken und vom plattformspezifischen Know-How moderner JVM-Implementierungen, wodurch ein eigener Codeoptimierer überflüssig und der dynamisch generierte Code dennoch hochoptimiert ausgeführt wird. Bisher modelliert und mit Jahris getestet wurden die DLX-, die i8086-, die ARMv4- sowie die 32-Bit-PowerPC Architektur. Anhand von Benchmarktests hat sich dabei gezeigt, dass trotz Retargierbarkeit, Plattformunabhängigkeit und befehlsgenauer Beobachtbarkeit bei der Simulation bis zu 78 Prozent der Performance von QEMU erreicht werden.

1 Einführung

ISA-Simulatoren sind für den Entwurfsprozess von Befehlssatzarchitekturen (*Instruction Set Architectures*, ISAs) zum unerlässlichen Werkzeug geworden. Sie ermöglichen einerseits die fortwährende Evaluation von Entwurfsentscheidungen im Rahmen des ISA-Entwurfsprozesses, sowie andererseits die Entwicklung und den Test von Software bereits vor Verfügbarkeit einer Hardwareimplementierung der Zielarchitektur. Hauptanforderungen an ISA-Simulatoren sind hohe Simulationsgeschwindigkeit sowie die befehlsgenaue Beobachtbarkeit konsistenter Architekturzustände. Die Berechnung konsistenter Zustände an den Grenzen der einzelnen Zielinstruktionen verursacht jedoch Overhead bei der Simulation. Hohe Simulationsgeschwindigkeit und feingranulare Beobachtbarkeit sind somit gegensätzliche Anforderungen. Weitere Designziele für ISA-Simulatoren

sind Portierbarkeit und Retargierbarkeit. Diese ermöglichen die Wiederverwendung eines Simulators und seines inhärenten Know-Hows für verschiedene Zielarchitekturen und Hostplattformen, was besonders aufgrund der steigenden Anzahl potentieller Zielarchitekturen und im Hinblick auf die Verwendung als ISA-Entwurfswerkzeug von großer Bedeutung ist. Jedoch vermindert auch die Umsetzung dieser Anforderungen die Simulationsperformance.

Dem in dieser Arbeit vorgestellten hochperformanten, retargierbaren, befehlsgenauen ISA-Simulator *Jahris* gelingt es, die genannten Anforderungen an die ISA-Simulation in sich zu vereinen und ihre wechselseitigen Konflikte weitgehend aufzuheben. Abschnitt 2 bietet einen kurzen Überblick zum aktuellen Stand auf dem Gebiet der ISA-Simulation. Abschnitt 3 beschreibt die Simulationsengine von Jahris. In Abschnitt 4 werden Benchmarkergebnisse und ein Vergleich mit dem Emulator QEMU aufgeführt, bevor Abschnitt 5 eine abschließende Zusammenfassung der im Rahmen dieser Arbeit erzielten Ergebnisse gibt.

