Dipl.-Ing. Andreas Genser

Estimation-Based Power Management

An Early Design Phase Investigation Method Based on Power Emulation

Dissertation

vorgelegt an der

Technischen Universität Graz

zur Erlangung des akademischen Grades
Doktor der Technischen Wissenschaften
(Dr. techn.)

durchgeführt am Institut für Technische Informatik
Technische Universität Graz
Vorstand: O. Univ.-Prof. Dipl.-Ing. Dr. techn. Reinhold Weiß

Graz, im Mai 2011

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....
(Unterschrift)

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Kurzfassung

Die immense Anzahl an mobilen Systemen, die in der Gesellschaft immer mehr allgegenwärtig werden, tragen einen stetig steigenden Anteil am globalen Energieverbrauch. Außerdem erlaubt der Fortschritt der CMOS Technologie immer komplexere Systeme, die ihrerseits eine immer breiter werdende Palette von Anwendungen ermöglichen. Die steigende Komplexität und der gleichzeitig vergleichsweise langsame Fortschritt in der Verbesserung von Batterietechnologien limitieren die Laufzeit mobiler Systeme. Diese Entwicklungen haben dazu geführt, dass sowohl in der Forschung als auch in der Industrie ein erhöhtes Augenmerk auf die Steigerung von Leistungs- und Energieeffizienz mobiler Systeme gelegt wird. Low-Power Design Methoden haben in den letzten Jahrzehnten entscheidend dazu beigetragen diese Effizienzsteigerungen zu erreichen. Die inzwischen erreichte Komplexität erfordert jedoch zusätzliche Werkzeuge und Methoden zur Entwicklung von immer leistungs- und energielimitierteren Systemen.

Diese Arbeit umfasst die Entwicklung einer Power-Emulations Plattform, welche Informationen zur Leistungsanalyse mobiler Systeme zur Verfügung stellt. Im Vergleich zu konventionellen Leistungsanalyse-Methoden, ermöglicht Power Emulation das Durchführen der Leistungsanalyse wesentlich früher im Designprozess, um folglich schon sehr früh Optimierungspotential erkennen und darüberhinaus Methoden zur Reduktion der Leistungs- und Energieaufnahme einsetzen zu können. Zur Optimierung werden üblicherweise Power Management Methoden verwendet. In dieser Arbeit wird Power Management zur Glättung von Leistungsspitzen bzw. zur Vermeidung von Versorgungsspannungseinbrüchen für Smart Card Systeme, welche über ein magnetisches Feld versorgt werden, untersucht. In dieser Klasse von Systemen sind nicht nur Leistungs- und Energieoptimierungen ein wichtiger Bestandteil, sondern auch das Gewährleisten der Systemstabilität, welche durch auftretende Leistungsspitzen bzw. Versorgungsspannungseinbrüchen gefährdet ist. Der Ansatz der Power Emulation hilft geeignete Power Management Methoden früh im Designprozess zu untersuchen und daraus gewonnene Erkenntnisse direkt in diesen zurückfließen zu lassen. Die effiziente Leistungsanalyse mobiler Systeme und die Möglichkeit Maßnahmen zur Leistungs- und Energieoptimierung als auch zur Steigerung der Systemstabilität früh im Designprozess untersuchen zu können, stellt den Hauptbeitrag dieser Arbeit dar.

Abstract

The huge amount of electronic equipment penetrating our everyday life contributes to a continuously growing fraction of the worldwide electricity consumption. In addition, the steadily increasing complexity of these systems provides a wide range of applications, which have constrained operating times due to the limitations of battery life that has not kept pace with system complexity. These trends have fueled efforts made in academia as well as in the industry to pay higher attention to power and energy efficiency enhancements. Low power design techniques employed in the last decades have essentially contributed to efficiency improvements in the mobile systems domain. However, the ever increasing system complexity, that designers are facing today, requires powerful power-aware design tools and design methodologies to cope with increasingly critical power and energy issues.

This work is part of the POWERHOUSE research project, which aims at establishing a power emulation platform that provides power consumption information in real-time at low hardware overhead in order to speed up power analysis over conventional power simulation methods. Based on the proposed high-level power emulation approach, power analysis can be employed much earlier in the design phase allowing for the early identification of optimization potential as well as the early investigation of the effectiveness of power management techniques. Power management techniques investigated in this work are power profile flattening and supply voltage drop compensation techniques targeted at RF-powered smart card systems, which are representative of severely power-constrained embedded systems designs. These systems do not only require power-aware optimizations that focus on power and energy efficiency improvements, but also on ensuring system reliability that is threatened by high load changes causing the supply voltage of the system to drop below a critical limit. Enabling the investigation of power management techniques by means of the high-level power emulation approach helps to determine the effectiveness of these techniques in an early design phase. This constitutes a significant contribution to existing power-aware design methods by incorporating drawn conclusions from the investigation phase early in the development cycle, which enhances the power and energy efficiency of these systems and ensures the maintenance of the reliability of severely power-constrained systems.

Acknowledgements

The POWERHOUSE project, funded by the Austrian Federal Ministry for Transport, Innovation and Technology, has been established in cooperation of the Institute for Technical Informatics, Infineon Technologies AG and Austria Card GmbH. The long way to finish this PhD thesis involves many people that contributed to the success of this work, too many to name them all. To a few of them I would like to express my gratitude.

Foremost, I would like to thank Prof. Reinhold Weiss for offering the opportunity to participate in the hardware/software codesign research group at the Institute for Technical Informatics. Furthermore, I appreciate his advice and thoughtful comments to compose and finish this thesis.

I also would like to thank Christian Steger, Josef Haid and Holger Bock for their efforts to establish the POWERHOUSE project, to set up a productive collaboration between industry and academia offering myself a competitive research and industry working environment. Their successful project management kept the group focused on research topics and the dissemination of achieved results. The funding provided by the Austrian Federal Ministry of Transport, Innovation and Technology opened the chance to attend a number of conferences to present research results on an international scale and to discuss own ideas and achievements with other international researchers.

Thanks also to Francois Reney, Johannes Grinschgl and Armin Krieg for their helpful comments given on this thesis.

In particular I would like to express my gratitude to my colleague Christian Bachmann for the great collaboration throughout almost three years of research, for creative discussions, his well-founded comments and for his support in publishing research achievements. He contributed significantly for making the POWERHOUSE project a success.

Finally, the greatest gratitude goes to Sarah and my family for enduring an extensive, stressful and exhausting time and for their continuous and faithful encouragement, which made me successfully complete this work. Without you, none of this would have been possible. Thank you.

Graz/Austria
May, 2011 Andreas Genser

Extended Abstract

Mobile systems have been penetrating our everyday life in a wide range of different areas. The range of these systems incorporating mobile phones, MP3 players, digital cameras or mobile positioning systems is constantly growing at a tremendous pace. This huge amount accounts to an increasing fraction of the global electricity consumption. Moreover, the growing diversity in system functionality renders long operating times of mobile systems a challenging task. Both trends have fueled efforts in academia as well as in the industry to achieve higher power and energy efficiency. The purely performance-driven system design paradigm followed until the early nineties has shifted towards more power-aware design strategies. Low-power design techniques have significantly contributed to higher power and energy efficiency in today's mobile systems. However, the increasing system complexity additionally requires effective power-aware design tools and design methodologies to ensure efficient system design in future. In addition, in recent years an emerging hardware design productivity gap has appeared expressing the discrepancy between technology capabilities provided by CMOS advances and the missing technology exploitation that is increasingly limited by high power densities on chip, which affects system reliability. An enormous complexity level can be achieved on a single chip today, however exploiting this potential requires to raise the level of abstraction of power-aware design tools and to utilize them earlier in the design phase.

Power-aware system design requires two inherent ingredients, which are (i) *effective power analysis methods* and (ii) *effective power management techniques*. Power analysis is exploited to profile a given system to identify power optimization potential. Usually this step is conducted late in the design process thus limiting the exploitation of optimization potential. Power simulation, an existing early design phase power analysis method, suffers from extensive simulation times, which renders power analysis of complex applications unfeasible. Alternatively, FPGA prototyping platforms allow for hardware-accelerated power analysis achieving high power analysis performance still in an early design phase. However, these methods still lack real-time capability and they suffer from high hardware overhead. An early design phase power analysis method utilizing hardware acceleration to achieve high power analysis performance on the one hand and low hardware overhead on the other hand provides a promising approach to enable the early design phase investigation of power management techniques.

Power management techniques have been introduced alongside low-power design techniques in order to enhance the power and energy efficiency of mobile systems. An enormous number has been proposed in the past ranging from methods that switch off unused system modules or that adapt system parameters, such as system frequency and supply voltage. The effectiveness of power management techniques, however, is not known a priori, hence an early design phase investigation of these techniques and their evaluation

would add additional value to a power-aware design process by determining the most apt power management techniques for a dedicated system and corresponding applications.

This work is part of the POWERHOUSE project, which aims at the realization of a power emulation platform that delivers power consumption information of power-constrained systems at a significant higher performance as compared to power simulations. Power consumption information gained is exploited for the investigation of power management techniques in an early design phase particularly focusing on RF-powered smart card systems. In this work, a power estimation architecture providing power consumption information for power analysis purposes is presented, aiming at real-time deliverance of power consumption information as well as low hardware overhead. This high-level power emulation approach is implemented on an FPGA prototyping platform to allow for early design phase power analysis[1]. In order to acquire accurate power estimates, a power model is created and integrated in the power estimation architecture. Moreover, the elaboration of the power model, the corresponding power characterization procedure and the automation of this process is presented[2]. The quality of the high-level power emulation approach in terms of power estimation accuracy, power estimation performance as well as hardware overhead is evaluated and a concept for the potential exploitation of this technique for power management is depicted[3].

Power management techniques investigated in this work by means of the high-level power emulation approach are targeted at RF-powered smart cards systems. These systems are highly susceptible to high load changes caused by large power consuming system modules. Sudden power consumption changes can lead to supply voltage drops that jeopardize system stability. Consequently, power management techniques investigated in this work focus on power profile flattening and supply voltage drop compensation techniques that adapt the system's power consumption by dynamic voltage and frequency scaling (DVFS). The early design phase investigation of these techniques on an FPGA prototyping platform by means of the high-level power emulation approach requires hardware extensions to model DVFS behavior[4]. Power profile flattening techniques are presented in a twofold manner, first as an offline power optimization variant[5] and as a run-time power management technique that autonomously flattens the system's power profile to avoid supply voltage drops[6]. Finally, a more sophisticated power management technique to avoid harmful supply voltage drops is presented. By introducing a supply voltage estimation architecture, a power management technique is introduced that does not only

[1]An Emulation-Based Real-Time Power Profiling Unit for Embedded Software, IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, Samos, Greece, July 20–23, 2009

[2]Automated Power Characterization for Run-Time Power Emulation of SoC Designs, Euromicro Conference on Digital Systems Design, Architectures, Methods, and Tools, Lille, France, September 1–3, 2010

[3]A Hardware-Accelerated Estimation-Based Power Profiling Unit - Enabling Early Power-Aware Embedded Software Design and On-Chip Power Management, Springer Transactions on High-Performance Embedded Architectures and Compilers (Transactions on HiPEAC), 2011

[4]Power Emulation Based DVFS Efficiency Investigations for Embedded Systems, IEEE International Symposium on System-on-Chip, Tampere, Finland, September 28–30, 2010

[5]An Automated Framework for Power-Critical Code Region Detection and Power Peak Optimization of Embedded Software, International Workshop on Power and Timing Modeling, Optimization and Simulation, Grenoble, France, September 7–10, 2010

[6]Estimation-Based Run-Time Power Profile Flattening for RF-Powered Smart Card Systems, IEEE Asia Pacific Conference on Circuits and Systems, Kuala Lumpur, Malaysia, December 6–9, 2010

aim at the flattening of the power profile, but on directly monitoring the system's supply voltage and steering DVFS adaptions to avoid voltage drops[7].

Power management techniques help to improve power-aware system design and contribute to counter the widening hardware design productivity gap. High power and energy efficiency and, as a consequence, better technology exploitation can be achieved by providing effective power analysis methods as well as power management techniques. Enabling the investigation of power management techniques by means of the high-level power emulation approach helps to determine the effectiveness of these techniques much earlier in the design phase, which allows the incorporation of conclusions drawn from the investigation phase early in the development cycle. This yields a significant contribution to existing power aware design methods by enhancing power and energy efficiency as well as by ensuring the applicability of severely power-constrained systems.

[7]Supply Voltage Emulation Platform for DVFS Voltage Drop Compensation Explorations, IEEE International Symposium on Performance Analysis of Systems and Software, Austin, Texas, April 10–12, 2011

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Importance of Power-Awareness

Worldwide electricity consumption is rapidly growing, which raises severe environmental challenges. Data center equipment consumed 61 billion kWh in 2006 accounting for 1.5% of the total US electricity consumption, which was twice the amount consumed in 2000. Data center electricity consumption keeps growing exponentially, presumably again doubling the electricity demand between 2006 and 2011 [14].

The electricity consumption by information and communication technologies (ICT) and consumer electronics (CE) has been the fastest growing in OECD and non-OECD countries between 2004 and 2009 contributing to 15% to the global electricity consumption. The increasing demand for these products and the increasing diversity in functionality lead to the growing electricity consumption despite of achieved system efficiency improvements. The *International Energy Agency (IEA)* predicts that the energy consumption of ICT and CE equipment will double by 2022. It is estimated that ICT and CE energy consumption in 2030 will rise up to three times that of the 2009 level (see Figure 1.1) [1].

The *International Technology Roadmap for Semiconductors (ITRS)* [2] projects a similar picture for portable consumer devices between 2010 and 2024. Figure 1.2 illustrates that the power demand of such systems is expected to increase by an order of magnitude by 2024, which by far exceeds low power requirements constrained by the slow pace of battery efficiency improvements.

Alongside developing new sources of clean energy to account for the growing electricity demand, efficiency improvements by means of power-aware design techniques are required. Power has become the major design constraint and power management techniques have been introduced to automatically adapt a system in a way that power is only consumed during phases when user services are required. Raising the power-awareness of system designers as well as implementing power management techniques to influence the power consumption yield higher power and energy efficiencies. Due to the huge number of systems deployed worldwide, this helps to counter the trend of growing global electricity consumption.

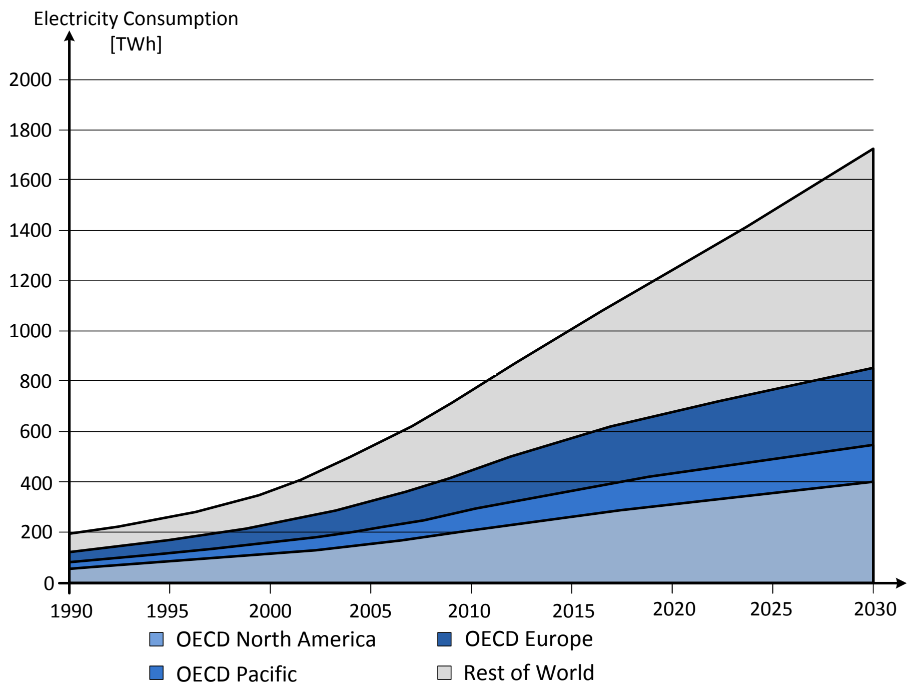


Figure 1.1: Electricty consumption in the residential sector for ICT and CE equipment estimated by the IEA, obtained with modifications from [1]

1.2 The Hardware Design Productivity Gap

With each new CMOS technology generation designers are facing an increasing number of design challenges. Advances in CMOS technology capabilities, according to Moore's law on the one hand and the lack of efficient design tools on the other hand, cause an emerging hardware design productivity gap. The ITRS update in 2010 outlines this growing gap as illustrated in Figure 1.3. Today a large collection of design tools and effective methodologies are required to establish products resulting in the reduction of the hardware design productivity gap [3]. However, the existing lack of high quality design tools involved in the design, implementation and functional verification phase of a system constrains productivity. Moreover, the growing transistor count within the same area increases power densities that worsens system reliability. Advances in power-aware design tools to identify harmful events and countermeasures, such as power management techniques, reduce the risk of system malfunctions by avoiding power-critical events, which helps to reduce the hardware design productivity gap.

Power analysis is an inherent ingredient that allows for the design of power-aware systems. The following key strategies are central to enable power-aware systems that exploit power management techniques: (i) *monitoring of the system's power consumption by means of power analysis methods*, (ii) *control policies leveraging the power consumption in order to adapt system parameters that influence the system's power consumption* and (iii) *early design phase applicability.*

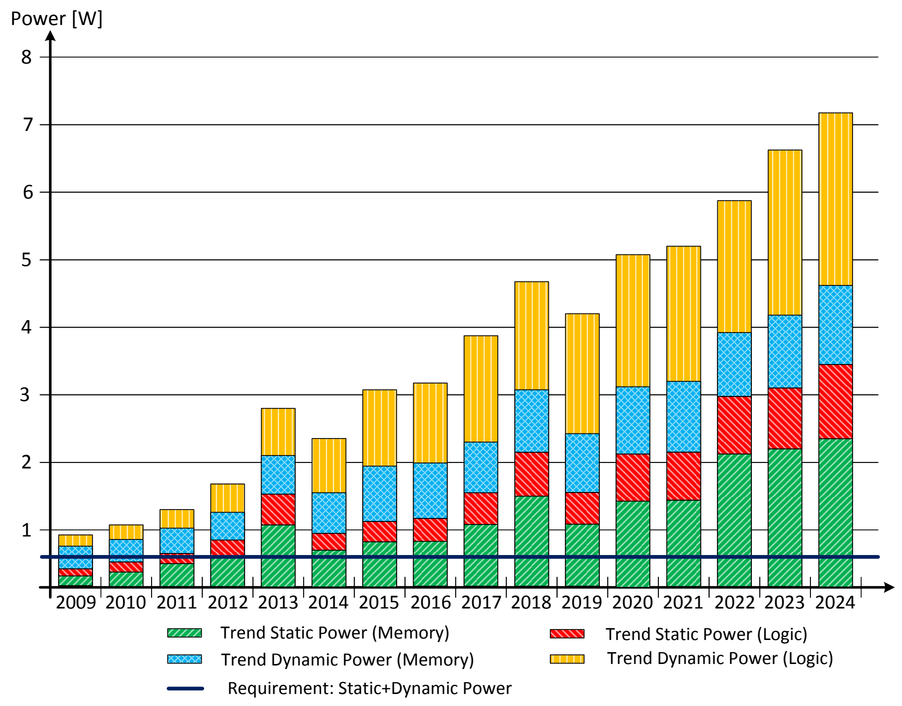


Figure 1.2: ITRS 2010 update, power consumption trends projected for consumer portable devices, obtained with modifications from [2]

1.3 Power Analysis Techniques and Their Deficiencies

The early design phase investigation of power-aware techniques, such as power management, relies on power analysis methods, which obtain power consumption information from a system. A number of power analysis methods are available and can be categorized as (i) *measurement-based* and (ii) *estimation-based approaches*.

Measurement-based power analysis combines the advantages of high accuracy and real-time capability [15, 16]. However, the major deficiency of these methods is their late design phase applicability, which is typically not before the first silicon of the system is available. In order to explore the effectiveness of power management techniques, early design phase applicability is of great importance to avoid expensive redesigns due to silicon respins.

As a consequence, *estimation-based methods* have been introduced and can be divided into (i) *simulation-based* [17, 18, 19, 20] and *hardware-accelerated* [21, 22, 23] methods.

Simulation-based power analysis suffers from extensive simulation times, in particular if employed on low abstraction layers, which is typically the case in industrial state-of-the-art power analysis tools operating at gate level. Raising the layer of abstraction speeds up power analysis, however this introduces the risk of high estimation errors. *Hardware-accelerated power analysis* on the other hand, relieves the burden of high simulation times by introducing additional hardware to speed-up the power estimation process.

Unlike measurement-based methods, estimation-based power analysis can be employed in an early design phase and, if implemented in a hardware-accelerated manner, it can also achieve high estimation speeds. This forms the basis for effective power-aware system design tools that allows the rapid investigation of power management techniques.

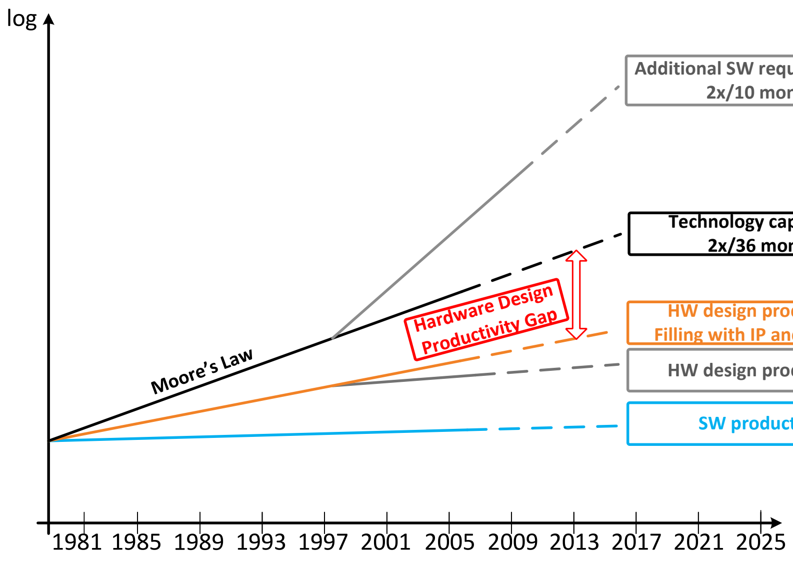


Figure 1.3: ITRS 2010 update, hardware design productivity gap, obtained with modifications from [3]

1.4 Early Design Phase Power Management Investigations

Power management techniques propose themselves as effective alternatives, which make systems more power efficient and in turn help to reduce power densities to exploit technology capabilities more efficiently. Power analysis provides the basis for early design phase power efficiency investigations of power management techniques. Hardware-accelerated power estimation as a power analysis technique delivers a promising approach enabling real-time and accurate power analysis. By implementing it on an FPGA prototyping platform, early design phase use becomes feasible. This method is referred to as *power emulation*.