2 Weitere Arbeiten auf dem Gebiet

Ziel der dynamisch compilierenden ISA-Simulation ist es, die hohe Performance der statisch compilierenden mit der Flexibilität der interpretierenden Simulation zu vereinen [1]. Im Gegensatz zur statisch compilierenden Simulation wird die Zielapplikation nicht vollständig und vorab der Laufzeit geholt, dekodiert und compiliert, sondern dies geschieht für ihre einzelnen Codeblöcke zur Laufzeit jeweils unmittelbar vor der ersten Ausführung. Die dekodierten Blöcke werden zur späteren Wiederverwendung in einem Simulationscache zwischengespeichert. Unterstützung für laufzeitdynamischen Code lässt sich durch Verwerfen der entsprechenden Blöcke aus dem Cache realisieren. Die Implementierung eines dynamisch compilierenden ISA-Simulators mit pipelinegenauer Modellierung der Zielarchitektur und taktgenauer Simulation wird von G. Braun et al. [2] beschrieben. Sie basiert auf einer LISA-Spezifikation [3] der Mikrooperationen der Zielarchitektur. Ein Hybrider Ansatz der statisch- und dynamisch compilierende Simulation kombiniert wird von Reshadi et al. [4] verfolgt. Diese praktizieren eine statische Übersetzung der Zielapplikation in nativen Code vorab der Laufzeit der Simulation. Lediglich im Rahmen dieser statischen Analyse nicht erkannte Codebereiche oder laufzeitdynamisch modifizierte Codeblöcke werden dynamisch compiliert. Jones und Topham schlagen große Übersetzungseinheiten für die dynamische Compilierung der Zielapplikation in Host-Maschinencode vor [5]. Dies erhöht die Simulationsperformance, da zum einen mehr Laufzeit für die eigentliche Befehlsausführung und weniger für den Rest der Simulationsschleife aufgewendet wird, und zum anderen größere Übersetzungseinheiten bessere Möglichkeiten für die Codeoptimierung eröffnen. Dynamisch compilierende Simulationstechniken werden auch vom Full-System-Emulator QEMU [6] angewendet. Dieser erreicht eine sehr hohe Performance. Sein Anwendungsgebiet liegt jedoch in der Emulation anstatt der Simulation, die feingranulare Beobachtbarkeit konsistenter Architekturzustände für den Benutzer spielt daher keine Rolle.

3 Die Simulations-Engine

3.1 Ablauf der Simulation

Der Ablauf der Simulation ist in Abb. 1 schematisch dargestellt. Ein Zyklus der Simulationsschleife beginnt am *Checkpoint*, an welchem sich die Zielarchitektur in einem konsistenten Zustand befindet. Dieser wird durch die Ausführung der jeweils nächsten *Ausführungseinheit* der Zielapplikation in einen Folgezustand transformiert. Die jeweilige Ausführungseinheit wird durch die *Instruction-Fetch-Adresse* (IFA) bestimmt und kann in interpretierender Form als *Mikrooperationssequenz* (μOP Seq.) oder in compilierter Form als *Translated Function* vorliegen. Sie wird

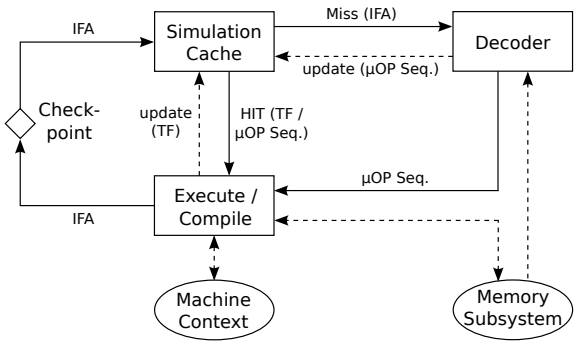


Abbildung 1: Ablauf der Simulation

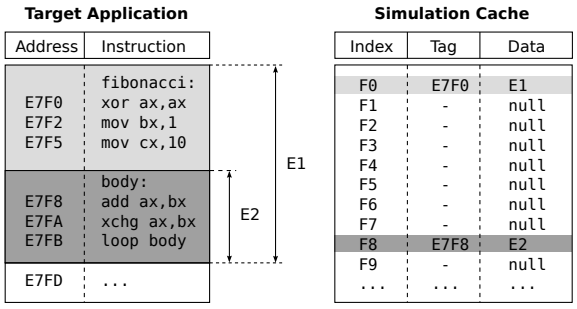


Abbildung 2: Überlappende Ausführungseinheiten (E1 und E2)

Aus Gründen der Übersicht wurde die Zielapplikation in Abb. 2 als Assemblerquelltext dargestellt, liegt tatsächlich jedoch als Maschinencode im Speicher der Zielarchitektur vor. Zudem beträgt die standardmäßige Cachegröße 1M- anstatt nur 256 Einträge wie durch die Abbildung suggeriert.

zunächst im *Simulationscache* nachgeschlagen. Im Falle eines Cache-Miss wird der Befehlsdekoder aufgerufen, um die Zielinstruktionen an der aktuellen IFA im Speicher der Zielarchitektur in eine entsprechende Mikrooperationssequenz zu dekodieren. Diese wird im Simulationscache zwischengespeichert und ersetzt sich selbst durch eine äquivalente TF, wenn die Anzahl ihrer Ausführungen den konfigurierbaren *Compilerschwellwert* überschreitet. Die aktuelle Ausführungseinheit gibt zudem die nächste IFA zurück. Der verwendete Simulationscache ist ein Direct-Mapped-Cache. Er wird durch den niederwertigen Teil der IFA indiziert, während die vollständige IFA als Tagwort dient. Die aktuelle Implementierung erlaubt IFA-Breiten bis zu 64 Bit.

Der Dekoder ist verantwortlich für das Holen von Ausführungseinheiten der Zielapplikation und ihre Dekodierung in funktional äquivalente Mikrooperationssequenzen. Dies geschieht einzelinstruktionsweise. Die dabei erhaltenen Mikrooperationssequenzen werden dann entsprechend den in der jeweiligen Ausführungseinheiten enthaltenen Zielinstruktionen zusammengesetzt. Eingabe für den Dekoder ist die IFA, welche den Einsprungpunkt der Ausführungseinheit markiert, sowie der Speicherbus der Zielarchitektur, von welchem die Befehlsworte zu holen sind. Der Dekoder darf den Architekturzustand darüberhinaus weder lesen noch modifizieren, da aufgrund der getrennt von der Befehlsausführung erfolgenden Dekodierung die für ein Befehlswort generierte Mikrooperationssequenz unabhängig vom aktuellen Architekturzustand stets dieselbe sein muss.

3.2 Ausführungseinheiten

Die kleinstmögliche Ausführungseinheit für nicht-zyklengenaue ISA-Simulatoren ist eine Zielinstruktion. Wie in Abschnitt 2 erwähnt, führen größere Ausführungseinheiten zu einer signifikant höheren Simulationsperformance. Ähnlich wie QEMU verwendet Jahris blockartige Ausführungseinheiten, welche sich von einem Einsprungpunkt (Label) bis einschließlich zur nächsten Kontrolltransfer-Zielinstruktion erstrecken. Solche Blöcke dürfen beliebig viele Zielinstruktionen umfassen. Da sie zudem weitere Labels enthalten können, können Ausführungseinheiten sich gegenseitig überlappen. Ein Beispiel hierfür ist in Abb. 2 dargestellt. Überlappende Ausführungseinheiten im Simulationscache gehen mit dem mehrfachen Holen und Dekodieren der sich überlappenden Teile einher. Zwar verursacht dies zusätzlichen Overhead, jedoch wird auch die Länge der einzelnen Ausführungseinheiten dadurch maximiert. Zudem würden sich gegenseitig ausschließende Ausführungseinheiten entweder Wissen über alle Einsprungpunkte und somit eine statische Codeanalyse, oder das nachträgliche Verwerfen bereits dekodierter Ausführungseinheiten bei Aufdecken einer Überlappung erfordern. In beiden Fällen überwiegen die damit verbundenen Kosten jedoch den Nutzen.

3.3 Hot-Spot Simulationscompiler

Die im Simulationscache zwischengespeicherten Mikrooperationssequenzen protokollieren die Anzahl ihrer Ausführungen. Überschreitet diese den konfigurierbaren Compilerschwellwert, wird die entsprechende Ausführungseinheit in eine TF compiliert, welche die ursprüngliche Mikrooperationssequenz im Simulationscache ersetzt. Hierbei wird heuristisch von der Annahme ausgegangen, dass Ausführungseinheiten, welche in der Vergangenheit häufig ausgeführt wurden (*Hot Spots*), auch in Zukunft noch häufig ausgeführt werden sodass sich die für die Compilierung benötigte Laufzeit amortisiert. Die Übersetzung von Ausführungseinheiten in nativen Code der tatsächlichen Hostplattform geschieht in zwei Phasen. Die erste erfolgt durch den Simulationscompiler von Jahris, welcher Mikrooperationssequenzen in nativen Code der virtuellen Hostplattform - in Java-Bytecode - transformiert. Dieser wird jeweils durch eine TF in Form einer eigenen Java-Klasse repräsentiert, die dynamisch in die Java-Laufzeitumgebung nachgeladen und einmalig instantiiert wird. Die zweite Phase erfolgt durch die JVM. Moderne JVM Implementierungen führen wiederum den Java-Bytecode zunächst interpretierend aus und erstellen gleichzeitig ein Laufzeitprofil, um schließlich dessen Hot-Spots zu compilieren und zu optimieren. Translated Functions werden hiernach optimiert und als Maschinencode der tatsächlichen Hostplattform ausgeführt.