1.4.1 Power Emulation

Hardware-accelerated power analysis overcomes the severe limitation of slow simulation speed by introducing additional hardware to evaluate power models that provide power estimates. *Power emulation*, first introduced by Coburn et al. in [24] and subsequent approaches in [25, 26], resemble a special form of hardware-accelerated power analysis. FPGA boards are exploited to emulate both functional system behavior and the system's power consumption by mapping an embedded system design and additional power estimation hardware on the FPGA platform.

Figure 1.4 overviews the benefits of an emulation-based development process over traditional ones. Unlike the traditional development process that is based on power simulations, power emulation offers a significant speed-up over power simulations. This allows efficient early design phase power analysis, reducing the risk of costly redesigns and thus decreasing time-to-market.

First *power emulation* attempts implemented at register transfer level achieved speed-

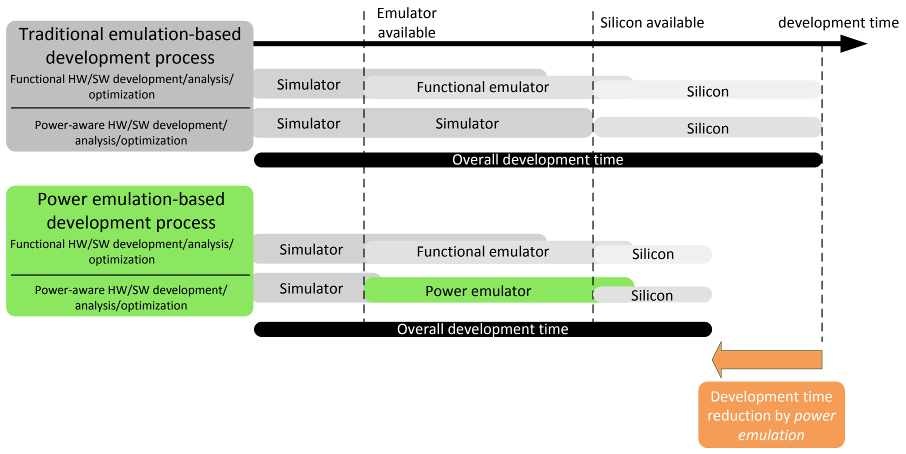

Emulator available
Silicon available
development time
Traditional emulation-based development process
Functional HW/SW development/analysis/optimization
Power-aware HW/SW development/analysis/optimization
Simulator
Functional emulator
Silicon
Simulator
Simulator
Silicon
Overall development time
Power emulation-based development process
Functional HW/SW development/analysis/optimization
Power-aware HW/SW development/analysis/optimization
Simulator
Functional emulator
Silicon
Simulator
Power emulator
Silicon
Overall development time
Development time reduction by power emulation

Figure 1.4: Power emulation principle yields significant improvements in development times and thus a faster time-to-market, obtained from [4]

ups of 10x to 500x compared to commercial power estimation tools. However, the significant hardware overhead introduced by this category of power analysis techniques and the lack of real-time capability remain major challenges to be solved. Raising power emulation on higher abstraction layers poses a promising alternative to cope with these challenges.

In this work, a *high-level power emulation approach* is introduced, tackling existing power analysis challenges. Details can be obtained from Section 6.1. The automated generation of the power estimation hardware and the elaboration of the power model is shown in Section 6.2. Finally, a thorough discussion of the high-level power emulation approach, which provides rich experimental data and potential applications is given in Section 6.3.

1.4.2 Power Management Techniques based on Frequency and Voltage Adjustments

An extensive variety of power management methods has been proposed in the recent decade ranging from dynamic power management (DPM) techniques [27] to dynamic voltage and frequency scaling (DVFS) approaches [28]. The focus of DPM is to minimize the number of active system components by switching them off in idle phases. On the other hand, DVFS has emerged as an appealing power management alternative by continuously adapting the system's frequency and the supply voltage. Depending on performance demands or power availability conditions, the power consumption of the system is influenced by DVFS adaptions.

Unlike the domain of battery-powered mobile systems where power management techniques focus on minimizing the system's power and energy consumption in order to maximize the system's operating time, this work focuses on autonomous systems, such as RF-powered smart cards, that are highly susceptible to power consumption variations (i.e., power peaks). Large power consuming system modules (e.g., cryptographic coprocessors) can cause power peaks leading to supply voltage drops below a critical limit, which could threaten system stability. Power management techniques mitigate harmful supply voltage drops and enhance system reliability.

Power profile flattening methods based on DVFS offer such power management techniques as an alternative to increase the reliability of RF-powered smart card systems. Moreover, in this work, a supply voltage emulation approach for RF-powered smart card systems is proposed that enables the exploitation of supply voltage drop compensation algorithms. These are superior over pure power profile flattening methods, since they focus directly on the avoidance of harmful supply voltage drops.

Section 6.4 gives a brief overview on hardware extensions required to emulate DVFS power management techniques on an FPGA prototyping platform. Publications in Section 6.5 and Section 6.6 incorporate power profile flattening techniques based on the proposed power emulation approach. Finally, the publication given in Section 6.7 introduces the concept of supply voltage emulation and corresponding voltage drop compensation schemes in the context of early design phase investigation of power management techniques.

1.4.3 The POWERHOUSE Project

This thesis is part of the POWERHOUSE project (*POWER-aware, Hardware-supported Operating System and Ubiquitous application Software development Environment*)[1]. The major goal of the project is the design and implementation of a real-time power emulation platform, which delivers power consumption information of power-constrained embedded systems at a significant higher performance compared to power simulations. Consequently, the accurate power consumption information gained from the power emulation approach can be exploited for power-aware hardware/software codesign techniques as well as for early design phase power management investigations. This principle is illustrated in Figure 1.5.

1.4.4 Problem Description

The paucity of design tools incorporating effective power analysis methods in order to allow for the investigation of early design phase power management techniques, contributes significantly to the increasing hardware design productivity gap. Major deficiencies can be summarized as follows:

- Late design phase applicability of rapid power analysis methods
- Extensive simulation times of existing early design phase power analysis methods
- High hardware overhead of existing high-performance power analysis methods employed in an early design phase
- Poor design process tool integration to analyze the effectiveness of power management techniques

[1]Project partners are Infineon Technologies Austria AG, Austria Card GmbH and TU Graz. The project is funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 815193.

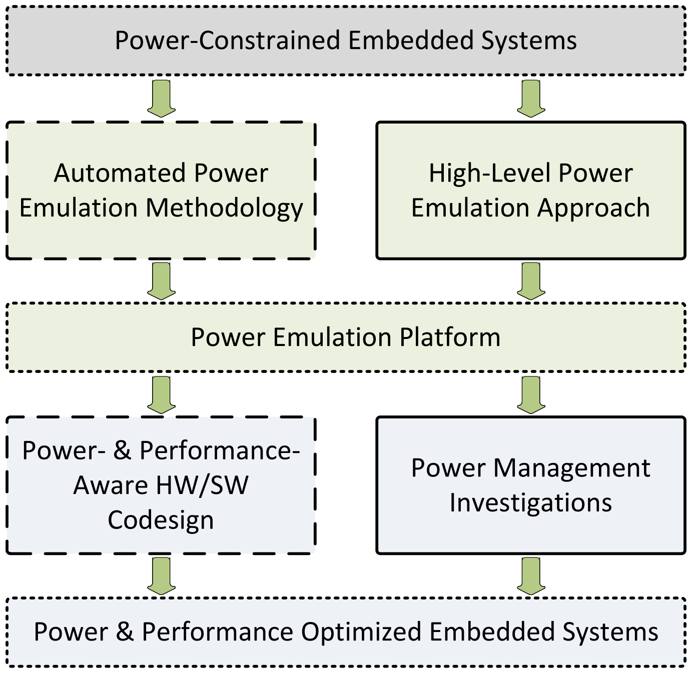

Power-Constrained Embedded Systems
Automated Power Emulation Methodology
High-Level Power Emulation Approach
Power Emulation Platform
Power- & Performance-Aware HW/SW Codesign
Power Management Investigations
Power & Performance Optimized Embedded Systems

Figure 1.5: Overview of the POWERHOUSE methodology, obtained with modifications from [5]

1.4.5 Contributions and Significance

The major contributions of this work can be given in a twofold manner:

1.) *High-level power emulation approach:* The late design phase applicability of measurement-based power analysis methods is not applicable for the early investigation of power management techniques. A high-level power emulation platform, as presented in this thesis, resolves this matter. Main attention is focused to achieve real-time capability as well as low hardware overhead in order to make this approach applicable to complex embedded system designs in a generic way not limited to a single system by utilizing standard FPGA platforms.

2.) *Early design phase power management investigations based on power emulation:* The proposed high-level power emulation approach permits the investigation of power management techniques prior to silicon chip manufacturing. This offers great flexibility and helps to determine the most apt power management technique for a given application. This work deals with power management techniques for RF-powered smart card systems aimed at power profile flattening techniques, which avoid harmful supply voltage drops. Moreover, this work is extended with supply voltage emulation functionality and supply voltage drop compensation techniques in oder to increase the reliability of future multi-core RF-powered smart card systems. The feasibility of both, the FPGA prototyping platform enabling the *high-level power emulation approach* and the *early design phase investigation of power management techniques* is illustrated in a number of case studies.

1.4.6 Thesis Structure

Chapter 2 overviews previous work on power analysis methods and power management techniques. In Chapter 3, the novelties of this work compared to the state-of-the-art are explained. Chapter 4 discusses case studies employed for the high-level power emulation approach and for the early design phase investigation of power management techniques with the purpose of proving their feasibility. Finally, in Chapter 5 conclusions are drawn and prospective research that could follow this work is outlined. Chapter 6 includes a selection of publications disseminated within the research project that deliver detailed information on the concepts developed in this work.

Chapter 2

Related Work

This chapter provides an overview on the advantages and disadvantages of existing power analysis methods that form the basis for the early design phase investigation of power management techniques. Moreover, state-of-the-art DVFS power management techniques are presented, which show present research objectives on either the investigation of simulation-based techniques or methods applicable late in the design phase. Finally, ongoing efforts in the area of power profile flattening as a power management technique and existing supply voltage drop compensation techniques and their limitations are discussed.

2.1 Power Analysis Techniques

Power analysis techniques are used to profile a given system to obtain its power consumption behavior or to determine optimization potential. They can be categorized as (i) *measurement-based* and (ii) *estimation-based* methods.

Measurement-based methods are performed by taking actual physical measurements, which yield high accuracy, but require expensive measurement equipment. Unlike measurement-based methods, power analysis by means of *estimation-based methods* delivers power estimates based on power models. These techniques are usually less accurate but provide greater flexibility, since power consumption for sub-modules of the system can be derived and early design phase applicability is feasible. The following ongoing research activities are compared in the field of power analysis techniques.

2.1.1 Measurement-Based Methods

In [15], PowerScope, an energy profiling tool for mobile applications, is introduced. The system's current consumption is automatically measured by a digital multimeter during run-time. Measurement data are collected for later analysis on a host computer. An oscilloscope measurement-based profiling technique is proposed by Texas Instruments in [16]. The current drawn by a DSP system is profiled and results are visualized on a host computer in TI's software development environment.

2.1.2 Estimation-Based Methods

Power analysis by means of estimation techniques can be subdivided into (i) *simulation-based* and (ii) *hardware-accelerated* approaches.

Simulation-based power estimation approaches execute programs on simulators to obtain circuit activity information, which feed power models that produce power estimates. In *hardware-accelerated power estimation approaches*, these power models are implemented in hardware.

Estimation techniques can be employed at various levels of abstraction yielding different estimation accuracies. Moreover, the degree of abstraction impacts on simulation times for *simulation-based approaches* and hardware effort for *hardware-accelerated methods*. Commercially available power estimation tools that operate at low abstraction levels, such as gate- or register transfer level (RTL), are mostly simulation-based [29]. Achievable estimation accuracies are high, while extensive simulation times render power estimation of elaborate applications unfeasible. Therefore, attempts are made to raise the level of abstraction to estimate a system's power consumption more efficiently.

A *simulation-based approach* employing power models for instruction level power estimations is proposed by Tiwari et al. in [17]. It allows for power and energy consumption estimation for given applications. The underlying power model considers the power consumption during instruction execution (i.e., base costs) and power consumption during the transition between instructions (i.e., circuit state overhead costs). In [18], Sami et al. consider additional microarchitectural effects to enhance the accuracy of instruction level power estimation based on a pipeline-aware power model for very long instruction word (VLIW) architectures. A co-simulation based power estimation technique is introduced by Lajolo et al. in [19]. This approach for embedded systems works at multiple abstraction levels. In principle, power estimation is performed at the system level, while for refinement purposes and accuracy enhancements various components are co-simulated at lower levels of abstraction. Countermeasures against high simulation times are caching, statistical sampling and macro-modeling. A simulation framework for system level SoC power estimation is introduced by Lee et al. in [20]. This approach is based on power models developed for the processor, memories and custom IP blocks. Power values derived are provided cycle-accurately to the designer in a dedicated profile-viewer. Ahuja et al. leverage a probabilistic RTL power estimation approach at system level. At the expense of a loss of accuracy of 3-9% they achieve simulation speed-ups of up to 12x compared to ordinary RTL power estimations [30].

Hardware-accelerated power estimation techniques are performed by augmenting the given system with existing or dedicated power estimation hardware. A power characterization process determines power values, which are mapped onto corresponding power states. For example, hardware events (e.g. CPU idle/run states, memory read/write states, etc.) are representatives of such power states. Available power estimation hardware can be extended to power state counters for energy consumption accounting. In [21], Bellosa gathers information by means of hardware event counters to derive thread-specific energy information for operating systems. Joseph et al. obtain the power consumption of a system by exploiting existing hardware performance counters of a microcontroller [22]. Microprocessor performance counters are utilized for system-wide power estimations by Bircher et al. in [31]. A power macro-model based coprocessor approach for energy accounting is proposed in [23]. Energy events identified by energy sensors are tracked by a central controller. In general, the additional power estimation hardware requires extra chip area but also yields a speed-up compared to simulation-based approaches.

2.1.3 Power Emulation

Power emulation represents a special case of hardware-accelerated power estimation. FPGA boards can be used as a typical prototyping platform to emulate not only the functional system behavior, but also its power consumption. A given system comprising power estimation hardware is mapped onto an FPGA platform. Functional verification and power estimation can be performed in real-time before the silicon implementation of the system is available.

Coburn et al. present the principle of power emulation in [24]. Run-time improvements of about 10x to 500x over commercial power estimation tools are achieved. Strategies to minimize the hardware overhead introduced by additional power estimation hardware are proposed. In [25], this approach is extended to a hybrid power estimation methodology for complex SoCs. This framework combines simulation and emulation techniques, which significantly reduce power analysis times. In [26], the power consumption of processor cores is estimated employing power emulation to guide process migration between cores.

The high evaluation performance of power models provided by the *power emulation* method and its early design phase applicability make it a promising power analysis method for the early investigation of power management techniques. However, power estimates delivered in real-time at low hardware overhead are still features not provided by state-of-the-art power emulation approaches.

2.2 Power Management Techniques

2.2.1 Overview on DVFS Power Management Techniques

First power management attempts concentrated on switching off inactive system modules. This method is often referred to as dynamic power management (DPM) [27]. In recent years, dynamic voltage and frequency scaling (DVFS) power management techniques have been proposed to influence the system's power consumption by continuously adapting the system frequency and the supply voltage [28]. This section gives an overview on DVFS-based power management techniques. It is limited to a subset of all available work on DVFS, which would be too numerous to discuss. The addressed subset discusses DVFS-based power management techniques that can be categorized in *simulation-based* and *in-system* approaches.

Pillai et al. propose a simulation-based DVFS power management technique by evaluating different DVFS algorithms for embedded systems in order to guarantee the real-time behavior of operating systems [32]. They compare energy efficiency gains by simulating the system within a simulator established in C++. However, for each executed cycle a constant amount of energy consumption is assumed. This is a rather simplified assumption, since power and energy consumption might vary greatly during algorithm execution. Moreover, they do not consider the voltage regulator impact on performance, power and energy consumption. In [33], the authors present a DVFS power management evaluation framework for multiprocessors based on an IBM system simulator, which limits the applicability of this approach to dedicated IBM architectures.

In-system approaches present DVFS power management techniques exploited in silicon system implementations. Calhoun et al. present a 90nm test chip that implements DVFS with the purpose of energy reduction for fixed throughput systems [34]. In [35], Lee et al. develop a hardware-software power control scheme to reduce the power consumption of a commercial processor by applying frequency and voltage adaptions. These adaptions are performed by a simple power control chip. Hotta et al. propose a profile-based power and performance optimization system based on DVFS adaptions for PC clusters [36].

While *simulation-based DVFS techniques* offer the investigation of power management techniques in an early design stage, they often lack the consideration of the exact power consumption of the system as well as the voltage regulator impact to enable DVFS. Moreover, accurate power consumption considerations are often neglected and lead to extensive simulation times if carried out on a simulational basis. *In-system approaches* suffer from expensive redesigns if potential flaws of the investigated DVFS-based power management techniques are discovered, since the final system has already been manufactured.

2.2.2 Power-Budget-Aware Power Management Techniques

Environmentally-powered systems, such as RF-powered smart cards, have driven power-budget-aware power management techniques, which are a niche of power management techniques. These techniques focus on adapting the system to present power availability conditions in contrast to general power management techniques, that focus on power and energy efficiency enhancements and operating time extensions.

In [37], Cebrian et al. propose various microarchitectural measures to accurately match power constraints. They evaluate their work by means of the HotLeakage power-performance simulator [38]. Similarly, another microarchitectural power management technique used to maintain power constraints is shown in [39] incorporating DVFS techniques. This approach is applied to multiprocessor designs based on a full system simulator that incorporates a power macromodel. Meng et al. propose an alternative simulation-based power-budget-aware DVFS power management technique for multiprocessor systems [40]. The system simulator includes dynamic power models of the widely used power simulator *Wattch* [41].

2.2.3 Power Profile Flattening Techniques

RF-powered smart cards acquire power via an externally generated RF-field. This provides only a limited amount of power dependant on the field strength and the distance between the smart card and the RF-field generating reader device. Power peaks (i.e., regions that exceed the power limit) that are often caused by large power consuming system modules (e.g., cryptographic coprocessors) can cause the supply voltage of the system to drop below a critical limit and in a way that stable system operation can not be guaranteed. A number of countermeasures used to ensure reliable system operation despite of power peaks have been proposed. These works can be grouped in two categories: (i) *software power profile flattening* and (ii) *hardware power profile flattening.*

Software Power Profile Flattening

Software power profile flattening techniques proposed in the past aim at the insertion of non functional instructions (NFI's) by modifying program code sections causing power peak regions [42, 43]. An automated software power peak optimization method is proposed in [44] by performing optimizations at the compiler level and by instruction reordering. In [45], an automated framework for power peak optimization is introduced based on NFI insertion and frequency scaling. However, the major drawback of software power profile flattening techniques is the determination of the number and the location of NFIs or the required system frequency to avoid power peaks. This information is not known during compile time, which might require multiple optimization iterations until satisfying results are achieved.

Hardware Power Profile Flattening

With the advent of *hardware power profile flattening techniques*, these limitations are circumvented by utilizing dedicated flattening hardware that performs run-time NFI insertion based on power profile analysis. In [42], power profiles are obtained by on-chip analog power measurements, while power constraints are held in registers in a digital manner. This requires D/A-converters to perform the decision making whether NFI insertion is necessary alongside additional analog measurements on-chip.

A second variant of *hardware power profile flattening* is based on current injection methods presented in form of digital as well as analog solutions [46]. Power profile flattening is achieved by forcing redundant switching activity or by adjusting a dedicated current sink depending on the present system's power consumption. In [47], a current injection method combined with the scaling of the supply voltage is presented. Current injection methods require a current reserve to keep the overall power consumption constant despite of a potentially highly variable power consumption. This current reserve reduces system efficiency.

Ongoing research in the field of hardware power profile flattening techniques mainly relies on analog power measurements or current injection methods. In order to investigate power profile flattening techniques in an early design phase, accurate and rapid power consumption information is required to be available. Hence, a purely digital approach of power profile flattening would simplify these techniques and, if employed on an FPGA prototyping platform, early design phase investigation can be ensured.

2.2.4 Supply Voltage Drop Compensation Techniques

Power profile flattening techniques aim to avoid harmful supply voltage drops that might hinder the system's standard operation. However, these techniques can be suboptimal. In some cases optimizations might be too aggressive by flattening power peaks that actually do not cause harmful voltage drops. Hence, using the supply voltage as a metric for voltage drop compensation algorithms is an interesting alternative to power profile flattening techniques. A number of works addressing supply voltage drop compensation techniques to ensure reliable system operation have been proposed.

Grochowski et al. present voltage simulation techniques in order to reduce supply voltage drops in [48]. The system is functionally simulated and current information is obtained from a current simulator. This current information is fed to a power distribution network that finally delivers supply voltage information. Using a feedback loop, the current consumption of the system is influenced by controlling clock gating logic, which in turn affects supply voltage variations. Joseph et al. propose a control technique whose purpose is to eliminate harmful voltage drops in [49]. A second order model of the supply network is simulated in MATLAB. An architectural level functional simulator combined with the *Wattch* power simulation tool [41] are employed to feed the supply voltage network and to derive voltage information. In [50], Reddi et al. set up a simulation environment based on a functional simulator, the *Wattch* power simulator and a second order power delivery model. Based on this environment, they evaluate a prediction scheme to reduce supply voltage fluctuations. A thermal-aware voltage drop compensation method is introduced in [51]. The temperature influence on the power consumption of a system is considered in order to optimize the system's performance while compensating voltage drops. The system is evaluated by means of a power delivery model simulated in SPICE, a multiprocessor architectural simulator and the power simulator *Wattch*.

None of the existing early design phase voltage drop compensation works address the demand of simulation time required for obtaining power and supply voltage information. However, simulation times for complex applications can become extensively high [4]. Introducing a supply voltage emulation technique based on the high-level power emulation approach would yield a significant speed up for the investigation of supply voltage drop compensation techniques, while still providing early design phase applicability.