Dieses zweistufige Verfahren verursacht einigen Overhead bei der dynamischen Compilierung, da der vom Simulationscompiler generierte Bytecode in Java-Klassen verpackt werden muss, welche von der JVM zunächst geparst und verifiziert werden. Dennoch birgt es eine ganze Reihe von Vorteilen. Zunächst stellt die JVM eine virtuelle Zielmaschine für den dynamischen Simulationscompiler dar und ermöglicht so trotz dynamischer Compilierungstechniken die Lauffähigkeit von Jahris auf allen Hostsystemen, für die eine entsprechende Java-Laufzeitumgebung verfügbar ist. Weiterhin nutzt der Simulationscompiler die hochentwickelten Optimierungstechniken und das plattformspezifische Know-How moderner JVM-Implementierungen aus, wodurch ein eigener Codeoptimierer überflüssig und der dynamisch generierte Code dennoch hochoptimiert ausgeführt wird. Selbst die meisten Javacompiler sind nichtoptimierend und überlassen die gesamte Codeoptimierung der JVM.

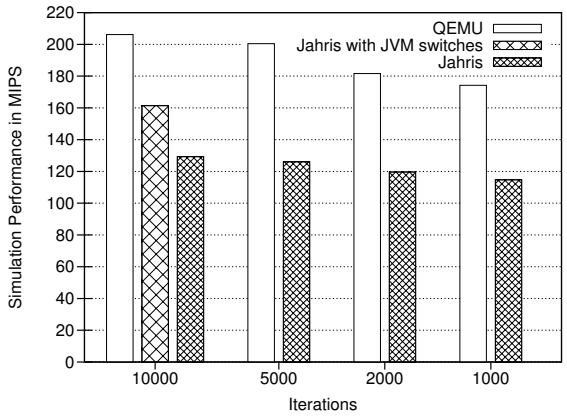
3.4 Breakpoints, Stepping und asynchroner Halt der Simulation

Am Checkpoint der Simulationsschleife wird der Status des Break-Flags überprüft (Abb. 1) und die Schleife verlassen, falls dieses gesetzt ist. Das Break-Flag ermöglicht so das asynchrone Anhalten der Simulation auf die Genauigkeit einer Ausführungseinheit. Da Ausführungseinheiten keine Sprünge und somit keine Schleifen enthalten, ist das Anhalten der Simulation durch Setzen des Flags garantiert. Auch die Beschränkung der Haltegenauigkeit auf Ausführungseinheiten ist kein Nachteil, da das Anhalten ohnehin asynchron erfolgt und somit keine Annahmen über die letzte ausgeführte Zielinstruktion gemacht werden dürfen. Trotz großer Ausführungseinheiten unterstützt Jahris auch synchrones, befehlsgenaues Anhalten der Simulation mittels *Breakpoints* und *Instruction Stepping*. Beide Methoden sind nichtinvasiv und erfordern weder eine explizite Unterstützung durch die Zielarchitektur, noch die Modifikation des Maschinencodes der Zielapplikation.

Breakpoints werden durch den Simulator verwaltet und können an jeder Position der Zielapplikation, auch innerhalb von Ausführungseinheiten, auftreten. Beim Setzen eines Breakpoints werden alle Ausführungseinheiten die ihn enthalten aus dem Simulationscache entfernt, sodass sie vor der nächsten Ausführung erneut dekodiert werden. Trifft der Befehlsdekoder auf einen Breakpoint, wird die Dekodierung der entsprechenden Ausführungseinheit abgebrochen, sodass diese unvollständig, nur bis zum Breakpoint ausgeführt wird. Um dabei Fragmentierung zu verhindern, werden unvollständige Ausführungseinheiten nicht im Simulationscache zwischengespeichert. Da alle Ausführungseinheiten bis zum Auftreten eines Breakpoints vollständig ausgeführt werden, wird die Simulationsperformance durch Breakpoints nicht vermindert. Ursprünglich verworfene