2.3 Summary

Early design phase investigation of power management techniques requires power analysis methods applicable before first silicon is available. A high-level power emulation approach operating in real-time at a low hardware overhead poses a promising solution to this issue. It forms the basis for the investigation of power management techniques, such as power profile flattening, without suffering from extensively high simulation times and late design phase applicability, which distinguish the main shortcomings of existing approaches. Moreover, the high-level power emulation approach provides the basis for a supply voltage emulation approach enabling rapid supply voltage estimations and hence, the efficient investigation of voltage drop compensation techniques. The joint integration of the power emulation and the supply voltage emulation approach on an FPGA prototyping platform incorporating power profile flattening and voltage drop compensation techniques allows for the early design phase investigation of the mentioned power management techniques. The choice of an apt power management technique helps to increase the power and energy efficiency of these systems as well as to ensure their reliability.

Chapter 3

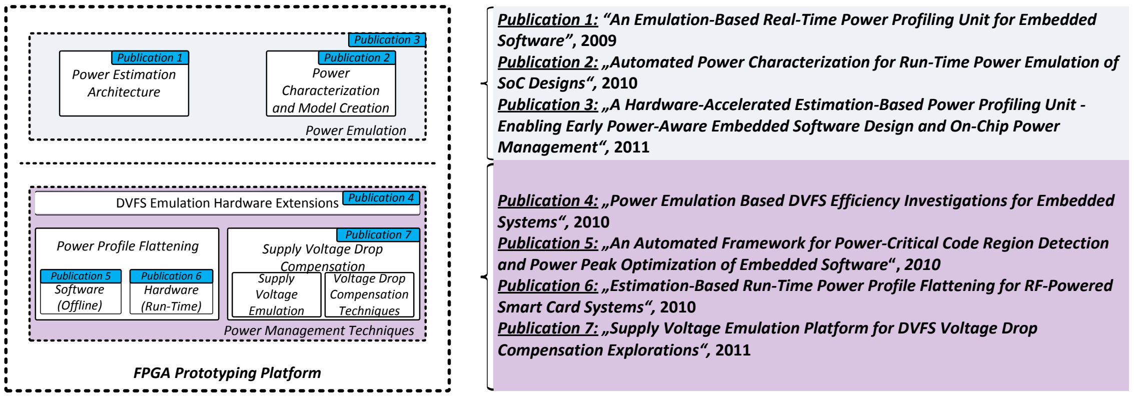
Early Design Phase Power Management Based on Power Emulation

3.1 Overview

Power emulation provides a promising approach to address limitations of existing power analysis techniques, such as late design phase applicability on the one hand and extensively high simulation times to obtain power information on the other hand. However, existing emulation-based methods still suffer from high hardware overhead and the lack of real-time capability. To overcome these issues, in this chapter, a high-level power emulation approach is presented. This approach makes power analysis significantly faster at moderate area overheads compared to existing power analysis methods, thus enabling the early design phase investigation of power management techniques, such as power profile flattening as well as supply voltage drop compensation techniques. Figure 3.1 illustrates work packages of this thesis contributing to an FPGA-based emulation platform allowing for early design phase power management investigations. The significance of this work has been proven in numerous publications linked to each work package. This chapter provides an overview on the delivered work packages, while details can be obtained from corresponding publications listed in Chapter 6.

In *Publication 1*, a power estimation architecture is introduced enabling hardware-accelerated power estimation, which forms the basis for the high-level power emulation approach. Moreover, a case study shows the use of the power emulation approach for power-aware optimizations intended for embedded software designers.

The setup of the underlying power model implemented in the power estimation architecture is discussed in *Publication 2*. In addition, this paper addresses an automated power model creation and power characterization methodology that enables the generic applicability of the power emulation approach to various systems. Both work packages are essential components for the high-level power emulation approach yielding real-time capability at low hardware overhead. More detailed information, rich experimental data and the principle of potential power management techniques based on the high-level power emulation approach can be found in *Publication 3*.


Publication 3
Publication 1
Power Estimation Architecture
Publication 2
Power Characterization and Model Creation
Power Emulation
DVFS Emulation Hardware Extensions
Publication 4
Power Profile Flattening
Publication 5
Software (Offline)
Publication 6
Hardware (Run-Time)
Publication 7
Supply Voltage Drop Compensation
Supply Voltage Emulation
Voltage Drop Compensation Techniques
Power Management Techniques
FPGA Prototyping Platform

Publication 1: ***"An Emulation-Based Real-Time Power Profiling Unit for Embedded Software"*****, 2009**
Publication 2: ***„Automated Power Characterization for Run-Time Power Emulation of SoC Designs"*****, 2010**
Publication 3: ***„A Hardware-Accelerated Estimation-Based Power Profiling Unit - Enabling Early Power-Aware Embedded Software Design and On-Chip Power Management"*****, 2011**

Publication 4: ***„Power Emulation Based DVFS Efficiency Investigations for Embedded Systems"*****, 2010**
Publication 5: ***„An Automated Framework for Power-Critical Code Region Detection and Power Peak Optimization of Embedded Software"*****, 2010**
Publication 6: ***„Estimation-Based Run-Time Power Profile Flattening for RF-Powered Smart Card Systems"*****, 2010**
Publication 7: ***„Supply Voltage Emulation Platform for DVFS Voltage Drop Compensation Explorations"*****, 2011**

Figure 3.1: Work packages and corresponding publications that enable early design phase power management investigations based on the high-level power emulation approach

Power management techniques enabling more power-efficient and energy-efficient as well as more reliable embedded systems designs can be investigated in an early design phase by exploiting the power emulation approach. This work focuses on power profile flattening and supply voltage drop compensation techniques based on dynamic voltage and frequency scaling (DVFS). The early design phase investigation of such power management techniques on an FPGA prototyping platform requires hardware extensions to enable DVFS. *Publication 4* addresses the modeling of DVFS and corresponding hardware extension issues. *Publication 5* and *Publication 6* deal with software and hardware power profile flattening techniques exploring various measures for the reduction of harmful power peaks based on power consumption information delivered by the power emulation approach. Voltage drop compensation techniques mark a more sophisticated approach to counter the negative influence of supply voltage drops. In *Publication 7*, a supply voltage emulation approach based on the power emulation approach is presented.

3.2 High-Level Real-Time Power Emulation

Power emulation, which is representative of hardware-accelerated power estimation, allows for fast power analysis alongside functionally emulating a system-of-interest by means of an FPGA prototyping platform. However, high hardware overhead and the lack of real-time capability are major limitations of these existing power emulation approaches (refer to Chapter 2 for details). Raising the level of abstraction resulting in a high-level power emulation approach makes real-time power estimation at significantly reduced hardware overhead feasible.

3.2.1 Power Estimation Architecture

The proposed high-level power emulation approach is based on a power estimation architecture. Figure 3.2 depicts its principle structure and system integration.

Internal system states are made observable at the system's top level and are fed to power sensors that track state information of system modules. For accuracy purposes lowering abstraction layer information (e.g., state information of functional units of the

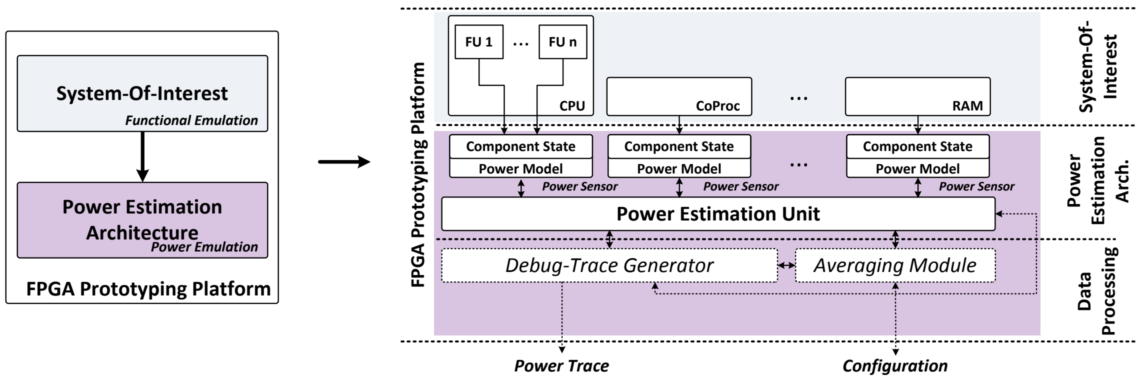


Figure 3.2: Power estimation architecture, obtained with modifications from [4]

system's CPU) can be considered. State-dependant information is stored in a software-configurable table that is integrated into the power sensors. This configuration ability makes the power estimation architecture applicable in a generic way to various systems. A power estimation unit accumulates the data on the power consumption of all system modules delivered by the power sensors, thus giving an instantaneous and cycle-accurate power estimate. An optional averaging module for post processing purposes allows for smoothing and denoising of power consumption information. Finally, power consumption information is assembled to power-trace messages by a debug-trace generator module to be transmitted from the FPGA prototyping platform to a host computer. In Section 6.1, the power estimation architecture, its system integration and its use in power-aware optimizations is discussed in more detail [4]. Moreover, additional experimental data on the high-level power emulation approach and its potential for power management applications are given in Section 6.3 [7].

3.2.2 Power Model and Characterization

Estimation-based power analysis is based on power models to deliver power estimates. The power estimation architecture is equipped with a power model based on linear regression methods that can be defined as

$$\hat{y}(t) = c_0 + \sum_{i=1}^{n-1} c_i x_i(t) + \epsilon. \tag{3.1}$$

$\mathbf{x(t)} = [x_1(t), x_2(t), ...x_{n-1}(t)]$ gives the time-dependant vector of model parameters, where $x_i(t)$ represent system states, such as CPU operating modes (e.g., idle, run) or memory accesses (e.g., read, write). The vector of model coefficients can be written as $\mathbf{c} = [c_1, c_2, ...c_{n-1}]^T$. c_0 represents the leakage power of the system-of-interest, while $\mathbf{c}$ account for the dynamic power consumption. The linear combination of model parameters $\mathbf{x(t)}$ and model coefficients $\mathbf{c}$ forms the power estimate $\hat{y}(t)$. The simplicity of this model makes it very apt for hardware integration. Model coefficients $\mathbf{c}$ are stored in a software-configurable table held in the power sensors. The deviation between the real power consumption $y(t)$ and the power estimate $\hat{y}(t)$ is stated as ϵ, which gives the estimation error.

Model coefficients **c** contain power consumption information and are determined during a power characterization process. These coefficients are found in three major steps: (i) Selection of model parameters $\mathbf{x(t)}$, (ii) selection of the training-set and (iii) utilizing a least squares fit method to determine the model coefficients **c**. The appropriate choice of model parameters c_i directly influences the hardware complexity and accuracy of the model. The chosen training-set determines how generic the model is applicable for a broad range of applications executed on the system, while still providing reasonable accuracies. Finally, the least squares fit method computes the model coefficients **c** by minimizing the sum of errors introduced by solving the overdetermined system of equations.

Seeking for model parameters $\mathbf{x(t)}$, determining the model coefficients **c** and the automation of the tedious power characterization process are discussed in Section 6.2 [52].

3.3 DVFS Emulation Hardware Extensions

Dynamic voltage and frequency scaling (DVFS) has become a popular power management technique in recent years. The continuous adaption of frequency and supply voltage enabled by voltage regulators has a cubic influence on the system's power consumption [53]. Ideally, a voltage regulator incorporates an infinite number of voltage levels allowing the switching between these levels in zero time. However, real voltage regulators do provide only a limited number of voltage levels, while each switching introduces a time delay until the voltage has settled to the next voltage increment. Figure 3.3 shows the differences between ideal and real DVFS behavior when executing a number of different tasks on a system. Power management based on dynamic frequency scaling (DFS) achieves power savings by slowing down the system while still meeting required task deadlines (3.3b) compared to an unoptimized system (3.3a). Introducing ideal DVFS yields a power and energy reduction (3.3c), however, considering the real voltage regulator behavior introduces a time and power/energy overhead (3.3d).

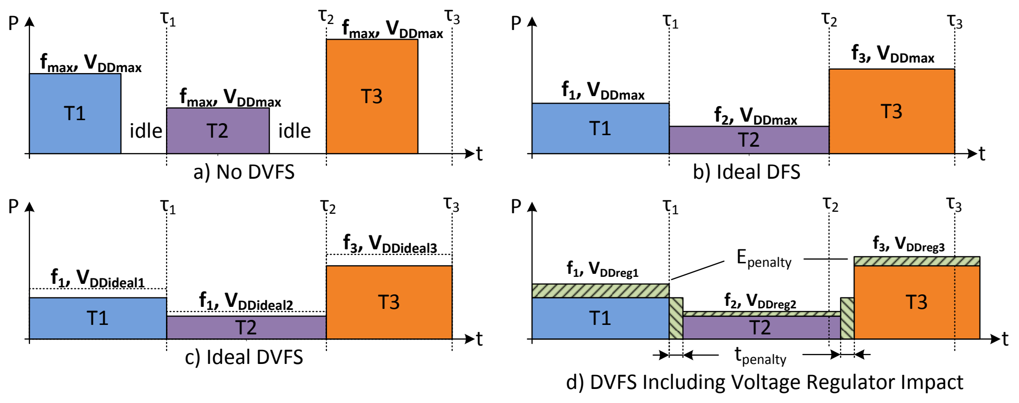


Figure 3.3: Power and energy reduction potential illustration of DVFS, obtained with modifications from [6]

In order to accurately model DVFS to explore power profile flattening or supply voltage drop compensation power management techniques on an FPGA prototyping platform, hardware extensions are required. The underlying DVFS models and corresponding hardware extensions are presented in Section 6.4 [6].

3.4 DVFS Power Management Techniques

Smart card systems powered by an RF-field generated by a reader device are operating reliably as long as the supply voltage remains above a minimum critical limit. Figure 3.4 shows the coupling of an RF-powered smart card system to a reader device. Power consumption peaks, caused by large power consuming system modules, such as cryptographic coprocessors, can cause the supply voltage to drop below this critical limit leading to system malfunctions.

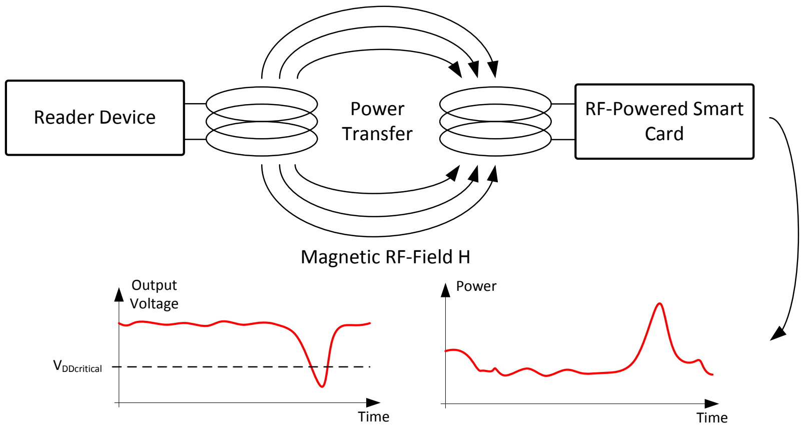


Figure 3.4: Power peak impact on the supply voltage of an RF-powered smart card system, obtained with modifications from [7]

This work focuses on the investigation of power management techniques to avoid severe supply voltage drops. This is achieved by power profile flattening as well as supply voltage drop compensation techniques. Both techniques exploit DVFS adaptions in order to influence the system's power consumption in a way that the supply voltage of the system is stabilized.

3.4.1 Power Profile Flattening Techniques

The high-level power emulation approach is exploited for power profile flattening techniques in a twofold manner. First, investigated power profile flattening techniques make use of power consumption information delivered by the power estimation architecture to steer DVFS adaptions. Second, power analysis of the investigated power profile flattening techniques can be performed by transferring power consumption information to a host computer for analysis purposes. Power profile flattening techniques can be distinguished by (i) *software power profile flattening* based on power critical source code detection and optimizations as well as (ii) *hardware power profile flattening* by DVFS adaptions.

Software power profile flattening utilizes the power emulation approach to analyze embedded software applications and to annotate power-critical code regions with corresponding power peaks. Based on that, the source code can be adapted to reduce the system's power consumption during the occurence of power peaks by non functional instruction (NFI) insertion or frequency scaling. An automated framework for software power profile flattening is explained in more detail in Section 6.5 [45].

Hardware power profile flattening adapts the power consumption of a system autonomously by DVFS adaptions based on power consumption information delivered by the power estimation architecture. Figure 3.5 depicts the principle structure of the proposed hardware power profile flattening technique. The system's power consumption is derived by the power estimation architecture and delivered to the power profile flattening controller, which controls the system frequency and the supply voltage by steering DVFS adaptions in a way that the system's power consumption converges to a given power budget.

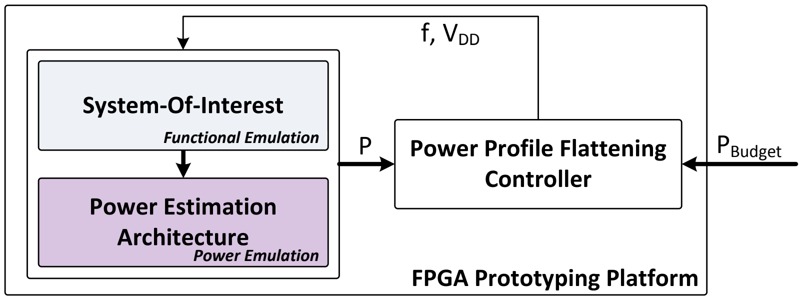

f, V_{DD}
System-Of-Interest
Functional Emulation
P
Power Profile Flattening Controller
P_{Budget}
Power Estimation Architecture
Power Emulation
FPGA Prototyping Platform

Figure 3.5: Hardware power profile flattening principle

The FPGA prototyping platform offers the early design phase investigation of power profile flattening techniques. After proving the feasibility by means of the FPGA prototyping platform, the approach is fully applicable to the final system by integrating the power estimation architecture and the power profile flattening controller on the final chip.

In Section 6.6 the proposed hardware power profile flattening technique is discussed in detail and flattening results for an RF-powered smart card system are presented [11].

3.4.2 Voltage Drop Compensation Based on Supply Voltage Emulation

Power profile flattening techniques aim at the reduction of harmful supply voltage drops by reducing the variability of the system's power consumption. Focusing solely on the power profile might aggressively flatten all power peaks and also might affect peaks that do not cause harmful supply voltage drops, which compromises system performance. A more sophisticated method to avoid supply voltage drops is to constantly analyze the supply voltage and adapt the system's power consumption according to the supply voltage level. For monitoring supply voltage levels efficiently in an early design phase, a supply voltage estimation architecture is introduced. This architecture delivers supply voltage information based on power consumption information that is derived from the data gathered by the power estimation architecture. In Figure 3.6, the principle structure of this concept is shown. Based on the present supply voltage level and a given minimum critical voltage, DVFS adaptions are performed to keep the supply voltage above the critical threshold.

In greater detail, the supply voltage emulation and compensation technique applied to a future multi-core RF-powered smart card system is illustrated in Figure 3.7. Each

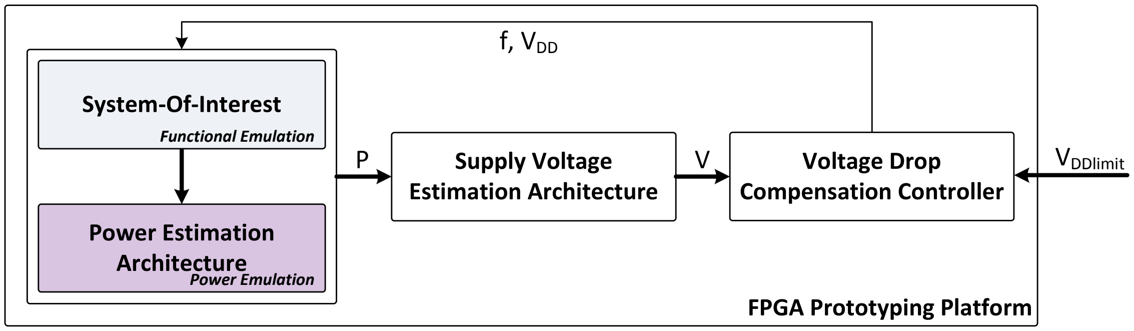

f, V_{DD}
System-Of-Interest
Functional Emulation
Power Estimation Architecture
Power Emulation
P
Supply Voltage Estimation Architecture
V
Voltage Drop Compensation Controller
$V_{DDlimit}$
FPGA Prototyping Platform

Figure 3.6: Supply voltage emulation and voltage drop compensation principle

core of a number of i smart card cores feeds system state information $\mathbf{x_i}$ to dedicated power estimation architectures that derive power information $P(\mathbf{x_i}(t))$, where $P(\mathbf{x_i}(t))$ gives the power consumption of core i for the system frequency f and supply voltage V_{DD} at which the power characterization process has been performed. However, in order to support DVFS adaptions, f and V_{DD} are required to be considered to scale the core's power consumption yielding $P(\mathbf{x_i}(t), f(t), V_{DD}(t))$. The overall power consumption $\mathbf{P}(\mathbf{x}(t), f(t), V_{DD}(t))$ of the system is given as the sum of each individual core's power consumptions as depicted in Equation 3.2. DVFS adaptions are provided globally, hence present system frequency settings f and supply voltage settings V_{DD} are applied to all cores equally.

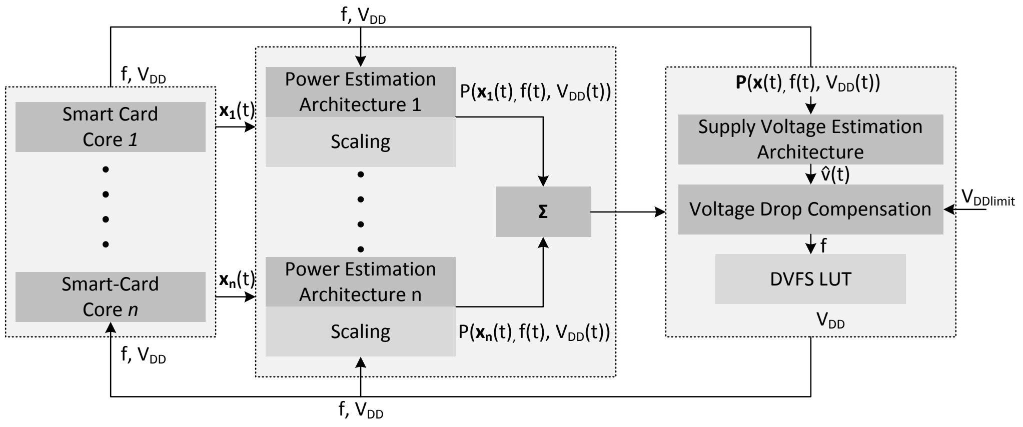

f, V_{DD}
f, V_{DD}
Smart Card Core 1
Smart-Card Core n
$\mathbf{x_1}$(t)
$\mathbf{x_n}$(t)
Power Estimation Architecture 1
Scaling
Power Estimation Architecture n
Scaling
P($\mathbf{x_1}$(t), f(t), V_{DD}(t))
P($\mathbf{x_n}$(t), f(t), V_{DD}(t))
Σ
P($\mathbf{x}$(t), f(t), V_{DD}(t))
Supply Voltage Estimation Architecture
$\hat{v}$(t)
Voltage Drop Compensation
$V_{DDlimit}$
f
DVFS LUT
V_{DD}
f, V_{DD}
f, V_{DD}

Figure 3.7: Multi-core supply voltage emulation and voltage drop compensation implementation, obtained from [8]

$$\mathbf{P}(\mathbf{x}(t), f(t), V_{DD}(t)) = \sum_i P(\mathbf{x_i}(t), f(t), V_{DD}(t)) \tag{3.2}$$

$\mathbf{P}(\mathbf{x}(t), f(t), V_{DD}(t))$ is then provided to the supply voltage estimation architecture that estimates the voltage $v(t)$, which is a measure for the voltage available at the multi-core RF-powered smart card system. The DVFS voltage drop compensation method compares supply voltage estimates $\hat{v}(t)$ to a given absolute minimum supply voltage limit

$V_{DD_{limit}}$. Based on the comparison result, a new system frequency set point f is computed. In order to determine the corresponding required supply voltage setting V_{DD}, a configurable DVFS lookup table (LUT) is provided. This table contains system frequency settings f and corresponding supply voltage levels V_{DD}. For each system frequency set point f, an according supply voltage lookup is performed and DVFS adaptions are carried out to hinder $v(t)$ from falling below $V_{DD_{limit}}$.

RF Energy Source Modeling

To accurately deliver the voltage $v(t)$, the behavior of the employed energy source must be modeled. An RF energy source powering a future multi-core smart card system provides only a limited amount of power at any given time. System load changes causing power peaks lead to supply voltage drops that can be modeled by an equivalent circuit of an RF energy source as illustrated in Figure 3.8 [9]. The voltage across the smart card system $v(t)$ can be modeled as a function of the load current $i(t)$ according to Equation 3.3, which is proportional to the power consumption delivered by the power estimation architecture. As long as the load current $i(t)$ is smaller than the current provided by the RF energy source $i_s(t)$, the output voltage is stable at V_z. If $i(t) \geq i_s(t)$, the missing current fraction $i(t) - i_s(t)$ can be provided by the capacitor C for a limited period of time. Starting from an initial condition V_0 at $t = 0$, the value of the supply voltage $v(t)$ at $t = T$ can be estimated as summarized in Equation 3.3.

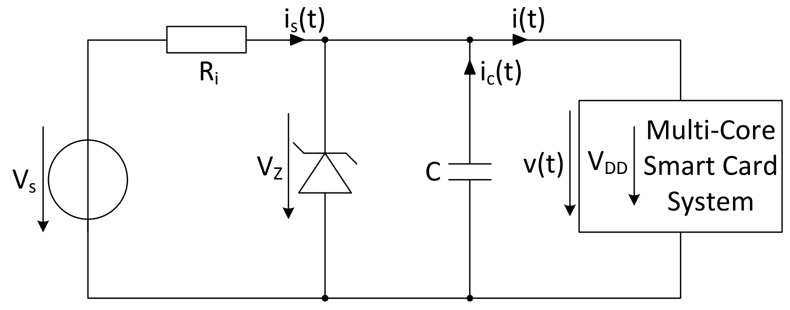


Figure 3.8: RF energy source equivalent circuit, obtained with modifications from [9]

$$\hat{v}(t) = \begin{cases} V_s - i(t)R_i \\ +(V_0 - V_s + i(t)R_i)e^{-\frac{T}{R_i\,C}} & \text{if } i(t) \geq i_s(t) \\ V_z & \text{else} \end{cases} \tag{3.3}$$

Supply Voltage Estimation Architecture

The proposed supply voltage estimation architecture computes a supply voltage estimate $\hat{v}(t)$ at each clock cycle depending on the current $i(t)$ drawn by the multi-core smart card system. Figure 3.9 shows its principle structure. The supply voltage estimation architecture is based on a 16-bit fixed-point implementation that is fed with power consumption information $\mathbf{P}(\mathbf{x}(t), f(t), V_{DD}(t))$ from the power estimation architecture as a proportional term for $i(t)$. Based on hardware adders and multipliers, the supply voltage estimate $\hat{v}(t)$

is computed according to Equation 3.3. The exponential term in Equation 3.3 marks the major obstacle to efficiently establish the supply voltage estimation architecture. By computing $\hat{v}(t)$ for a constant time frame $t = T$ (i.e., the clock period of the system), the exponential term can be reduced to a constant, since T, R_i, and C are constant. This circumvents more complex hardware structures to compute the exponential term, such as Cordic implementations [54]. The hardware effort is mainly determined by the multipliers complexity and could be reduced by multiplexing a single multiplier by compromising the evaluation speed of the supply voltage estimation architecture.

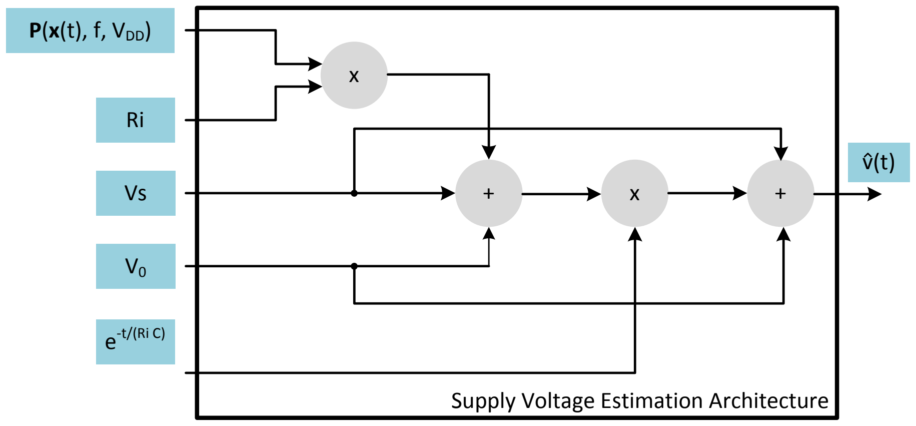


Figure 3.9: Basic internal structure of the supply voltage estimation architecture

Voltage Drop Compensation

The supply voltage estimates $\hat{v}(t)$ and a given minimum supply voltage limit $V_{DD_{limit}}$ form the basis for the DVFS voltage drop compensation technique. Figure 3.10 depicts a state machine based on a greedy strategy aiming at the minimization of supply voltage drops.

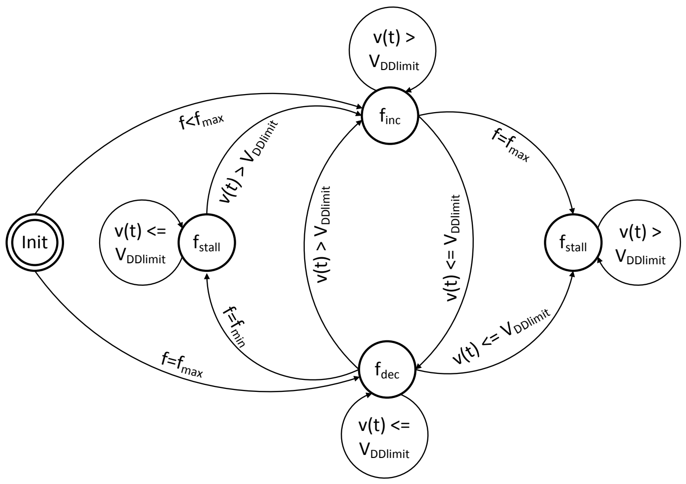


Figure 3.10: Voltage drop compensation technique by means of a greedy-based state machine

Depending on the comparison result of $\hat{v}(t)$ and $V_{DD_{limit}}$, a new system frequency set point f is computed and the supply voltage V_{DD} is determined by an according table lookup in the DVFS lookup table. Starting from an *Init* state, the state machine changes to f_{inc} or f_{dec} states depending on the present supply voltage level. In the f_{inc} state, the system frequency is increased as long as $\hat{v}(t) > V_{DD_{limit}}$, else a voltage drop occurred and a state transition to f_{dec} is performed. The algorithm continues to stay in f_{dec} as long as $\hat{v}(t) \leq V_{DD_{limit}}$. Moreover, two f_{stall} states exist that prevent the system from running out of bounds if the state machine reaches the minimum or maximum possible system frequency settings. The proposed supply voltage estimation and supply voltage compensation techniques are intended for early design phase investigations implemented on an FPGA prototyping platform. The published idea of this work can be found in Section 6.7 [8].

Chapter 4

Evaluation and Case Studies

The purpose of this chapter is to provide a comprehensive feasibility study for the developed concepts enabling the early design phase investigation of power management techniques. The presented high-level power emulation approach provides the basis for the early design phase investigation of power profile flattening and supply voltage drop compensation techniques. This chapter presents a proof of concept of the real-time capability and low hardware overhead stated in this work. Moreover, a power profile flattening technique is investigated for an RF-powered smart card system aiming at the avoidance of harmful supply voltage drops. Finally, the effectiveness of a supply voltage drop compensation technique by means of supply voltage emulation is shown. The approach is evaluated on a future dual-core smart card system. Note that for confidentiality reasons all published results are normalized.

4.1 Evaluation Systems

Two systems are used to evaluate the concepts presented in this thesis. The high-level power emulation and power profile flattening concepts are first evaluated on a conventional state-of-the art RF-powered smart card system. The supply voltage emulation and the supply voltage drop compensation technique are implemented and evaluated on a dual-core test system, which is more representative of sophisticated future RF-powered smart card systems.

4.1.1 Conventional Smart Card Evaluation System

Figure 4.1 illustrates the investigated RF-powered smart card system incorporating the power estimation architecture. The smart card system is based on a 16-bit pipelined cache architecture. It incorporates volatile and non-volatile memories, a memory encryption/decryption unit, cryptographic coprocessors as well as random number generators and communication interfaces. Both, the smart card system and the power estimation architecture are synthesized on an FPGA prototyping platform yielding the high-level power emulation evaluation platform. Power profiles obtained from this platform by executing a number of benchmarking applications are transferred to a host computer for evaluation purposes.

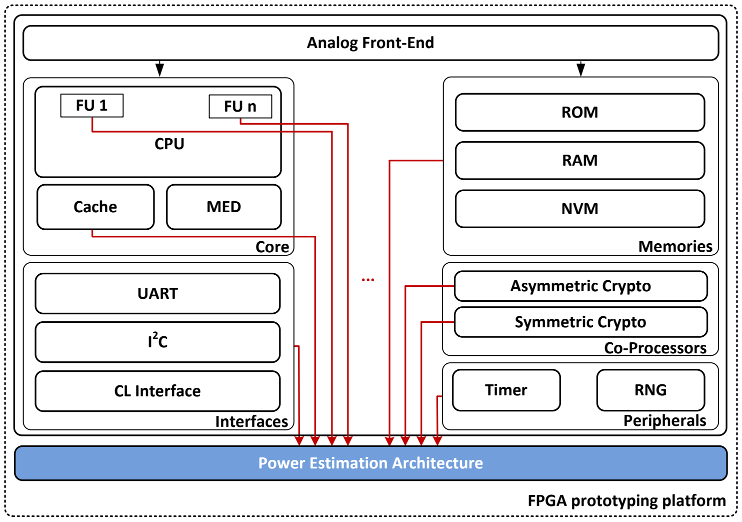


Figure 4.1: Block diagram of a typical RF-powered smart card system incorporating the power estimation architecture, obtained with modifications from [7]

4.1.2 Dual-Core LEON3 Smart Card Evaluation System

The dual-core smart card evaluation system is based on a LEON3 system-on-chip provided by *Aeroflex Gaisler* [10]. The principle structure of the system is depicted in Figure 4.2. The original dual-core system is extended with the power estimation architecture enabling high-level power emulation, the supply voltage estimation architecture enabling supply voltage emulation and the supply voltage drop compensation controller, each of which connected to the AMBA advanced peripheral bus (APB).

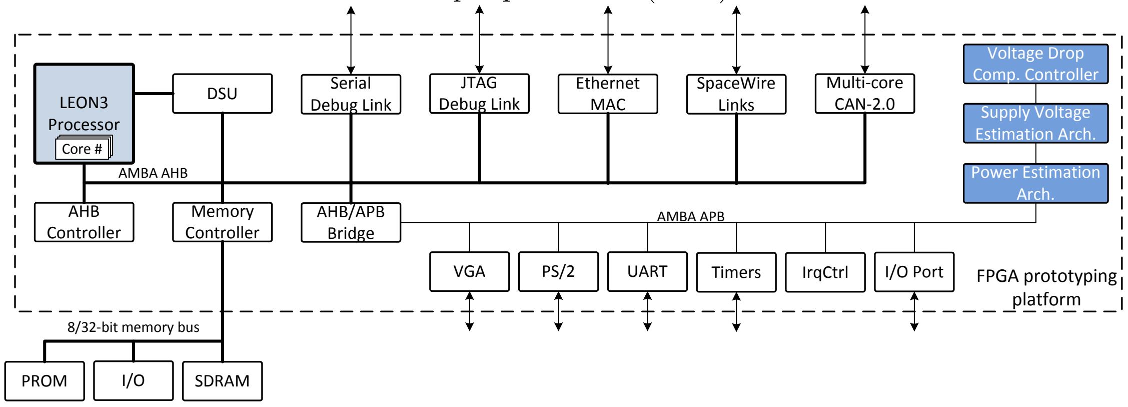


Figure 4.2: Principle structure of a future multi-core RF-powered smart card system incorporating the power estimation architecture, supply voltage estimation architecture and supply voltage drop compensation controller, obtained with modifications from [10]

Table 4.1 provides an overview on architectural system parameters. Moreover, the power model implemented in the power estimation architecture and DVFS adaption intervals are shown at which supply voltage drop compensation can be carried out.

Table 4.1: LEON3 architectural and power modeling parameters, obtained from [8]

Architectural parameters	
Architecture	32-bit SPARC V8, 7-stage pipeline
I-/D-Cache	1-way, 4 kByte, 32 Byte blocks
Nominal Frequency	33 MHz
Nominal Supply Voltage	1.5V
DVFS Interval	1 MHz / 0.1V, range: 1-33MHz, 0.5-1.5V
Technology	90nm
Power modeling parameters	
Modeled system modules	Core (Integer Unit, MMU) I-/D-Cache Reg. files
Modeling Technique	Least Squares Fit (Core) eCACTI (Caches, Reg. files) [55]
# Power model parameters	53 (Core), 6 (Caches), 2 (Reg. files)

4.2 High-Level Power Emulation

The quality of the power estimation approach is given by evaluating power estimation accuracy, power estimation performance and resource utilization. Typical smart card applications are executed on the RF-powered smart card system depicted in Figure 4.1. Power profiles to generate evaluation data are acquired by means of the power estimation architecture.

4.2.1 Power Estimation Accuracy

The power estimation accuracy of the high-level power emulation approach is compared to gate level simulations carried out with *Magma Blastfusion 5.2.2* [56]. Figure 4.3 illustrates the power estimation results obtained from the high-level power emulation approach compared to gate level power simulations.

In Section 6.1 and Section 6.2 average relative power estimation errors and corresponding variances are given for a number of applications. For the entire set of applications, estimation errors of less than 10% have been reported.

4.2.2 Power Estimation Performance

The proposed high-level power emulation approach is capable of acquiring power profiles in real-time, which is a major advantage over existing power emulation approaches. In Table 4.2, execution times of power simulations carried out with *Magma Blastfusion* are compared to power estimations obtained by means of the high-level power emulation approach. For the investigated set of applications, the power emulation approach delivers power profiles in real-time, while power simulations require hours of simulation time on a typical server system. This yields a speed-up of several orders of magnitude.

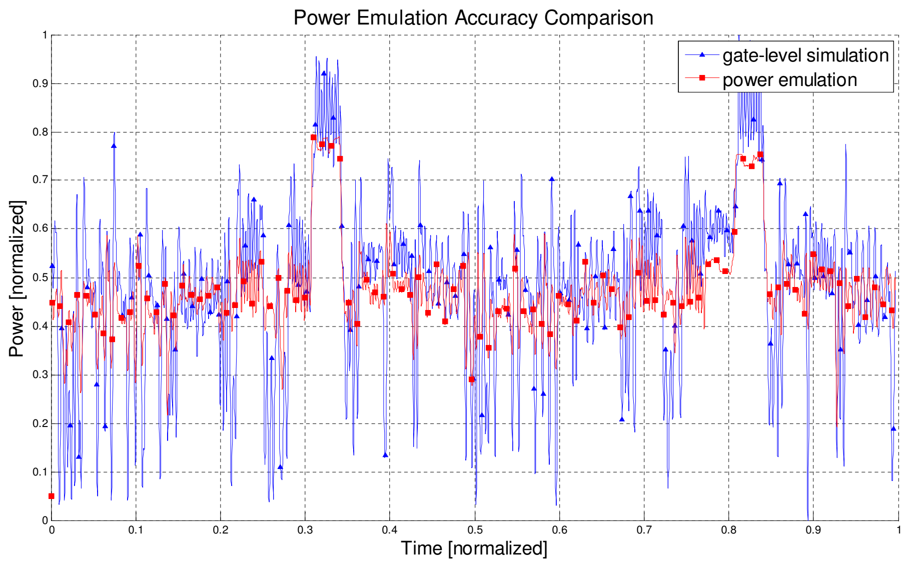


Figure 4.3: Power profile comparison of the power estimation result compared to a gate level power simulation, obtained with modifications from [7]

4.2.3 Resource Utilization

The hardware overhead added by the power estimation architecture to the smart card system has been evaluated by synthesizing the entire system on an *Altera Stratix II FPGA* [57]. A hardware overhead of 1.5% has been introduced by adding the power estimation architecture to the system. A break down of the introduced hardware overhead of the power estimation architecture is shown in Table 4.3. Additional power emulation results are provided in Section 6.1 and 6.2. Moreover, a case study employing the power emulation approach for power-aware software optimizations is presented.

4.3 Estimation-Based Power Profile Flattening

The investigation of power profile flattening by means of the high-level power emulation approach aims at the avoidance of harmful supply voltage drops. Results of an early design phase investigation are presented in this section. The evaluation system is similar to the one depicted in Figure 4.1 and has been extended with a power profile flattening controller. For details refer to Section 6.6.

4.3.1 Power Profile Flattening Evaluation Results

The power analysis results obtained for the power profile flattening technique are determined in a twofold manner. First, the power profile of a typical smart card application is analyzed and second, the power profile flattening technique is analyzed and the corresponding power profile is obtained. The original power profile illustrated in Figure 4.4 shows two distinct power peaks caused by the activation of large power consuming system modules, such as cryptographic coprocessors. It is apparent that one phase of low power

Table 4.2: Hardware-accelerated power estimation performance comparison to gate level power simulations, obtained from [7]

Application	# Cycles	Duration	
		Simulation [1] (hours)	Estimation-Based (microseconds)
ALU	2293	4.1	69.5
Cache	3978	14.9	120.5
Dhrystone	4176	18.6	126.5
Memory	1722	5.9	52.1
Payment (Coproc., CPU halt)	4510	17.8	136
Payment (Coproc., CPU run)	4837	23.8	146.6
DES (Coproc., CPU halt)	2899	9.5	87.8
DES (Coproc., CPU run)	3072	10.2	93.1

[1] Simulation has been performed on a state of the art server system. The sampling rate at which power simulations have been performed is 33 MHz, which is the clock frequency of the smart card system.

consumption originating from exploited low-power features (e.g., CPU sleep mode) occurs. The figure additionally shows frequency and internal supply voltage settings of the system as well as the external supply voltage simulated by SPICE [13]. It is evident that the two power peaks cause the external supply voltage to drop below a critical minimum limit when running the system at the fastest system frequency and the maximum internal supply voltage setting.

Figure 4.5 depicts the power analysis result of a flattened power profile achieved by DVFS adaptions steered by the power profile flattening controller. The two distinct peaks have vanished by reducing the system frequency and internal supply voltage within critical regions. This yields an external supply voltage that does not suffer from critical supply voltage drops. In addition, the power drop as shown in Figure 4.4 can be removed by boosting the system, which leads to a smoother power and supply voltage profile. A less

Table 4.3: Hardware break down of the amount of combinational and register resources allocated by the power estimation architecture. In total, the power estimation architecture accounts for 1.5% to the overall system resources allocated on an *Altera Stratix II FPGA*, obtained from [7]

	Combinational Logic (%)	Registers (%)
Bus IF, Load Mechanism [2]	32.6	7.2
Pwr. Sensors, PE Unit	58.1	78.5
Averaging	5.4	10.7
Debug-Trace Gen.	3.9	3.6

[2] The PE-architecture is interfaced to the system bus allowing for memory-mapped power sensor configuration via register accesses

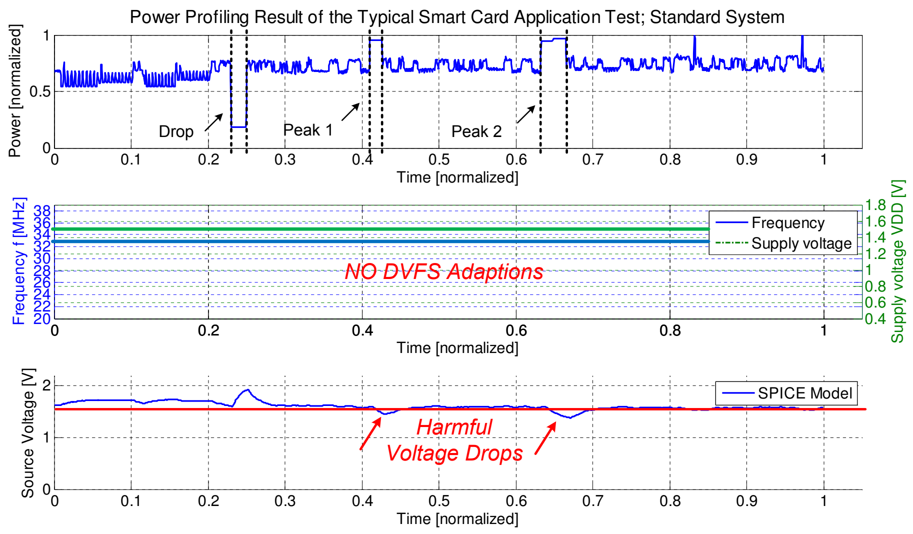


Figure 4.4: Power analysis result without employing frequency and supply voltage adaptions to achieve profile flattening, obtained with modifications from [11]

variable power profile and hence a supply voltage profile without harmful drops produces more reliable systems.