CPU	2 × Dual-Core AMD Opteron(TM) @2.4GHz, 2×128 kb L1, 2×1 MB L2
Hauptspeicher	4GB
OS	Ubuntu 9.04, Kernel 2.6.28-16-generic
JRE	Java(TM) SE Runtime Environment (build 1.6.0_16-b01)
QEMU Version	0.10.0
QEMU Zielplattform	ARM Versatile/PB (ARM926EJ-S)
QEMU Gastsystem	Linux 2.6.18 for Versatile (Debian 2.6.18.dfsg.1-23+versatile)

Tabelle 1: Konfiguration der Hostplattform



Sieve Iterationen	QEMU MIPS	Jahris MIPS	Jahris %[1)]
1 000	174.24	114.70	65.83
2 000	181.59	119.57	65.85
5 000	200.45	126.01	62.86
1 0000	206.18	129.26	62.69
		161.38[2)]	78.27[2)]

1) verglichen mit QEMU

2) mit JVM-Schaltern (siehe Text)

Abbildung 3: Simulationsperformance von Jahris und QEMU

Ausführungseinheiten werden in einer Liste zwischengespeichert und im Simulationscache restauriert, sobald ihr letzter Breakpoint gelöscht wurde. Instruction Stepping ermöglicht die Simulation einer genauen Anzahl von Zielinstruktionen oder bis hin zum nächsten Breakpoint. Dies wird durch einen Zähler für die Anzahl der noch auszuführenden Zielinstruktionen realisiert, welcher für jede Ausführungseinheit um die Anzahl der in ihr enthaltenen Instruktionen dekrementiert wird. Wiederum werden alle Ausführungseinheiten vollständig ausgeführt, bis auf die letzte für welche der Zähler den Wert Null unterschreiten würde. Lediglich die Verwaltung des Zählers führt zu einem minimalen Overhead gegenüber dem regulären Simulationsmodus.

4 Benchmark und Vergleich mit QEMU

Als Zielapplikation für den Benchmarktest wurde ein Sieve-Test in Form von ARMv4-Maschinencode im ELF-Dateiformat verwendet. Innerhalb einer Iteration des Benchmarks wird im Speicher der Zielarchitektur ein Primzahlsieb für die Werte 3 bis 16381 berechnet. Die Applikation wurde mit Jahris und QEMU auf dem in Tab. 1 aufgeführten Hostsystem ausgeführt. Aufgrund der zu erwartenden zunehmenden Amortisation der Kosten der dynamischen Compilierung für längere Simulationsläufe wurde die Simulationsperformance für verschiedene Lauflängen im Bereich 1 000 bis 10 000 Iterationen ermittelt. Die dafür herangezogene Metrik ist die durchschnittliche Anzahl ausgeführter Zielinstruktionen pro Zeiteinheit während eines vollständigen Benchmarklaufs in Millionen Instruktionen pro Sekunde (MIPS).

Die Benchmarkergebnisse sind in Abb. 3 zusammengefasst. Wie erwartet steigt die Simulationsperformance sowohl für Jahris als auch QEMU jeweils mit der Benchmarklauflänge. Unabhängig von dieser liegt die Performance von Jahris konstant bei etwa 63 bis 66 Prozent der von QEMU. Sie lässt sich jedoch für längere Simulationsläufe noch erheblich vergrößern, indem beim Start der JVM die Optionen `-Xverify:none` (deaktiviert den Bytecode-Verifier) und `-XX:+AgressiveOpts` (aktiviert aggressivere Optimierungsstrategien) ausgewählt werden. Wie für den Benchmarklauf mit 10 000 Iterationen gezeigt, erreicht Jahris unter Verwendung dieser Optionen sogar bis zu 78 Prozent der Performance von QEMU.