4.4 Supply Voltage Drop Compensation Based on Voltage Emulation

The proposed supply voltage drop compensation technique estimates the supply voltage behavior of a system based on its present power consumption. Supply voltage estimates are used to steer supply voltage drop compensation techniques to prevent the occurrence of harmful supply voltage drops. In order to achieve this in an early design phase, the system is extended with a supply voltage estimation architecture to provide supply voltage information. This information is fed to a supply voltage drop compensation controller that adapts the system frequency and supply voltage in a way that harmful drops are avoided. Power and supply voltage analysis results are presented in this section for typical smart card applications executed on a dual-core RF-powered smart card system emulated on an FPGA prototyping platform as depicted in Figure 4.2. In addition to supply voltage drop compensation results, supply voltage estimation accuracy and resource utilization results are shown.

4.4.1 Supply Voltage Drop Compensation Evaluation Results

A number of typical smart card benchmarking applications have been executed on the dual-core LEON3 system. A benchmarking application overview is given in Table 4.4. First, a power and supply voltage profile analysis by means of the power estimation and supply voltage estimation architectures for the *alu_arithmetic* and *alu_logical* benchmarking applications as depicted in Figure 4.6 have been carried out. By operating the system

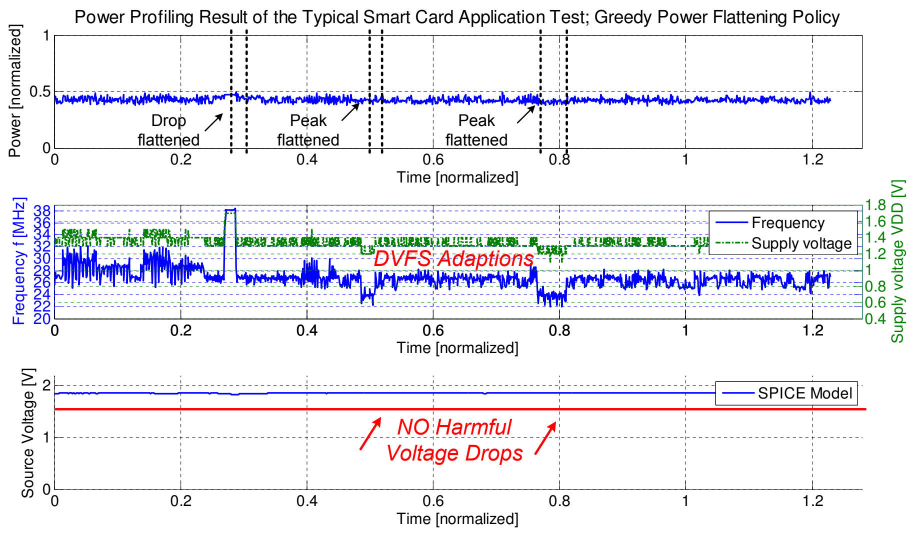


Figure 4.5: Power profile flattening result by employing frequency and supply voltage adaptions, obtained with modifications from [11]

at the maximum system frequency and maximum internal supply voltage, the superposition of the cores' power consumption leads to severe supply voltage drops jeopardizing system stability.

Table 4.4: Evaluation benchmarking applications executed on the dual-core LEON3 FPGA platform

	Core1	Core2
Profiling 1	alu_arithmetic (AA)	alu_logical (AL)
Profiling 2	RAM	ROM
Profiling 3	Coremark (CM)	Dhrystone (DS)
Profiling 4	Basicmath (BM)	Bitcount (BC)

After introducing the supply voltage estimation architecture and the supply voltage drop compensation controller, DVFS adaptions are performed to reduce the power consumption of the system within critical sections, which keeps the external supply voltage above a minimum critical limit. The corresponding results are depicted in Figure 4.7 that clearly show the avoidance of supply voltage drops.

4.4.2 Power Estimation and Supply Voltage Estimation Accuracy

Table 4.5 provides an overview on the power estimation accuracy provided for the evaluated dual-core LEON3 system. For all evaluated benchmarking applications an average relative error below 4% and a corresponding variance below 3% have been reported.

The supply voltage estimation results are provided in Table 4.6. For all benchmarking applications estimation errors and variances below 1.6% and below 0.16% are achieved compared to SPICE simulations, respectively.

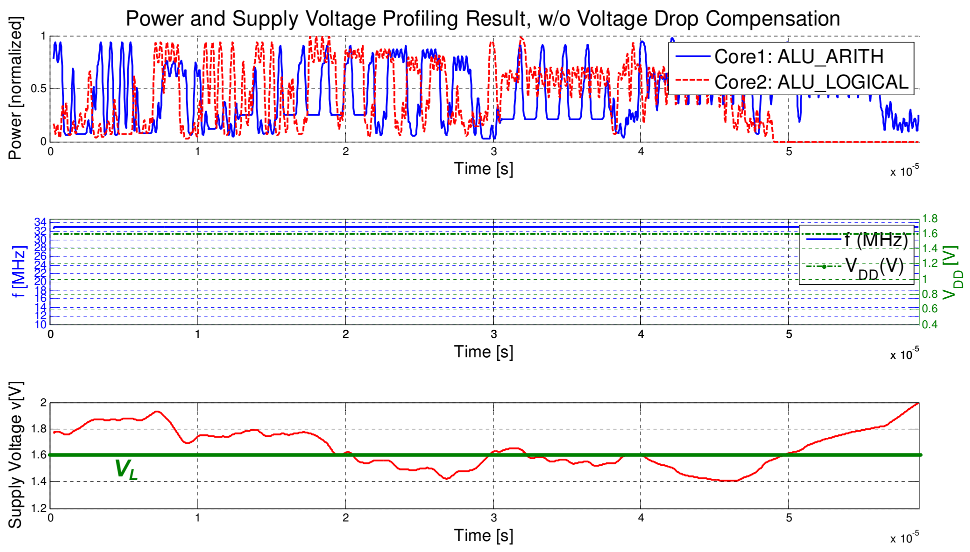


Figure 4.6: Power and supply voltage analysis result without employing DVFS adaptions, obtained from [8]

Table 4.5: Power estimation accuracy comparison to gate level power simulations for various benchmarking applications

	AA	AL	CM	DS
Power Error avg. [%]	-3.89	2.74	-2.29	0.95
Power error σ^2 [%]	2.33	2.46	3.08	2.62

4.4.3 Resource Utilization

Table 4.7 shows the resource utilization of the FPGA prototyping platform holding the dual-core LEON3 system. Occupied FPGA slices, slice registers and LUTs are compared for a number of LEON3 system modules. System cores incorporating I-caches and D-caches mark the largest modules requiring 11619 of 18207 slices. Additional modules introduced for power estimation, supply voltage estimation and supply voltage drop compensation purposes contribute only to 3.2%, 0.8% and 0.1% to the overall resource utilization, respectively. The impact of these modules can be considered as minor, thus making this power management technique an effective countermeasure against harmful supply voltage drops.

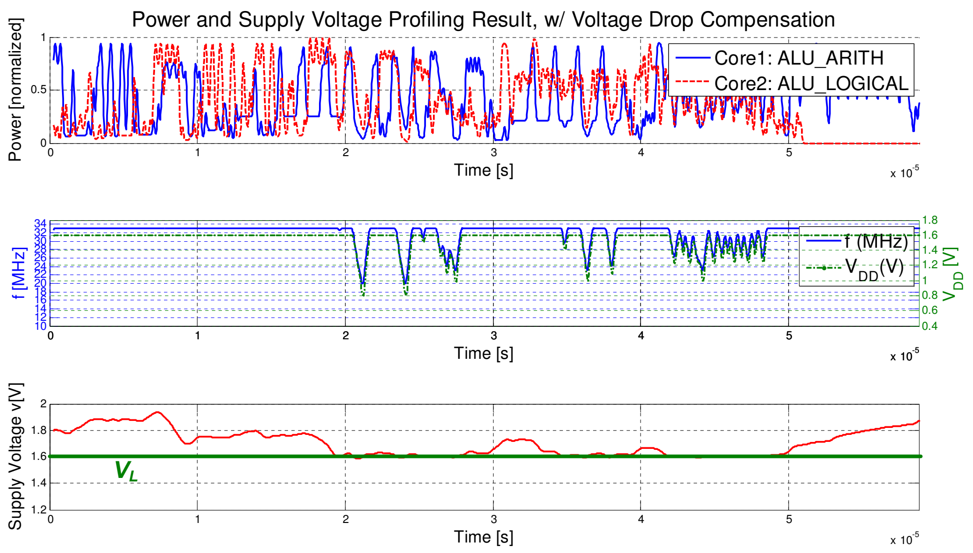


Figure 4.7: Supply voltage drop compensation result by employing DVFS adaptions, obtained from [8]

Table 4.6: Supply voltage estimation accuracy, compared to simulation results of the equivalent circuit obtained from LTSpice [13]

Application	Duration	Voltage Error	
	$[\mu s]$	avg. [%]	σ^2 [‰]
AA / AL	58.79	1.69	1.60
RAM / ROM	87.33	0.78	0.74
CM / DS	2325.36	0.87	0.86
BM / BC	4790.24	0.88	0.85

Table 4.7: Xilinx Spartan III XC3S2000 resource utilization overview

	Slices	Slice Registers	LUTs
Top	18207	8136	25887
Core1	5844	2714	8738
CPU	4787	2329	7129
Caches	1057	385	1609
Core2	5775	2714	8768
CPU	4745	2329	7155
Caches	1030	385	1613
PE Architecture + Scaling, DVFS LUT	586	529	612
SVE Architecture	161	20	289
VDC Scheme	23	10	34

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The enormous number of worldwide mobile systems accounts for an increasing portion of the overall global electricity consumption. Advances in CMOS technology leads to the possibility of more functionality for each new product generation. However, the associated extra power demands have not been accommodated by improvements in battery technology. Consequently, this constrains the exploitation of technology capabilities provided by CMOS technology advances. In order to counter these trends by enhancing power and energy efficiency of these systems while fully exploiting CMOS technology capabilities, the availability of power-aware system design tools and design methodologies has become a stringent requirement.

A high-level power emulation approach has been proposed in this work to address the lack of high-quality power-aware design tools. Compared to power simulations, power emulation offers a significant speed up for power analysis, while still being applicable in an early design phase. Power emulation is employed on an FPGA prototyping platform by integrating additional hardware into existing systems to derive power consumption information. The presented approach is superior in terms of power analysis evaluation speed and hardware overhead compared to existing approaches.

The real-time applicability of the high-level power emulation approach forms the basis to investigate power management techniques in an early design phase. In this work, main attention has been paid to power management techniques for RF-powered smart card systems. These systems are highly susceptible to system load changes, such as power peaks. Power profile flattening or supply voltage drop compensation techniques have been investigated for these systems in order to reduce their vulnerability. The early design phase investigation provides valuable information about the quality of the power management techniques and enables the designer to determine the most apt technique for a system-of-interest. The presented work provides a relief to the existing lack of efficient power-aware design tools and provides measures to leverage early design phase power consumption information for developing more power and energy-efficient systems or to increase the reliability of severely power-constrained systems.

5.2 Future Work

5.2.1 Low-Impact On-Chip Integration

The proposed high-level power emulation approach is not limited to the exploitation of early design phase power analysis and the investigation of power management techniques. The reported low hardware overhead of the presented power estimation architecture provides a promising alternative to gather on-chip power consumption information compared to analog on-chip power measurements. The proposed approach is purely digital and thus can be easily integrated into embedded systems designs to be exploited by power management techniques. This digital alternative drastically reduces complexity compared to mixed signal power management approaches as presented in [58].

5.2.2 Prediction-Based Power Management

Environmentally-powered systems require measures against supply voltage drops to ensure reliable system operation. In order to avoid high load changes caused by high power-consuming system modules (e.g., cryptographic coprocessors in high-security smart card systems or RF-transceivers in wireless sensor nodes) highly sophisticated power management features are required. The power emulation approach can be extended to perform power predictions, which could be accomplished by monitoring not only internal system state information, but also by observing pipeline state information (e.g., from the instruction-decode stage). Power prediction can further decrease the susceptibility of loosely powered systems to severe supply voltage drops.

5.2.3 Prospective Research

Two subsequent research projects exploiting the achievements of the POWERHOUSE project have been funded by the Austrian Federal Ministry for Transport, Innovation, and Technology. The long term perspectives of these projects are illustrated in Figure 5.1.

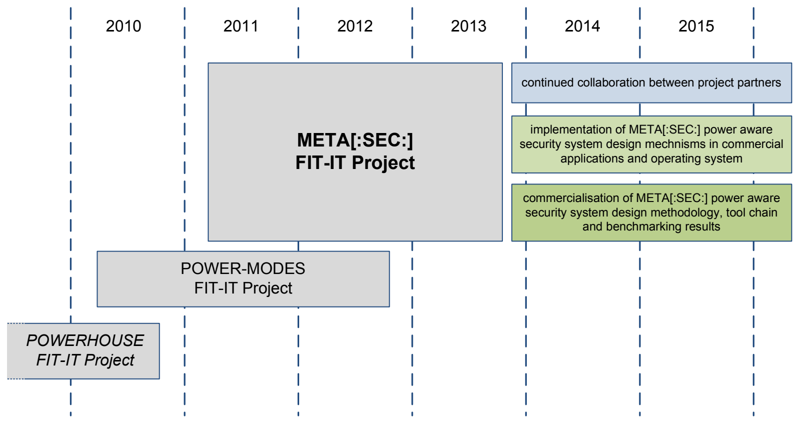

2010
2011
2012
2013
2014
2015
META[:SEC:]
FIT-IT Project
continued collaboration between project partners
implementation of META[:SEC:] power aware security system design mechnisms in commercial applications and operating system
commercialisation of META[:SEC:] power aware security system design methodology, tool chain and benchmarking results
POWER-MODES
FIT-IT Project
POWERHOUSE
FIT-IT Project

Figure 5.1: Long term perspectives of the POWERHOUSE project, obtained from [12]

The POWER-MODES[1] (*POWer EmulatoR and MOdel based DEpendability and Security evaluation platform*) project has been established to focus on power-aware design of security systems and the security strength analysis of fault detection mechanisms based on concepts developed in the POWERHOUSE project. Both projects incorporate power-aware design strategies to achieve power and security optimizations of stand-alone power-constrained systems.

The META[:SEC:][2] (*Mobile Energy-efficient Trustworthy Authentication Systems with Elliptic Curve based SECurity*) project unifies optimizations of smart card and reader systems to address future trends of portable readers, where both, the smart card and the reader system are severely power-constrained.

Establishing multiple research projects in the domain of power-awareness within the same institution incorporating industrial partners helps to exploit long-term synergies to make significant research contributions. This supports the improvement of future power-aware products yielding lower power and energy consumption as well as better technology exploitation and reliability.

[1]Project partners are Infineon Technologies Austria AG, Austria Card GmbH and TU Graz. The project is funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 825749.

[2]Project partners are Infineon Technologies Austria AG, RF-iT Solutions and TU Graz. The project is funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 829586.

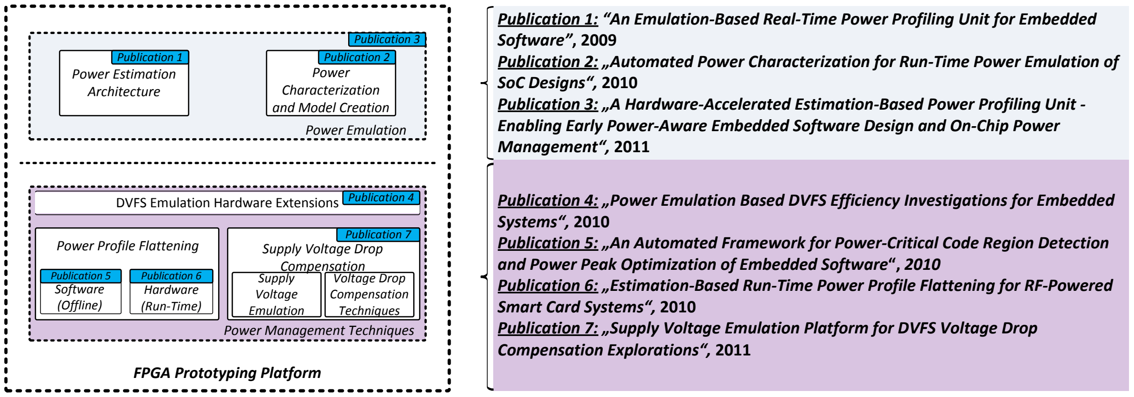
Chapter 6

Publications

This chapter comprises all relevant publications that have been presented at various conferences during the course of this work. Figure 6.1 illustrates relevant work packages and corresponding publications of this thesis, which finally yields an FPGA prototyping platform for early design phase power management investigations based on power emulation.

Publication 1 covers the concept and development of a power estimation architecture that forms the basis for the proposed high-level power emulation approach. Power consumption information acquired by the power estimation architecture is exploited on an FPGA prototyping platform allowing for the profiling and optimization of embedded software. In order to enable the derivation of accurate power consumption information from the power estimation architecture, a power model creation and power characterization process is required. In Publication 2, a methodology to establish a power model and the corresponding power characterization process is proposed. Moreover, a method to automatically perform these steps is presented, which enables the generic applicability of the high-level power emulation approach. Publication 3 provides additional information and experimental data on the high-level power emulation approach and presents an outlook on potential power management techniques enabled by the power estimation architecture. Publication 1 to 3 mark essential contributions to enable a high-level power emulation approach, which forms the basis for the early design phase investigation of power management techniques.

To allow the investigation of DVFS power management techniques in an early design phase, hardware extensions to emulate the DVFS behavior on an FPGA prototyping platform are required. This is presented in Publication 4. Based on the high-level power emulation approach and DVFS hardware extensions, the early investigation of power management techniques becomes feasible. Publication 5 presents software power profile flattening as a power management technique. By annotating power consumption information delivered by the power estimation architecture with the application's source code, the minimization of power peaks is achieved by automated source code adaptions. Alternatively, power profile flattening established in hardware is proposed in Publication 6. This approach aims at the minimization of harmful power peaks during run-time by performing DVFS adaptions. Finally, Publication 7 deals with supply voltage drop compensation techniques based on the high-level power emulation approach. A supply voltage emulation method exploiting power consumption information from the power emulation approach and voltage drop compensation techniques are presented to minimize harmful supply volt-


Publication 1
Power Estimation Architecture
Publication 3
Publication 2
Power Characterization and Model Creation
Power Emulation
DVFS Emulation Hardware Extensions
Publication 4
Power Profile Flattening
Publication 7
Supply Voltage Drop Compensation
Publication 5
Software (Offline)
Publication 6
Hardware (Run-Time)
Supply Voltage Emulation
Voltage Drop Compensation Techniques
Power Management Techniques
FPGA Prototyping Platform

***Publication 1:* “An Emulation-Based Real-Time Power Profiling Unit for Embedded Software”, 2009**
***Publication 2:* „Automated Power Characterization for Run-Time Power Emulation of SoC Designs“, 2010**
***Publication 3:* „A Hardware-Accelerated Estimation-Based Power Profiling Unit - Enabling Early Power-Aware Embedded Software Design and On-Chip Power Management“, 2011**

***Publication 4:* „Power Emulation Based DVFS Efficiency Investigations for Embedded Systems“, 2010**
***Publication 5:* „An Automated Framework for Power-Critical Code Region Detection and Power Peak Optimization of Embedded Software“, 2010**
***Publication 6:* „Estimation-Based Run-Time Power Profile Flattening for RF-Powered Smart Card Systems“, 2010**
***Publication 7:* „Supply Voltage Emulation Platform for DVFS Voltage Drop Compensation Explorations“, 2011**

Figure 6.1: Overview on early design phase power management investigations based on the high-level power emulation approach and corresponding publications

age drops. In the following paragraphs, all publications listed incorporate details on the developed concepts that are explained earlier in previous chapters.

Publication 1: *An Emulation-Based Real-Time Power Profiling Unit for Embedded Software*, IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, Samos, Greece, July 20–23, 2009

Publication 2: *Automated Power Characterization for Run-Time Power Emulation of SoC Designs*, Euromicro Conference on Digital Systems Design, Architectures, Methods, and Tools, Lille, France, September 1–3, 2010

Publication 3: *A Hardware-Accelerated Estimation-Based Power Profiling Unit - Enabling Early Power-Aware Embedded Software Design and On-Chip Power Management*, Springer Transactions on High-Performance Embedded Architectures and Compilers (Transactions on HiPEAC), 2011

Publication 4: *Power Emulation Based DVFS Efficiency Investigations for Embedded Systems*, IEEE International Symposium on System-on-Chip, Tampere, Finland, September 28–30, 2010

Publication 5: *An Automated Framework for Power-Critical Code Region Detection and Power Peak Optimization of Embedded Software*, International Workshop on Power and Timing Modeling, Optimization and Simulation, Grenoble, France, September 7–10, 2010

Publication 6: *Estimation-Based Run-Time Power Profile Flattening for RF-Powered Smart Card Systems*, IEEE Asia Pacific Conference on Circuits and Systems, Kuala Lumpur, Malaysia, December 6–9, 2010

Publication 7: *Supply Voltage Emulation Platform for DVFS Voltage Drop Compensation Explorations*, IEEE International Symposium on Performance Analysis of Systems and Software, Austin, Texas, April 10–12, 2011

An Emulation-Based Real-Time Power Profiling Unit for Embedded Software

Andreas Genser[1], Christian Bachmann[1], Josef Haid[2], Christian Steger[1] and Reinhold Weiss[1]
[1]Institute for Technical Informatics, Graz University of Technology, Austria
[2]Infineon Technologies Austria AG, Design Center Graz, Austria
{andreas.genser, christian.bachmann, steger, rweiss}@tugraz.at
josef.haid@infineon.com

***Abstract*—The power consumption of battery-powered and energy-scavenging devices has become a major design metric for embedded systems. Increasingly complex software applications as well as rising demands in operating times while having restricted power budgets make power-aware system design indispensable. In this paper we present an emulation-based power profiling approach allowing for real-time power analysis of embedded systems. Power saving potential as well as power-critical events can be identified in much less time compared to power simulations. Hence, the designer can take countermeasures already in early design stages, which enhances development efficiency and decreases time-to-market. Accuracies achieved for a deep sub-micron smart-card controller are greater than 90% compared to gate-level simulations.**

I. Introduction

Rising complexity of embedded software applications and the advance in processing power available in embedded systems require power analysis techniques to identify power saving potential. Furthermore, the detection of power-critical events, such as power peaks, which can affect system stability of energy-scavenging devices (e.g. contact-less smart-cards) is of great importance.

Among all abstraction layers the greatest power reduction potential can be identified on the application layer [1]. To enable the design of power-efficient software applications, power consumption feedback to the software designer should be available already at early design stages. However, commercially available power estimation and analysis tools are often operating on low abstraction layers, which are usually not available to the software designer. Moreover, low-level power simulations lead to extensive run-times. This makes power simulations for complex designs unfeasible.

Functional hardware emulation by means of prototyping platforms, such as FPGA-boards, has become a widespread technique for functional verification. Power information, however, is still in many cases gathered by power simulators. In this work, which is part of the PowerHouse[3] project, we propose an emulation-based real-time power profiling approach to circumvent this limitation. A given design augmented with power estimation hardware allows for obtaining power alongside functional characteristics in real-time. Power saving potential or power peaks can hence be detected earlier in the design cycle, which normally is not feasible before the design is available in silicon and actual physical measurements can be carried out.

[3]Project partners are Infineon Technologies Austria AG, Austria Card GmbH and TU Graz. The project is funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 815193.

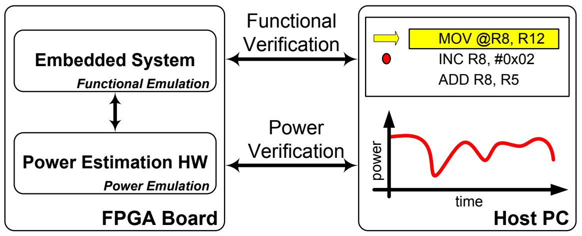


Fig. 1. Overview of an emulation-based power profiling approach for embedded systems comprising host computer interaction and visualization

By coupling this approach with a software development environment, valuable power information can be transfered to the software designer. This concept is depicted in Fig. 1. The FPGA-platform collects functional verification and power characteristics information, which is transmitted to a host computer. These information can be evaluated and visualized in a software development environment.

This paper is structured as follows. Section II provides information on previous work on power profiling. Section III briefly shows our research contributions. In Section IV the design of the real-time power profiling unit is discussed. Section V outlines a case-study applying the concepts developed in this work to a contact-less deep sub-micron smart-card controller and finally, conclusions drawn from the current work are summarized in Section VI.

II. Related Work on Power Profiling

Embedded software application power profiling can be categorized in (i) *measurement-based* and (ii) *estimation-based* methods.

Measurement-based methods are performed by taking actual physical measurements. This yields high accuracy compared to other approaches but requires additional measurement-equipment.

In contrast, power profiling by means of estimation methods is often based on power modelling. These techniques are usually less accurate but provide greater flexibility, since also power consumption for sub-modules of the system can be derived. In the following we compare ongoing research activities in the field of power profiling.

A. Measurement-Based Methods

In [2], PowerScope an energy profiling tool for mobile applications is introduced. The system's current consumption is automatically measured during run-time by a digital multimeter. Measurement data are collected for later analysis on a host computer.

An oscilloscope measurement-based profiling technique is proposed by Texas Instruments in [3]. The current drawn by a DSP system is profiled and results are visualized on a host computer in TI's software development environment.

B. Estimation-Based Methods

Power profiling by means of estimation techniques can be subdivided into (i) *simulation-based* and (ii) *hardware-accelerated* approaches.

Simulation-based power estimation executes programs on simulators to obtain circuit activity information. Power values are acquired using these information. In hardware-accelerated power estimation approaches, power information is derived from power models, which are implemented in hardware.

Estimation techniques can be employed on various levels of abstraction resulting in different estimation accuracies. Moreover, the degree of abstraction influences simulation times for simulation-based approaches and hardware-effort for hardware-accelerated methods. Real-time power estimation, however, is limited to hardware-accelerated estimation techniques.

Commercially available power estimation tools (e.g. [4]) operate on low abstraction levels, such as gate- or register-transfer level (RTL). Achievable estimation accuracies are high, while extensive simulation times render power estimation of elaborate applications unfeasible. On top of this, low-level simulators are often not available to software designers. Therefore, attempts to estimate the system's power consumption on a higher level of abstraction are carried out.

A *simulation-based* approach employing power models for instruction-level power estimations is proposed by Tiwari et al. in [5]. It allows for power and energy consumption estimation for given applications. The underlying power model considers the power consumption during instruction execution (i.e. base costs) and power consumption during the transition between instructions (i.e. circuit state overhead costs). In [6], Sami et al. consider additional microarchitectural effects to enhance the accuracy of instruction-level power estimation based on a pipeline-aware power model for Very-Long-Instruction-Word (VLIW) architectures. A co-simulation based power estimation technique is introduced by Lajolo et al. in [7]. This approach for System-On-Chips (SoCs) works on multiple abstraction levels. In principle, power estimation is performed on system level, while for refinement purposes and accuracy enhancements various components are co-simulated on lower levels of abstraction. Countermeasures against high simulation times are caching, statistical sampling and macro-modelling. A simulation framework for system-level SoC power estimation is introduced by Lee in [8]. This approach is based on power models developed for the processor, memories and custom IP blocks. Power values derived are provided cycle-accurately to the designer in a dedicated profile-viewer.

Hardware-accelerated power estimation techniques are performed by augmenting the given system with dedicated hardware blocks. A power characterization process performed beforehand determines power values, which are mapped towards corresponding power states. For example, hardware events (e.g. CPU idle/run states, memory read/write states, etc.) are representatives of such power states. For energy accounting, existing power estimation hardware can be extended to power state counters. In [9], Bellosa gathers information by means of hardware event counters to derive thread-specific energy information for operating systems. Joseph et al. obtain the power consumption of a system by exploiting existing hardware performance counters of a microcontroller [10]. A power macro-model based coprocessor approach for energy accounting is proposed in [11]. Energy events identified by energy sensors are tracked by a central controller. The additional power estimation hardware requires extra chip area but yields also a speed-up compared to simulation-based approaches.

Power emulation represents a special case of hardware-accelerated power estimation. FPGA-boards can be used as a typical prototyping platform to emulate not only the functional system behavior but also its power consumption. A given system comprising power estimation hardware is mapped onto an FPGA-platform. Functional verification and power estimation can be performed in real-time even before the silicon implementation of the system is available.

An overview of the power emulation principle is presented in [12]. Run-time improvements by power estimation hardware-acceleration of about 10x to 500x compared to commercial power estimation tools are achieved. Strategies to minimize the hardware overhead introduced by power estimation are proposed. In [13], this approach is extended to a hybrid power estimation methodology for complex SoCs. This framework combines simulation and emulation techniques, which significantly reduce power analysis times. In [14], the power consumption of processor cores is estimated employing power emulation to guide process migration between cores.

III. Contributions

Power profiling by means of physical measurements is typically very coarse-grained and limited to the entire chip due to chip integration and packaging. Moreover, the final chip is not available at early design stages.

Simulation-based power profiling techniques can be employed at the beginning of the design cycle. However, they are rendered unfeasible for complex applications due to extensive simulation times. To encourage the software designer to consider power aspects at early design stages, we provide a real-time emulation-based power profiling approach. Power

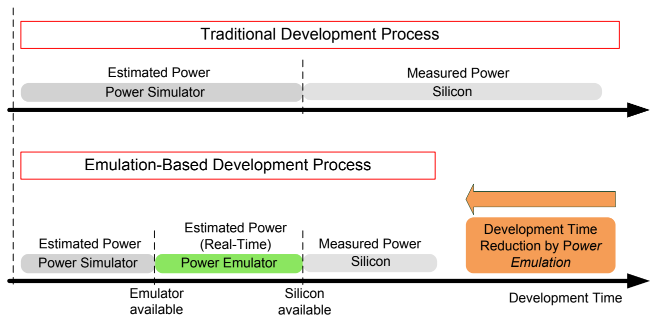


Fig. 2. Emulation-based vs. traditional power profiling approach

information is delivered to the software designer before silicon is available by utilizing an FPGA prototyping platform comprising power estimation hardware. Expensive redesigns caused by 'power bugs' can be avoided, which helps to decrease time-to-market (see Fig. 2).

The main goals of this work can be defined as follows:

- Deliver power information to the designer at early design stages to allow for:
  - Power-efficient software application design
  - Power-critical event detection
- Reduce development times

IV. Design of a Real-Time Power Profiling Unit

Estimation-based power profiling methods derive power information by exploiting power models. The abstraction layer on which these models are set up determines complexity and accuracy. Low-level models established on transistor- or gate-level are complex and therefore not suitable for emulation-based power profiling. In contrast, on a higher abstraction layer only main system components (e.g. CPU, memory, coprocessor, etc.) are taken into account. This leads to more compact models, hence real-time power profiling with moderate area increases is only feasible following this approach.

A. Power Model

Power models on a high level of abstraction are often based on linear regression methods. Details can be obtained in [15] and implementations are discussed in [5], [16]. A linear regression model can be given as

$$y = \sum_{i=0}^{n-1} c_i x_i + \epsilon. \qquad (1)$$

$\mathbf{x} = [x_1, x_2, ...x_{n-1}]$ gives the vector of model parameters. x_i represent system states, such as CPU modes (e.g. idle, run) or memory accesses (e.g. read, write) etc. The vector of model coefficients is given as $\mathbf{c} = [c_1, c_2, ...c_{n-1}]^T$. Each model coefficient c_i contains power information and has to be determined during a preliminary power model characterization process. The linear combination of model parameters $\mathbf{x}$ and model coefficients $\mathbf{c}$ form the power estimate y. The deviation between the real power value and its estimate y is stated by ϵ (i.e. the estimation error).

B. Power Characterization Process

Typically a linear regression model can be designed in three major steps.

(i) Selection of model parameters. The choice of model parameters directly influences the model's accuracy and is therefore of great importance. In addition, the cross-correlation between model parameters reflects the amount of redundancy in the model. This metric helps to keep a model as small as possible and thus efficient.

(ii) Selection of the training-set. The training-set is based on m power measurements for a number of m vectors $\mathbf{x^i}$, each of which containing n model parameters

$$\mathbf{x^i} = [x_0^i, x_1^i, ...x_{n-1}^i] \text{ for } 0 \leq i \leq m-1. \qquad (2)$$

Vectors $\mathbf{x^i}$ can be combined to the matrix

$$\mathbf{X} = [\mathbf{x^0}, \mathbf{x^1}, ...\mathbf{x^{m-1}}]^\mathbf{T}. \qquad (3)$$

Power values y acquired for corresponding vectors $\mathbf{x^i}$ can be expressed as

$$\mathbf{y} = [y_0, y_1, ...y_{m-1}]^T. \qquad (4)$$

Finally, $\mathbf{X}$ and $\mathbf{y}$ define the training-set and can be given as the tuple $\mathbf{T}$ in (5).

$$\mathbf{T} = (\mathbf{y}, \mathbf{X}) \qquad (5)$$

The linear regression model given in (1) can also be written in matrix-form. This is depicted in (6).

$$\mathbf{y} = \mathbf{Xc} \qquad (6)$$

Vectors $\mathbf{x^i}$ in $\mathbf{X}$ are derived from test applications (benchmarks) on a given embedded system and corresponding power values $\mathbf{y}$ are determined by physical power measurements or gate-level simulations.

(iii) Least squares fit method. The number of elements in the training-set $\mathbf{T}$ is usually much higher than the number of model parameters $\mathbf{c}$. This implies that the number of rows in $\mathbf{y}$ and the number of columns in $\mathbf{X}$ are higher than actually required to solve the linear system of equations in (6). Hence, the system is overdetermined and no exact solution exists. To overcome this issue model parameters are determined while minimizing the square error by using the least squares fit method.

The above steps can be carried out iteratively. If the model's accuracy does not meet the requirements, a model refinement by accounting more low-level information can be applied.

C. Power Emulation Architecture

The power emulation (PE) architecture that integrates the power model in hardware is illustrated in Fig. 3.

Power sensors are employed to track state information of system modules. For accuracy purposes also lower abstraction layer information can be considered (e.g. state information of functional units of the CPU). State vector and power

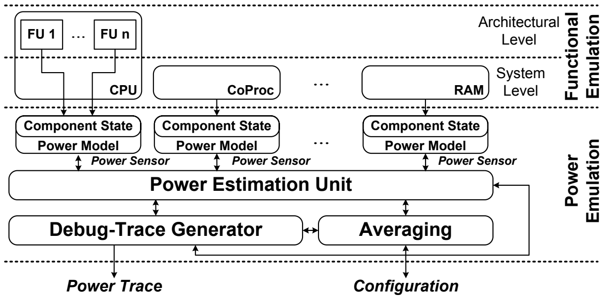


Fig. 3. Power emulation architecture

information are stored in a software-configurable table. These state information is mapped towards power values using a table-lookup approach. Fig. 4 depicts the principle structure of a power sensor module.

Each of a number of k power sensors covers l system states and contributes to the entire power model as expressed in (7). The PE-architecture delivers power information each cycle, hence time-dependency t is introduced in the following equations to account for power values estimated at different points in time.

$$y_{j,i}(t) = c_i\ x_i(t) \text{ for } 0 \leq i \leq l-1 \ \wedge \ 0 \leq j \leq k-1 \quad (7)$$

16-bit registers are provided to configure the power sensors with the power coefficients information obtained from **c**. It is worth noting that this power table can also be reconfigured during program run-time. This enables the masking of system modules, allowing the tracking of the power consumption of single sub-modules.

The power estimation unit accumulates 16-bit power sensor outputs according to (8). This constitutes an instantaneous, cycle-accurate up to 32-bit wide power estimate $y(t)$ for the overall system. The entirety of power sensors comprising the power estimation unit represent the power model established in hardware (see equality in (8)).

$$y(t) = \sum_{j=0}^{k-1}\sum_{i=0}^{l-1} y_{j,i}(t) = \sum_{i=0}^{n-1} c_i\ x_i(t) \quad (8)$$

Further post-processing is applied by the averaging module, which allows for smoothing and de-noising of a sequence of power values. This is enabled by a configurable moving average filter as shown in (9). Filtering properties can be changed by adjusting N.

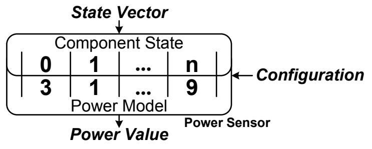


Fig. 4. Power sensor

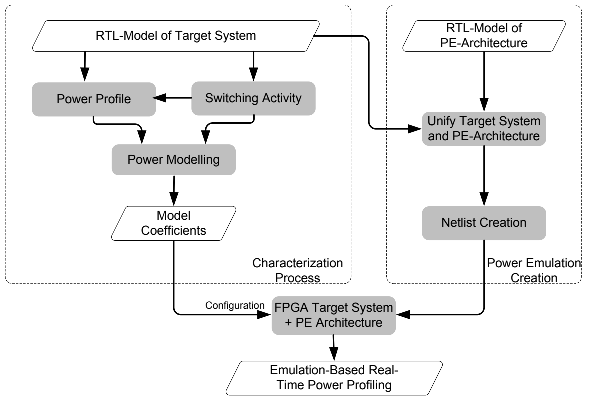

RTL-Model of Target System
RTL-Model of PE-Architecture
Power Profile
Switching Activity
Power Modelling
Model Coefficients
Unify Target System and PE-Architecture
Netlist Creation
Characterization Process
Power Emulation Creation
Configuration
FPGA Target System + PE Architecture
Emulation-Based Real-Time Power Profiling

Fig. 5. Design flow for emulation-based real-time power profiling

$$y_{avg} = \frac{1}{N}\sum_{j=0}^{N-1} y(t-j) \quad (9)$$

The debug-trace generator unit captures power information of the power estimation unit and the averaging module. A debug-trace message is composed out of these data and is delivered to the host computer for evaluation and further processing.

D. Design Flow

Fig. 5 outlines the design flow of the emulation-based real-time power profiling approach. A synthesizeable RTL-model of the target system is provided to perform the characterization process. After synthesis gate-level simulations based on benchmarking applications are performed and activity information as well as power profiles are acquired from value change dump (VCD) files. These information is fed to a power modelling process, deriving power model coefficients.

The target system RTL-models and the PE-architecture are merged to allow the generation of a single netlist. After downloading the netlist onto the FPGA-platform, power model coefficients determined beforehand are used to configure the power sensors for tailoring the PE-architecture to the given target system. Now applications of interest can be executed and real-time power profiles can be obtained.

E. System Set-Up

Power model coefficients obtained during the characterization process deliver configuration data for the power sensors. Listing 1 illustrates how to configure power sensors to tailor them to the power consumption of system modules. 16-bit registers are provided for this purpose.

```
// start of program

//configuration of power sensor1
PWRSEN0_STATE0 = 0x005A;   //CPU run mode
PWRSEN0_STATE1 = 0x0011;   //CPU halt mode
PWRSEN0_STATE2 = 0x0013;   //CPU sleep mode

//configuration of power sensor2
PWRSEN1_STATE0 = 0x0013;   //memory read
```

```
PWRSEN1_STATE1 = 0x001A;   //memory write
...
activate_power_emulation();
start_main_program();
```

Listing 1. Power emulation set-up, power sensor configuration

A variable number of system states for a variable number of system modules can be configured with power coefficients. Finally, power profiling is activated before normal application execution starts. The run-time overhead introduced due to power sensor configuration is negligible compared to the number of cycles executed by a typical application. A debug-trace containing power information is automatically generated and transfered to the host computer for power information evaluation and profile visualization without the interaction of the software designer.

If power analysis of sub-modules of the system is desired, the power sensor attached to the module of interest is configured as usual. The configuration of the remaining modules is skipped, so they do not contribute to the overall power consumption.

V. Case Study: Profiling of Power-Critical Smart-Card Applications

Smart-card applications have been penetrating manifold market segments in the last years. Access control, electronic passport or payment are only a few out of many existing applications.

Smart-cards in general can be categorized in (i) *contact-based* and (ii) *contact-less* derivatives. Contact-based smart-cards are powered if inserted into a reader device, while contact-less systems consume power via an RF-field generated by the reader. Therefore, contact-less devices are subjected to stringent power limitations.

Fig. 6 illustrates the coupling of the reader device with the contact-less smart-card by a magnetic field H. A certain amount of power is transfered from the reader device to the smart-card at a time. The available power is limited, hence exceeding a maximum power limit due to power-peaks affects system stability and can cause malfunctions. This case-study demonstrates the capability of our emulation-based power profiling approach to support the software designer early in the development process to avoid such worst-case scenarios.

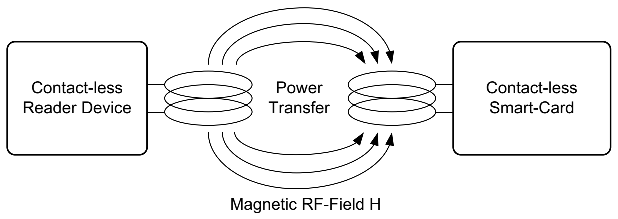


Fig. 6. Power supply of a contact-less smart card by a magnetic field generated by a reader device

A. Smart-Card Architecture Overview

Fig. 7 depicts a typical contact-less smart-card system. It is based on a 16-bit pipelined cache architecture comprising volatile and non-volatile memories. A symmetric coprocessor (SCP) is included for Advanced Encryption Standard (AES) and Data Encryption Standard (DES) algorithm acceleration. Moreover peripherals, such as a UART for communication purposes, timers or a random number generator (RNG) are provided. System modules are powered by an externally generated RF-field. Energy is collected by an antenna system and power supply conditioning and stabilization by means of an analog front-end are carried out.

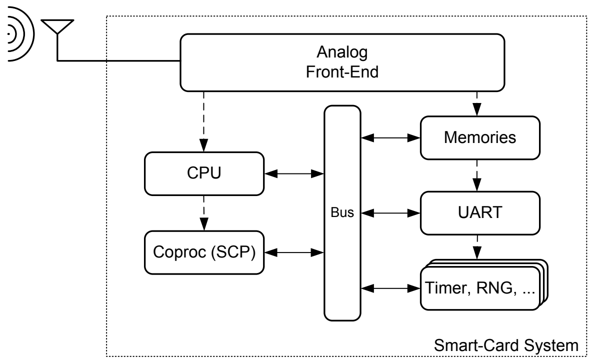


Fig. 7. Block diagram of a typical contact-less smart-card system

System modules that are major contributors to the overall power consumption are identified during a power characterization process. Moreover, available operating modes of each system module influencing the amount of power consumed are considered. A number of benchmarking applications to test many of these operating modes were applied. Based on the result of the characterization process, power model coefficients were obtained to configure the power sensors as shown in Listing 1. Table I summarizes system modules and corresponding operating modes relevant for the power model.

TABLE I
Operating Modes of Typical Smart-Card Components Considered in the Power Model

Unit	Mode(s)
CPU	run, halt, sleep
Cache	read, write (hit, miss)
Memories	read, write
UART	read, write
Peripherals	on, off
SCP	Encryption: AES128/192/256, (Single-, Double-, Triple-) DES
SCP	Decryption: AES128/192/256 (Single-, Double-, Triple-) DES

B. Payment Application Profile Analysis

A typical application for smart-cards incorporating a symmetric cryptographic coprocessor is payment. Payment applications usually contain authentication procedures requiring cryptographic operations. Fig. 8 illustrates a typical power profile of a future payment application obtained with the emulation-based power profiling approach. The first power peak marks the power consumption of an AES computation, while the remaining slightly smaller and shorter power peaks result from DES computations.

We assume that the maximum available power provided by the RF-field is 0.9 as shown in Fig. 8 (Note that power values are normalized). Hence, the payment authentication process would fail due to power peaks caused by the SCP.

If the source of these power peaks is not obvious to the designer already at this step, the power profile can be decomposed into sub-modules by reconfiguring power sensors. Fig. 9 depicts power profile results for the CPU, memories, SCP and UART decomposed for sub-module power profile analysis. As a reference also the cumulated power profile is shown.