QEMU weist zwar eine etwas höhere Performance auf als Jahris, sein Anwendungsbereich liegt jedoch auch nicht in der Simulation sondern in der Full-System-Emulation. Die feingranulare Beobachtbarkeit konsistenter Zustände der Zielarchitektur ist dabei nicht von Belang. Zudem erfordert die Anpassung von QEMU an eine Zielarchitektur die manuelle Beschreibung derselben in C ausschließlich unter Benutzung des bereitgestellten Frameworks. Jahris hingegen verwendet eine dedizierte ISA-Beschreibungssprache zur raschen Modellierung von Zielarchitekturen. Ein weiterer großer Vorteil von Jahris besteht in der Verwendung der Java-Laufzeitumgebung als virtueller Hostplattform für die Simulation, was einerseits zu unerreichter Plattformunabhängigkeit führt, und andererseits Jahris direkt von den fortgeschrittenen Optimierungsstrategien moderner JVM Implementierungen profitieren lässt. In Anbetracht dieser Vorteile ist sind die Performanceeinbußen gegenüber QEMU vertretbar.

5 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein retargierbarer, hochperformanter ISA-Simulator mit Hot-Spot-Compilierung vorgestellt, welcher unterschiedliche Anforderungen an ISA-Simulatoren erfüllt und ihre wechselseitigen Konflikte weitgehend aufhebt. Er wurde in Java implementiert und verwendet die Java-Laufzeitumgebung als virtuelle Hostplattform für die Simulation, was ihn einerseits trotz dynamischer Compilierungtechniken außerordentlich portabel macht und andererseits die Ausnutzung der plattformspezifischen, hoch entwickelten Optimierungstechniken moderner JVM-Implementierungen erlaubt. Es wurde gezeigt, dass die Java-Laufzeitumgebung eine geeignete Plattform für die ISA-Simulation darstellt. Die dabei erzielte Simulationsperformance ist mit der von QEMU vergleichbar. Die Retargierung von Jahris erfolgt mittels HPADL, einer neuen ISA-Beschreibungssprache, welche die rasche Modellierung von Befehlssatzarchitekturen, ein hohes Maß and Flexibilität und die Unterstützung fortgeschrittener Simulationstechniken in sich vereint. Letzteres wird durch eine klare konzeptionelle Trennung von Dekoderverhalten und Befehlsverhalten in Architekturbeschreibungen erreicht. Die Leistungsfähigkeit von Jahris und HPADL wurde durch die Modellierung und den erfolgreichen Test einer Reihe von Zielarchitekturen demonstriert, darunter DLX, i8086, ARMv4 und 32-Bit-PowerPC. Die weiterführende Arbeit wird sich mit der verbesserten Unterstützung von laufzeitdynamischem Code sowie einer weiteren Steigerung der Simulationsperformance beschäftigen.

Literatur

[1] Marco Kaufmann. Erschließung von Just-in-Time-Compilierungstechniken in der Realisierung eines retargierbaren Architektursimulators, Dezember 2009.

[2] Achim Nohl, Gunnar Braun, Oliver Schliebusch, Rainer Leupers, Heinrich Meyr, and Andreas Hoffmann. A universal technique for fast and flexible instruction-set architecture simulation. In *DAC '02: Proceedings of the 39th annual Design Automation Conference*, pages 22–27, New York, NY, USA, 2002. ACM.

[3] Vojin Zivojnovic, Stefan Pees, and Heinrich Meyr. Lisa machine description language and generic machine model for hw/sw co-design. In W. Burleson, K. Konstantinides, and T. Meng, editors, *VLSI Signal Processing, IX, 1996.*, October 1996.

[4] Mehrdad Reshadi, Prabhat Mishra, and Nikil Dutt. Instruction set compiled simulation: a technique for fast and flexible instruction set simulation. In *DAC '03: Proceedings of the 40th annual Design Automation Conference*, pages 758–763, New York, NY, USA, 2003. ACM.

[5] Daniel Jones and Nigel Topham. High speed cpu simulation using ltu dynamic binary translation. In *HiPEAC '09: Proceedings of the 4th International Conference on High Performance Embedded Architectures and Compilers*, pages 50–64, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 41–41, Berkeley, CA, USA, 2005. USENIX Association.