The major contributor to the power consumption in this example is the CPU with more than 60%. Memories (including cache) and the SCP account for the remaining power consumption. The UART is inactive in this application and therefore consumes no power. It can clearly be seen that the SCP's power consumption causes the overall power profile to exceed the absolute maximum power of 0.9.

Various countermeasures could be taken to circumvent this issue. The AES algorithm could be implemented in software to avoid using the SCP. Another alternative is reducing the system clock frequency. Both solutions reduce the payment application's speed, but ensure reliable operation. Fig. 10 shows the power profile when scaling down the system frequency from 33 MHz to 28 MHz. Power peaks are below 0.9, hence system operation is stable.

C. Accuracy of Emulation-based Power Profiling

Fig. 11 shows power profiles of the payment application. A comparison between the emulation-based power profiling result and gate-level power simulation profiles obtained with *Magma Blastfusion 5.2.2* [17] are given. The relative error on average and the variance are 8.4% and 9.4%, respectively.

Accuracy considerations for other executed applications are summarized in Table II. The relative power error on average and the corresponding variance are given. Moreover, relative energy error values are depicted. For all tested applications relative power errors are less than 10% on average. Energy accounting reaches accuracies of greater than 93%.

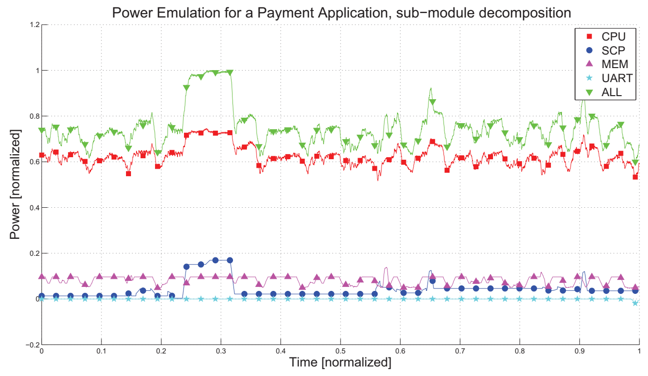


Fig. 9. Payment authentication application power profile decomposed for sub-module power analysis

TABLE II
POWER EMULATION ACCURACY COMPARISON FOR VARIOUS APPLICATIONS

Algorithm	Duration (μs)	Performance (Cycles)	Error (%) Power avg.	Error (%) Power σ^2	Error (%) Energy avg.
ALU1	70.8	2336	7.3	0.5	6.4
ALU2	44.2	1458	4.0	0.2	3.9
CPU	31.3	1032	-2.1	0.9	-3.2
Cache	12.4	4092	-1.5	0.9	-2.6
RAM	56	1848	-4.9	0.3	-5.5
SCP-AES128	13.5	4455	1.2	1.8	-0.2
SCP-AES256	15.7	5181	1.1	1.6	-0.2
SCP-DES	82.2	2712	1.1	0.8	0.3
SCP-DDES	76.9	2537	0.5	0.7	1.0
Payment	338	11160	8.4	9.4	2.0
Dhrystone	139	4587	0.4	2.0	-2.0

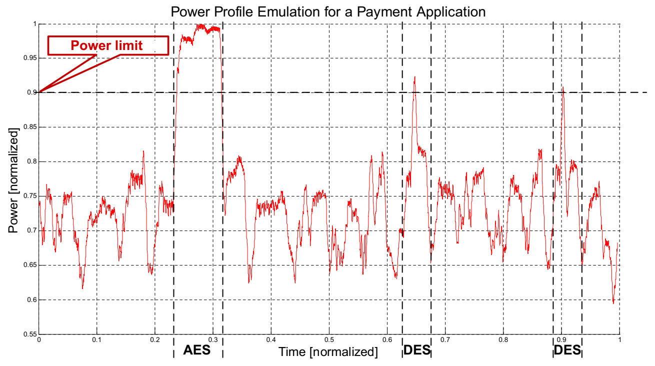


Fig. 8. Payment authentication application power profile obtained by emulation-based power profiling (power exceeds affordable level)

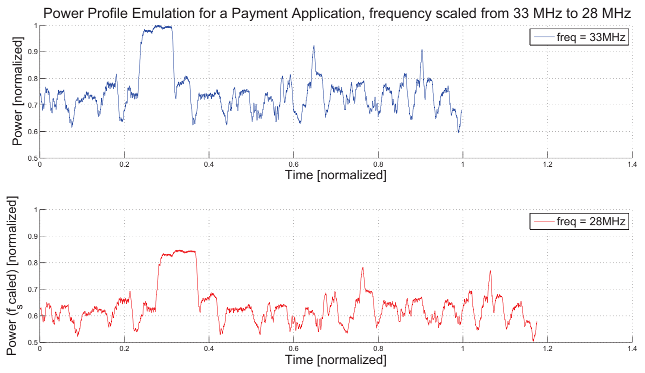


Fig. 10. Payment authentication application power profile obtained by emulation-based power profiling (stable system operation due to frequency-scaling)

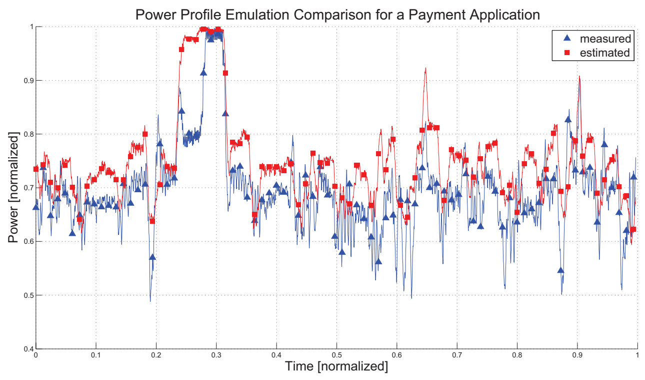


Fig. 11. Power profile comparison, gate-level power simulation profile vs. emulation-based power profile

D. *Performance Evaluation*

One of the major advantages of the emulation-based profiling approach is the capability to acquire power profiles in real-time. Power estimation takes no longer than application execution on the target-system. Hence, the power profile is available immediately after execution. Table III shows execution times of power profile simulations on a gate-level basis using *Magma Blastfusion* compared to the emulation-based approach. Extensively high simulation times are depicted compared to emulation run-times of a few hundreds of microseconds.

As illustrated in Table III, power emulation of the payment application takes 338 microseconds, whereas the power simulation is about 98 hours. Additional hardware introduced for power emulation contributes only to 1.5% to the overall system area.

It is obvious that the power simulation of complex applications is rendered unfeasible due to far too extensive simulation times. Therefore, power saving potential or power peaks leading to system failure as discussed in this section cannot be detected in an early design stage. By means of emulation-based power profiling, power estimates are available immediately and already when working with FPGA prototyping platforms. Countermeasures to circumvent power peaks causing system failures can be taken before the device is available on silicon and physical measurements are performed.

VI. Conclusion

Extensive run-times of power simulators render power analysis of increasingly complex embedded software applications unfeasible. The power profiling approach proposed in this work, delivers power information to the software designer in real-time. Moreover, by employing FPGA prototyping platforms, these power information is available already at early design stages. Emulation-based power profiling has proven to be an effective option to estimate a system's power consumption, delivering power information with accuracies of 90% on average. This paves the way for power-efficient embedded software design and the capability to cope with power critical events more efficiently.

TABLE III
Performance Comparison of Power Profiling for a Simulation and Emulation Approach

Algorithm	Performance	Duration	
	(Cycles)	Simulation[4] (hours)	Emulation (microseconds)
ALU1	2336	4.1	70.8
ALU2	1458	2.2	44.2
CPU	1032	0.78	31.3
Cache	4092	14.0	12.4
RAM	1848	2.9	56
SCP-AES128	4455	17.2	13.5
SCP-AES256	5181	24.0	15.7
SCP-DES	2712	5.5	82.2
SCP-DDES	2537	6.3	76.9
Payment	11160	98.3	338
Dhrystone	4587	18.1	139

[4] Simulation is performed at a sampling rate of 33 MHz, which corresponds to the clock frequency of the smart-card system. The server system for simulations comprises 12 CPUs and 50 gigabytes of physical memory for user processes.

References

[1] E. Macii and M. Poncino, *Power Macro-Models for High-Level Power Estimation*. CRC Press, edited by C. Piguet, 2005, ch. 39 Low-Power Electronics Design, pp. 39–1—39–18.
[2] J. Flinn and M. Satyanarayanan, "PowerScope: a tool for profiling the energy usage of mobile applications," in *WMCSA*, 1999, pp. 2–10.
[3] *Analyzing Target System Energy Consumption in Code Composer Studio^{TM} IDE*, Texas Instruments, 2002.
[4] J. Flynn and B. Waldo, "Power Management in Complex SoC Design," Synopsys Inc. White Paper, Tech. Rep., 2005.
[5] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis Of Embedded Software: A First Step Towards Software Power Minimization," in *ICCAD*, 1994, pp. 384–390.
[6] M. Sami, D. Sciuto, C. Silvano, and V. Zaccaria, "An instruction-level energy model for embedded VLIW architectures," in *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 21, 2002, pp. 998–1010.
[7] M. Lajolo, A. Raghunathan, S. Dey, and L. Lavagno, "Cosimulation-based power estimation for system-on-chip design," in *IEEE Trans. on Very Large Scale Integration Systems*, vol. 10, 2002, pp. 253–266.
[8] I. Lee, H. Kim, P. Yang, S. Yoo, E. Chung, K. Choi, J. Kong, and S. Eo, "PowerViP: SoC Power Estimation Framework at Transaction Level," in *ASP-DAC*, 2006, pp. 551–558.
[9] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *SIGOPS European Workshop*, 2000, pp. 37–42.
[10] R. Joseph and M. Martonosi, "Run-time power estimation in high performance microprocessors," in *ISLPED*, 2001, pp. 135–140.
[11] J. Haid, G. Kaefer, C. Steger, and R. Weiss, "Run-time energy estimation in system-on-a-chip designs," in *ASP-DAC*, 2003, pp. 595–599.
[12] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: a new paradigm for power estimation," in *DAC*, 2005, pp. 700–705.
[13] M. Ghodrat, K. Lahiri, and A. Raghunathan, "Accelerating System-on-Chip Power Analysis Using Hybrid Power Estimation," in *DAC*, 2007, pp. 883–886.
[14] A. Bhattacharjee, G. Contreras, and M. Martonosi, "Full-System Chip Multiprocessor Power Evaluations Using FPGA-Based Emulation," in *ISLPED*, 2008, pp. 335–340.
[15] A. Bogliolo, L. Benini, and G. D. Micheli, "Regression-based RTL power modeling," in *ACM Trans. on Design Automation of Electronic Systems*, vol. 5, 2000, pp. 337–372.
[16] C. Krintz and S. Gurun, "A run-time, feedback-based energy estimation model for embedded devices," in *CODES+ISSS*, 2006, pp. 28–33.
[17] "Magma Design Automation Inc., Blastfusion," (http://www.magma-da.com/), April 2009.

Automated Power Characterization for Run-Time Power Emulation of SoC Designs

Christian Bachmann*, Andreas Genser*, Christian Steger*, Reinhold Weiss* and Josef Haid†
*Institute for Technical Informatics, Graz University of Technology, Austria
†Infineon Technologies Austria AG, Design Center Graz, Austria
{andreas.genser, christian.bachmann, steger, rweiss}@tugraz.at
josef.haid@infineon.com

***Abstract*—With the advent of increasingly complex systems, the use of traditional power estimation approaches is rendered infeasible due to extensive simulation times. Hardware accelerated *power emulation* techniques, performing power estimation as a by-product of functional emulation, are a promising solution to this problem. However, only little attention has been awarded so far to the problem of devising a generic methodology capable of automatically enabling the power emulation of a given system-under-test. In this paper, we propose an automated power characterization and modeling methodology for high-level power emulation. Our methodology automatically extracts relevant model parameters from training set data and generates an according power model. Furthermore, we investigate the automation of the power model hardware implementation and the automated integration into the overall system's HDL description. For a smart card controller test-system the automatically created power model reduces the average estimation error from 11.78% to 4.71% as compared to a manually optimized one.**

I. Introduction

Power consumption has become a major design metric for electronic systems and mobile devices in particular. For these devices the power consumption severely affects operating time as well as system stability. In order to verify power requirements during the design phase of a given system, power estimation has become an essential part of the design process.

Due to the advances in process technology scaling as well as rising demands for computational performance and functionality, increasingly complex designs have to be handled in the power estimation process. Systems-on-chips (SoC) are typically composed of a large number of sub-components, each contributing to the overall power consumption. Furthermore, it can be observed that the power consumption of these devices is progressively more dependent on software applications, determining the utilization of system components as well as actuating available on-chip power management features. It is therefore favorable, to provide power estimation resources not only to system architects and hardware designers but also to power-aware software application and operating system developers.

For estimating the system's power consumption while executing elaborate program sequences (e.g., the booting sequence of an operating system), state-of-the-art gate- and RTL-level power simulators require extensive simulation times. Higher-level simulators, such as behavioral- or system-level tools, however fail at delivering cycle-accurate power estimates required for, e.g., evaluating the reaction of power management algorithms on power transients. This curtails the usability of these tools in power-aware software application development.

For this reason, hardware-accelerated power estimation approaches have been introduced, employing existing hardware counters [1–3], dedicated power estimation coprocessors [4, 5] and emulation-based approaches (i.e., *power emulation*) [6–10]. For the purpose of fast yet accurate software power estimation, high-level power emulation approaches [9, 10] as depicted in Figure 1 are considered promising. While requiring only small hardware overhead on a given FPGA already employed for functional emulation, they achieve a considerable power estimation speed-up compared to simulation-based methods. However, the time-consuming task of power model generation and required hardware description language (HDL) model adaptation for these power emulation approaches has, to the best of our knowledge, so far not been further investigated.

In the context of high-level power emulation, the novel contributions of this paper are as follows:

- We propose a systematic, automated power characterization and modeling methodology that automatically determines power model parameters for a given system-under-test.
- We develop an automated technique for implementing this power model in hardware and for integrating this generated power emulation hardware into the system-under-test.
- A case study on a smart card microcontroller test-system illustrates the benefits of our automated approach.

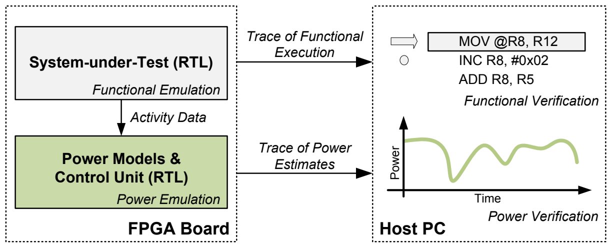


Fig. 1. Power emulation principle: Concurrent functional and power emulation (adapted from [10])

II. Related Work

Various hardware-accelerated power estimation techniques have been explored, leveraging existing hardware event counters [1–3], dedicated coprocessors [4, 5] and emulation-based approaches [6–10].

By correlating hardware events and power consumption, initial works have been using existing hardware event counters for the purpose of power estimation. The feasibility of this approach has been shown for commercially available desktop processors [1, 2] as well as for embedded processors [3].

Especially for enabling dynamic power management in mobile embedded systems, dedicated energy and power estimating coprocessors have been introduced [4, 5]. For estimating the power consumption, event-based energy and power macromodels are utilized in these works.

By augmenting the HDL model of a given system with dedicated power estimation hardware, *power emulation* can be performed as a by-product of functional emulation. Power emulation approaches using register transfer level (RTL) macromodels [6] and architectural-level component event-based power models [9] have been presented. In [8] a hybrid simulation- and emulation-based approach is introduced. This cosimulation approach can be used to simulate non-synthesizable parts of the overall system.

While RTL power emulation approaches achieve high estimation accuracy (mean error 3.4% as compared to RTL), they also suffer high area overhead (on average 3.1 times the area of the original design [6]). Furthermore, the additionally required logic negatively influences the maximum reachable operating frequency of the design. A high-level power emulation approach as presented in [9], circumvents this limitation due the use of a simplified power model. The estimation error in comparison to gate-level simulations is reported to be below 10% while requiring less than 3% of the FPGA LUTs. However, the power model proposed in [9] targets a specific implementation of a LEON3 chip multiprocessor (CMP) system introduced by the authors and is not a generic methodology.

In recent work, we have introduced our initial high-level power emulation hardware unit [10] as well as the use of this high-level power emulation methodology in power-aware software development [11]. Based on these works, we introduce and evaluate our automated power modeling and HDL implementation methodology in the following sections.

III. High-Level Power Emulation

The principle of power emulation, as initially established in [6], is based on augmenting the emulated system with special power estimation hardware. Power estimates are generated as a by-product of functional emulation during the run-time of the system. The traces of execution for both, functional and power emulation, can then be analyzed on a host computer as depicted in Figure 1.

A. Power Model

Our power model is based on the assumption that the power consumption of a given system and its sub-components can be expressed by *power states* at given points in time [12]. Furthermore we assume that for a given system-under-test, interface as well as internal signals down to a given hierarchical level are observable. This assumption is valid due to the fact that the HDL model of the system is required for synthesizing and mapping it onto the FPGA for functional emulation in the first place. Based on these assumptions, our power macromodel tries to map a number N of state- and power-relevant signals x_i to a power consumption estimate $\hat{P}$.

The total power consumption P of the system can be given as $P = P_{dyn} + P_{sta}$, where P_{dyn} resembles the well-known equation of dynamic power consumption in CMOS and P_{sta} accounts for static power consumption and leakage power. For $P_{dyn} = \alpha \cdot (f \cdot C \cdot V^2)$, α expresses the activity of a given component and the term $(f \cdot C \cdot V^2)$ captures clock frequency f, the amount of capacitance C being switched and the supply voltage V.

We model the relationship of state-dependent activity and the power consumption estimate as an additive linear equation in the form of

$$\hat{P} = c_0 + \sum_{i=i}^{N} c_i x_i = c_0 + c_1 x_1 + \ldots + c_N x_N \qquad (1)$$

where the coefficient c_0 models sources of static power consumption P_{sta} such as analog components and leakage. The coefficients c_i $(i = 1 \ldots N)$ express the non-activity-dependent term of CMOS power consumption $(f \cdot C \cdot V^2)$. The activity factor α of a given component is determined by the according state signal x_i. We can also express Equation 1 in the vector form $\hat{P} = \mathbf{x}\mathbf{c}^T$, where $\mathbf{x} = [1, x_1, \ldots, x_N]$ and $\mathbf{c} = [c_0, c_1, \ldots, c_n]$. Note that this model, while appropriately simple for implementing it in power emulation hardware, is sufficient to capture the power consumption dynamics of our system-under-test as shown in Section VI.

The problem that arises from this high-level power macromodeling approach is twofold: 1) The adequate selection of model parameters x_i, i.e., the identification of power-relevant state signals for a given system-under-test and 2) the fitting of model coefficients c_i for the selected parameters.

B. Power Emulation Unit

The power emulation (PE) unit monitors the power-relevant state signals x_i and derives cycle-accurate power estimates $\hat{P}$ from these data. It contains the hardware implementation of the power model as introduced in Equation 1. The architecture of our initial power emulation unit [10], as depicted in Figure 2, is used as the foundation for this work.

The PE unit consists of a number of power sensors, monitoring the state and activity of various system components. Each power sensor maps the observed state signals to a corresponding power value for the given module using a look-up table approach. The power estimator itself accumulates the

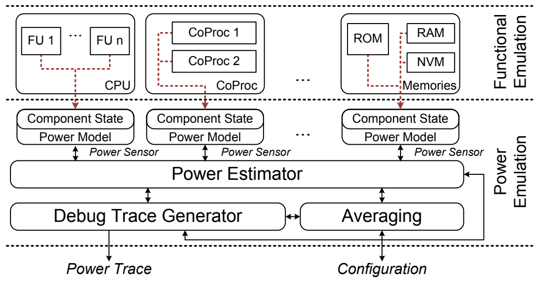


Fig. 2. Basic architecture of the power emulation unit monitoring system components and deriving power estimates according to their states (adapted from [10])

values generated by all power sensors and calculates the cycle-accurate overall power consumption estimate $\hat{P}$ of the system.

The power estimates are collected by a debug trace generation unit that assembles trace messages and delivers them to a host computer. Additionally, a reconfigurable moving average filtering module, serving the purpose of smoothing and denoising of power traces, is present in the PE unit.

While the initial power emulation unit presented in [10] resembles the manually created prototype used as the proof-of-concept for our high-level power emulation concept, in this paper[1] we present an automation approach that covers the two main obstacles for the power emulation unit's hardware implementation: 1) The automated adaptation of the system-under-test's existing HDL model for the purpose of power emulation. Internal power-relevant state signals, originating at various levels of design hierarchy, have to be connected to the power emulation unit. 2) The automated adaptation of the power emulation unit itself to the automatically generated power model.

IV. Automated Power Characterization

The manual characterization and power modeling of a given system is a time-consuming and tedious task. Therefore we employ an automated characterization methodology, consisting of multiple stages as depicted in Figure 3.

Based on a set of microbenchmarks, covering all components of the system-under-test, state-of-the-art gate-level power estimation tools are used to estimate power profiles of the system's physical implementation. At later stages of design, i.e., after the first silicon implementation is available, additional measurement profiles can be used for a *joint simulation- and measurement-based characterization*. The next step in creating a power model is the automated *selection of model parameters*. Finally, for the parameters selected before, coefficients are determined by means of a *model coefficients fitting* process.

A. Joint Simulation- & Measurement-based Characterization

For the purpose of characterizing the system-under-test and constructing a power macromodel, training set data are required [13]. The training set data tuple $\mathbf{T} = (\mathbf{X}, \mathbf{p})$, consisting of an observed signal state matrix (SSM) $\mathbf{X}$ and an observed power vector $\mathbf{p}$, are used to construct the power model. By executing m microbenchmarking applications in RTL or gate-level simulations, we obtain the signal state matrix $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_m]^T$. $\mathbf{X}_i = [\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,K}]^T$ is the signal state matrix for a given K-cycle benchmark, containing a signal state vector $\mathbf{x}_{i,k}$ for every clock cycle k. The power vector $\mathbf{p} = [\mathbf{p}_1, \ldots, \mathbf{p}_m]^T$, where similar to above for a k-cycle benchmark $\mathbf{p}_i = [P_{i,1}, \ldots, P_{i,k}]^T$, is typically generated either in gate-level power simulations or, during later design stages, through physical power consumption measurements. In our characterization methodology we seek to improve training set quality by offering the possibility to integrate physical measurements performed on the same microbenchmarks that are used in gate-level simulations.