Autorenverzeichnis

Seite

B
René Bergelt, TU Chemnitz *67*
Enno Böhme, ZMD AG Dresden *25*
Volker Boos, Institut f. Mikroelektronik und Mechatronik Systeme GmbH Ilmenau *61*
Rainer Brück, Universität Siegen *49*

D
Jörg Doblaski, X-FAB Semiconducter Foundries AG Erfurt *7*

F
Dietmar Fey, Universität Erlangen-Nürnberg *123*
Robert Fischbach, TU Dresden *37*
Wolfgang Flohrer, X-FAB Semiconducter Foundries AG Erfurt *7*
André Froß, TU Chemnitz *55, 111*
Daniel Froß, TU Chemnitz *55, 111*

G
Andreas Gajda, TU Dresden *85, 91*
Matthias Glockner, TU Chemnitz *67*
Achim Graupner, ZMD AG Dresden *25*
Hagen Grätz, Fraunhofer IPMS Dresden *103*
Wolfgang Grimm, X-FAB Semiconducter Foundries AG Erfurt *7*
Stefan Günther, Anvo-Systems Dresden GmbH *103*

H
Joachim Haase, Fraunhofer IIS, Institutsteil EAS Dresden *13*
Kai Hahn, Universität Siegen *49*
Wolfram Hardt, TU Chemnitz *67*
Johann Hauer, Fraunhofer IIS Erlangen *117*
Ulrich Heinkel, TU Chemnitz *55, 111*
Stephan Henker, TU Dresden *97*
Jörg Hertwig, TU Dresden *43*
Hans-Jürgen Holland, Fraunhofer IPMS Dresden *103*
Fabian Hopsch, Fraunhofer IIS, Institutsteil EAS Dresden *19*

K
Marco Kaufmann, TU Dresden *129*
Tobias Koal, BTU Cottbus *73*
Steffen Köhler, TU Dresden *85, 91*

L
André Lange, Fraunhofer IIS, Institutsteil EAS Dresden *13*
Jan Langer, TU Chemnitz *111*
André Lerch, X-FAB Semiconducter Foundries AG Erfurt *7*
Jens Lienig, TU Dresden *37, 43*

M
Christian Mayr, TU Dresden *97*
Hendrik T. Mau, Globalfoundries Dresden *13*
Tilo Meister, TU Dresden *37*
Matthias Mielke, Universität Siegen *49*
Dietmar Mörtl, ZMD AG Dresden *25*
Lutz Muche, Fraunhofer IIS, Institutsteil EAS Dresden *13*

Seite

N
Holger Neubert, TU Dresden *43*

O
Dirk Ortloff, Process Relations GmbH Dortmund *49*

P
Johannes Partzsch, TU Dresden *97*
Stefan Potyra, Universität Erlangen-Nürnberg *123*
Thomas B. Preußer, TU Dresden *129*
Michael Pronath, MunEDA GmbH München *25*

R
Christian Roßberg, TU Chemnitz *55*

S
Matthias Sand, Universität Erlangen-Nürnberg *123*
Andreas Scade, Anvo-Systems Dresden GmbH *103*
Thilo Schmidt, Universität Siegen *49*
Stefan Scholze, TU Dresden *97*
Mario Schölzel, BTU Cottbus *79*
René Schüffny, TU Dresden *97*
Volkmar Sieh, Universität Erlangen-Nürnberg *123*
Udo Sobe, ZMD AG Dresden *25*
Rainer G. Spallek, TU Dresden *91, 129*
Bernd Straube, Fraunhofer IIS, Institutsteil EAS Dresden *19*

V
Wolfgang Vermeiren, Fraunhofer IIS, Institutsteil EAS Dresden *19*
Heinrich T. Vierhaus, BTU Cottbus *73*
Matthias Vodel, TU Chemnitz *67*
Matthias Völker, Fraunhofer IIS Erlangen *117*

W
Ulf Wetzker, Fraunhofer IIS, Institutsteil EAS Dresden *31*

Z
Haiyan Zhou, Fraunhofer IIS Erlangen *117*

www.eas.iis.fraunhofer.de

ISBN 978-3-8396-0126-6