We address the issue of merging simulation- and measurement-derived training sets in a twofold manner: First, marker operations that serve the goal of detecting beginning and end of a test are inserted in each microbenchmark. Second, we perform a cross-correlation between the two resulting power profiles to compensate their temporal mismatch due to the non-ideal triggering conditions of the physical measurement. Afterwards, the simulated profile as well as the delay-compensated measured profile are stored in a joint measurement database that is used in the subsequent model parameter selection and coefficient fitting steps.

B. Model Parameter Selection

The automated selection of adequate model parameters represents one of the key contributions of our work. Specifically we try to minimze the number N of used power model parameters so that $N \ll N_{all}$, i.e., the number of selected parameters is by far smaller than the number of all available parameters. A small number of parameters, and thus a small number of coefficients to be fitted, vastly reduces model fitting effort and improves model efficiency.

The power model parameter selection algorithm is composed of three main sub-algorithms, as described in Algorithm 1, that are performed for the entire set of microbenchmarks. In a first *signal name pattern matching* step, the large number

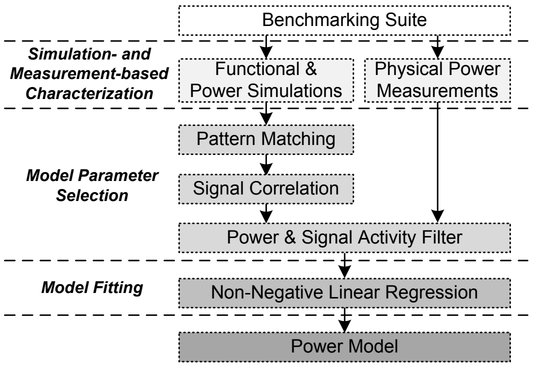

Benchmarking Suite
Simulation- and Measurement-based Characterization
Functional & Power Simulations
Physical Power Measurements
Pattern Matching
Model Parameter Selection
Signal Correlation
Power & Signal Activity Filter
Model Fitting
Non-Negative Linear Regression
Power Model

Fig. 3. Overview of the automated power characterization methodology

[1]This work is part of the POWERHOUSE project that is funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 815193.

Algorithm 1: Power model parameter selection

```
Input: List of benchmarks $L_{BM}$, List of signals $L_S$,
       Training set data tuple $\mathbf{T}$ (containing transient
       power profiles for all benchmarks $\mathbf{p}$ and signal
       activity data for all benchmarks $\mathbf{X}$), Signal name
       pattern list $L_{npat}$, Intra-signal correlation
       threshold $Th_{cor}$, Power activity threshold $Th_{pcor}$,
       Lag threshold $Th_{lag}$
Output: List of power model parameters $L_{pms}$
Step 1, signal name pattern matching:
List of candidate model parameters $L_{pms} := \{\}$
List of candidate signals $L_{cs} := \{\}$
foreach Signal $x_i$ in $L_S$ do
    if Name of $x_i \in$ name pattern list $L_{npat}$ then
        $L_{cs} := L_{cs} \cup \mathbf{x}_i$
foreach Benchmark b in $L_{BM}$ do
    List of candidate signals per benchmark $L_c := L_{cs}$
    Step 2, intra-signal correlation:
    foreach Signal $x_i$ in $L_c$ do
        if Signal activity $\alpha(x_i)$ = 0 then
            Remove signal $\mathbf{x}_i$ from $L_c$, continue;
        $\mathbf{R}_{\mathbf{X}_i} = \text{CorrCoef}(\mathbf{x}_i, \text{SSM for benchmark } \mathbf{X}_b)$
        if Elements in $|\boldsymbol{R}_{\boldsymbol{x}_i}| = 1$ then
            Remove same-cycle correlated signals from
            $L_c$ except lowest hierarchical level one
        foreach Signal $x_j$ in $L_c$, $i \neq j$ do
            if $|r_{(\boldsymbol{x}_i,\boldsymbol{x}_j)}(\tau)| > Th_{cor}$ for $0 \leq \tau \leq Th_{lag}$
            then
                Remove delayed, correlated signals from
                $L_c$ except lowest hierarchical level one
    Step 3, power - signal activity:
    foreach Signal $x_i$ in $L_c$ do
        if $|r_{(\Delta\boldsymbol{p},\Delta\boldsymbol{x}_i)}(\tau)| < Th_{pcorr}$ for $0 \leq \tau \leq Th_{lag}$
        then
            Remove not power-related signals from $L_c$
    $L_{pms} := L_{pms} \cup L_c$
```

of potential model parameter signals x_i is largely reduced by only selecting signals matching a user-supplied list of certain name patterns. This list is typically dependent on the coding standard employed while creating the HDL implementation of the given system-under-test. A default name pattern list includes potential candidate signal name patterns, such as "enable", "read", "write", "busy", "ready", "wait", "halt", "sleep", etc., that are common to a large number of HDL designs. Note that by adapting this list of name patterns to the coding standard used while implementing the given system-under-test, the filtering result can be further narrowed down, reducing the run-time of the subsequent parameter selection algorithms.

The *intra-signal correlation analysis* represents the second sub-algorithm. It aims at analyzing signal activity statistics and removing redundant signals. The following main tasks are performed: 1) Removal of signals stuck at one logical level for the entire set of benchmarks, i.e., signals lacking any switching behavior. Note that a power benchmarking suite will also include tests varying otherwise mainly static signals, such as software configuration bits dictating the operating modes of the design. Thus, these mainly static but power-relevant configuration bits will not be eliminated by the intra-signal correlation analysis.

2) Calculation of the correlation coefficients matrix $\mathbf{R}$ for all N remaining signals $\mathbf{x}$ given as

$$\mathbf{R}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{C}(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{\mathbf{C}(\mathbf{x}_i, \mathbf{x}_i)\mathbf{C}(\mathbf{x}_j, \mathbf{x}_j)}}, \quad 1 \leq i, j \leq N \qquad (2)$$

where $\mathbf{C}$ is the covariance matrix. For each row of the matrix $\mathbf{R}$ all columns containing high absolute correlation coefficient values resemble redundant signals, switching at similar points in time. A negative correlation coefficient indicates an inverted signal. These correlated signals are filtered out, only the signal originating at the highest hierarchical level is retained[2]. 3) Calculation of the cross-correlation $r_{(\mathbf{x}_i,\mathbf{x}_j)}(\tau)$ of all the remaining signals within a specified maximum lag of Th_{lag}:

$$r_{(\mathbf{x}_i,\mathbf{x}_j)}(\tau) = \sum_{k=0}^{K-1} \mathbf{x}_i(k) \cdot \mathbf{x}_j(k+\tau), \quad 0 \leq \tau \leq Th_{lag} \qquad (3)$$

Signals exhibiting a high maximum correlation within the maximum lag Th_{lag}, e.g., typically a small number of clock cycles, are also considered to be redundant. These redundant and delayed signals are filtered out as well.

The third sub-algorithm performs the *power - signal activity cross-correlation analysis* of the remaining signals. We calculate the cross-correlation $r_{(\Delta\mathbf{p},\Delta\mathbf{x}_i)}(\tau)$ for the forward differences $\Delta f(x) = f(x+1) - f(x)$ of both the remaining signals $\mathbf{x}_i$ and the transient power profile $\mathbf{p}$ as expressed in Equation 4.

$$r_{(\Delta\mathbf{p},\Delta\mathbf{x}_i)}(\tau) = \sum_{k=0}^{K-1} \Delta\mathbf{p}(k) \cdot \Delta\mathbf{x}_i(k+\tau), \quad 0 \leq \tau \leq Th_{lag} \qquad (4)$$

This can be interpreted as a measure for the relation of signal state changes to changes in the transient power consumption of the system. Note that this is a measure similar to *power sensitivity* as introduced in [14]. All signals exhibiting low correlation within a certain lag Th_{lag}, e.g., typically a small number of clock cycles, are not considered power-relevant and are removed.

The list of candidate model parameters L_c now only contains power-related and non-redundant signals. While iterating over the whole set of available microbenchmarks, the list of

[2]Keeping the signal originating at the highest level of design hierarchy is in general beneficial for the automated HDL model integration as it reduces the amount of hierarchical levels - and therefore typically also the amount of HDL files - that have to be modified (see Section V-A).

potential power model parameters L_{pms} is compiled as the concatenation of all candidate parameter lists L_c. This list can then be used in the model coefficient fitting process.

C. Model Coefficient Fitting

Based on the power model parameters list L_{pms} compiled by the parameter selection algorithm and containing N_{pms} parameters, the complete signal state matrix $\mathbf{X}$ of size $K \times N_{all}$ can be reduced to $\mathbf{X}_{pms}$ of size $K \times N_{pms}$ by eliminating all non-used parameters. The reduced training set data tuple is then $\mathbf{T}_{pms} = (\mathbf{X}_{pms}, \mathbf{p})$, with the either simulated or measured power vector $\mathbf{p}$ as introduced above.

Determining the vector of fitting coefficients $\mathbf{c}$ requires solving the typically heavily overdetermined equation system $\mathbf{p} = \mathbf{X}_{pms}\mathbf{c}$. To this end, we employ Lawson's well known linear least squares regression technique with non-negativity constraint (*Nonnegative Least Squares*, NNLS) [15]. This algorithm aims at minimizing $||\mathbf{X}_{pms}\mathbf{c} - \mathbf{p}||$ so that $\mathbf{c} \geq 0$. The non-negativity constraint on the power model coefficients helps reducing power emulation hardware complexity while increasing model robustness.

V. Automated HDL Model Implementation

The automated power characterization, parameter selection and model fitting methods as introduced in Section IV allow for the rapid power modeling of a given system-under-test for the purpose of enabling power emulation. The second obstacle in this scope is constituted by the time-consuming and tedious task of implementing this automatically devised power model in hardware and integrating it into the existing HDL model of the system-under-test.

We address this obstacle in a twofold manner as depicted in Figure 4: 1) The *automated adaptation of the existing HDL model of the system-under-test*, allowing for the monitoring of internal power-relevant state signals by the power emulation unit, i.e., the "routing" of power-relevant state signals originating in different design entities at various levels of hierarchy. 2) The *automated power emulation hardware generation*, i.e., the HDL implementation of the devised power model.

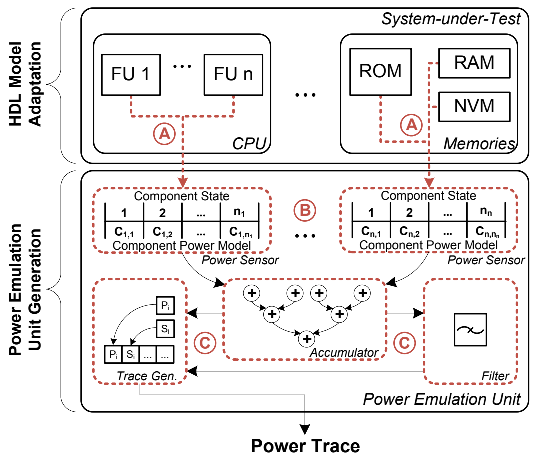


Fig. 4. Overview of the automated HDL model implementation, dotted lines indicate automatic modifications: (A) "Routing" of state signals in HDL model of system-under-test, (B) number and structure of power sensors according to the used power model and (C) internal structure of the power emulation unit

A. HDL Model Adaptation

One of the remaining challenges for automatically enabling the power emulation of the system-under-test is the vital task of adding the additional connections of power-relevant signals, originating at various hierarchical levels of the design, to the power sensors as illustrated in Figure 4. For this purpose we have implemented an algorithm that utilizes the VMAGIC (VHDL manipulation and generation interface) Java library presented in [16]. VMAGIC allows for VHDL code analysis, manipulation and generation. This library itself builds upon the ANTLR 3.1 parser generator [17]. Our adaptation algorithm consists of the following steps as illustrated in Figure 5:

- VHDL parsing: Parsing of all files associated with the system's VHDL model.
- Dependency extraction: Creation of a tree-representation of the dependencies and hierarchical levels of all design entities.
- Signal routing: Routing of all signals specified in the power model to the according power sensor. This step includes the insertion of additional intermediate signals as well as the inherent modification of the interfaces to include these signals.
- VHDL modification and write-back: Generation of synthesizable VHDL code of the modified components.

B. Power Emulation Unit Generation

After adapting the system-under-test's HDL model for power emulation, the final step is integrating the HDL model of the power emulation unit itself with the system-under-test in a common design that can be synthesized and downloaded to the FPGA platform. For this purpose, the HDL model of the PE architecture is split into several HDL template files, containing a number of tags that can be easily parsed and replaced by an adaptation algorithm. Essentially, this algorithm is adapting the following components and properties of the power emulation unit (see Figure 4):

- The number of *power sensors* as well as their internal structure, i.e., the number of power tables and their according bit-widths according to the power model and the desired resolution of the power estimates.
- The power estimates *accumulation unit*, i.e., essentially the width and height of the implemented adder tree.
- The length and bit-width of the *filtering unit*.

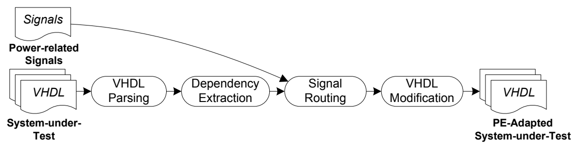


Fig. 5. HDL model adaptation algorithm

- The debug *trace generator* bit-widths according to the accumulation and filtering unit.

VI. Experimental Results

We have evaluated our automated power characterization and HDL model implementation methodology on a smart card microcontroller test system supplied by our industrial partner. This system has been further extended for enabling power emulation in an industrial setting. For different power models, with parameters selected either manually or automatically, we have evaluated the characterization effort as well as the accuracy and the hardware implementation effort.

A. Test System and Used Power Models

A smart card microcontroller based on a 16-bit pipelined cache architecture, composed of volatile and non-volatile memories as well as a number of peripherals, e.g., cryptographic coprocessors, UARTs, timers and random number generators, is used as our system-under-test (see Figure 6). This system has been extended with the power emulation unit as shown in Section V. What makes the system particularly interesting in terms of power characterization is the fact that due to its typical operating environment particular care has been taken to achieve a power-optimized design. Therefore, dedicated power-aware system states (such as low-power halt modes of certain components), also have to be considered by the characterization process.

For illustrating the benefits of using the automated power characterization process, we have compared various power models in terms of accuracy and characterization effort. To this end, four different power models have been benchmarked: 1) A *manual model*, i.e., the parameters were selected manually, devised for the first implementation of our power emulation unit. 2) A naive *brute force* approach - based power model that was created by performing linear regression separately for each microbenchmark using the entire large set of signals found by the *signal name pattern matching* algorithm. All parameters assigned significant coefficient values in this process were considered candidate model parameters. 3) A

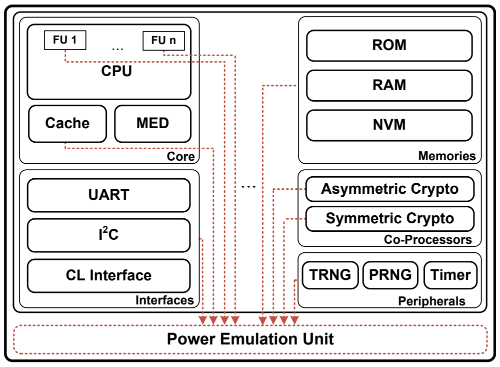


Fig. 6. 16-bit smart card microcontroller test system augmented by power emulation unit and internal state-signal connections

power model only using parameters retained after the *intra-signal correlation* filter. Finally, 4) a model employing only parameters retained after the *intra-signal correlation and the power-signal activity* filter. Note that for all these models coefficient fitting was performed using the same technique as described in Section IV-C.

B. Comparison of Accuracy

We have evaluated the accuracy of the power emulation results employing these power models against gate-level power simulations using *Synopsys Primetime PX V2008.12 SP3* [18] on the basis of a cycle-by-cycle comparison. This evaluation was performed on the set of microbenchmarks used for the characterization process, i.e, the training set (TS) as well as for a different set of control benchmarks, i.e, the control set (CS). For this control set of benchmarks we have used general purpose embedded benchmarks from the MiBench suite [19] as well as Dhrystone. Due to the application-specific nature of our test system, we have employed a custom benchmarking application utilizing system components such as the cryptographic co-processors and other peripherals that are not covered by standard CPU benchmarks.

Figure 7 illustrates the average estimation error for these benchmarks. Figure 8 further summarizes these numbers

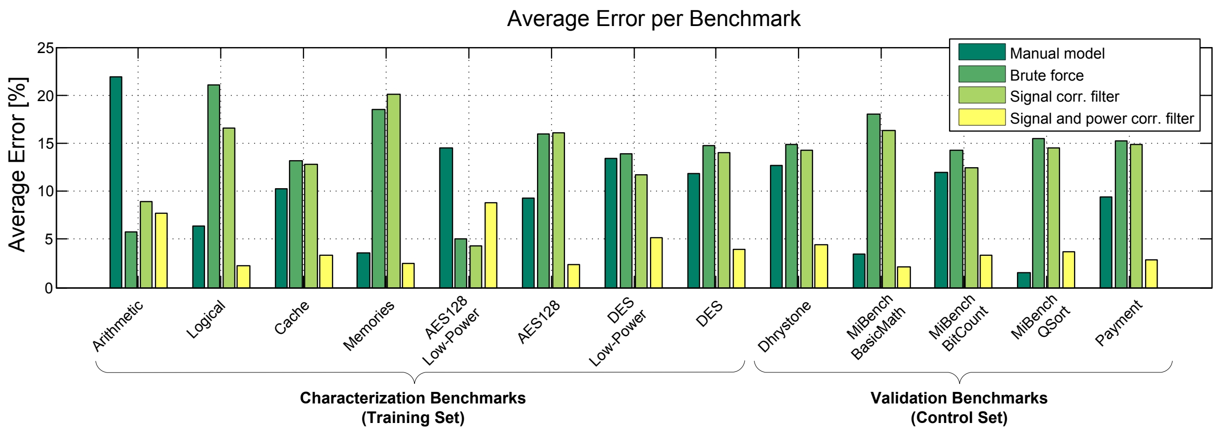


Fig. 7. Comparison of average estimation error for various benchmarks

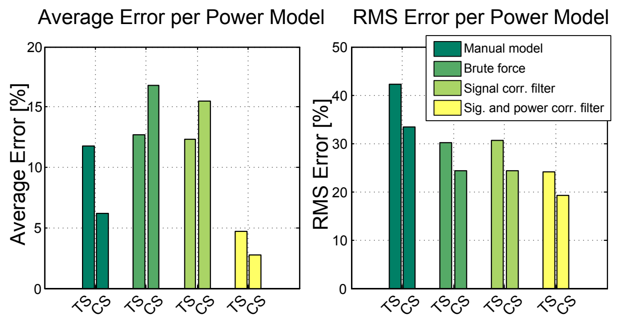


Fig. 8. Comparison of average and RMS estimation error for different power models over all benchmarks from the training set (TS) and the control set (CS)

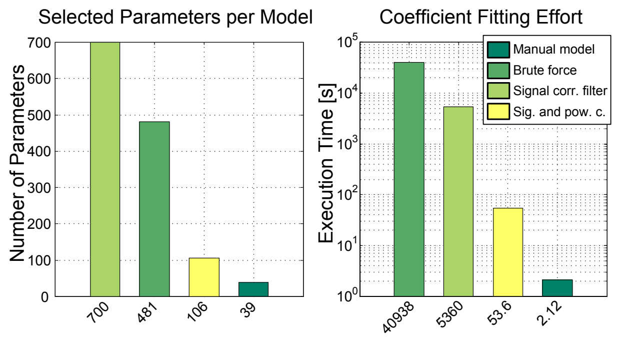


Fig. 9. Comparison of number of coefficients and required execution time for coefficient fitting (Note semilogarithmic scale for second plot)

across all benchmarks and additionally illustrates root-mean-square error (RMSE) values used as a measure for the cycle-by-cycle accuracy. It can be seen that the power model created by the intra-signal correlation and the power-signal activity algorithm reduces the average error as compared to the manual model from 11.78% to 4.71% on the training set and from 6.18% to 2.78% on the control set. The same is valid for the RMSE that is brought down from 42.24% to 24.04% (TS) and from 33.46% to 19.07% (CS). Furthermore, this approach also outperforms the brute force and the stand-alone intra-signal correlation approaches with regard to the same metrics. The relatively high RMSE at this point is due to fact that we are utilizing a *single* power model representing the entire test system in order to decrease the impact of power emulation onto the FPGA. We expect to further decrease this number by using a model splitting approach that models on-chip co-processors and memories seperately.

C. Power Modeling and HDL Adaptation Effort

Next, we compare the effort required for selecting the parameters of the models compared above, for performing the linear regression based coefficient fitting and for the required HDL adaptation. Note that all methods operate on the same training set data. Hence, this portion of the total time required for characterization is constant for these models and is not reflected in Table I.

Due to the different nature of the used parameter selection methods, the number of obtained parameters varies (see Figure 9). Likewise, the execution time for performing linear regression - based coefficient fitting varies with the number of coefficients. Note, however, that the execution time is not only dependent on the number of parameters but also on the conditioning of the input data. Even though the number of parameters determined by the signal correlation filter and the naive brute force approach is by far larger than for the combination of signal and power correlation filter, these models fail to deliver the same level of accuracy. Another interesting observation is that the percentage of remaining non-zero coefficients after performing the NNLS fitting algorithm is the highest for the signal and power correlation filter method as shown in Table II. When considering the lower model parameter selection and coefficient fitting efforts (Figure 9), the automated parameter selection process is even more favorable.

To give an understanding of the required efforts for the automatic HDL implementation as introduced in Section V, we have compared the efforts for adapting the HDL model of this particular system-under-test for the automatically generated

TABLE I
COMPARISON OF PARAMETER SELECTION EFFORT FOR DIFFERENT POWER MODELS

Parameter Selection Method	Effort / Execution Time
Manual selection[a]	several days
Brute force[b,c]	∼ 12 days
Signal corr. filter[c]	8.2 min
Signal and power corr. filter[c]	8.3 min

[a]Iterative parameter selection process by the power emulation unit HW designer.
[b]Sequential linear regression for all benchmarks, candidate parameters from *signal name pattern matching*.
[c]Executed on a 3 GHz AMD Opteron system.

TABLE II
COMPARISON OF SELECTED PARAMETERS AND REMAINING NON-ZERO COEFFICIENT AFTER NNLS FOR DIFFERENT POWER MODELS

Power Model	# Sel. Param.	# Rem. Coeff.	%
Manual model	39	15	38.5
Brute force	481	196	40.7
Signal corr.	700	186	26.6
Sig. & pow. corr.	106	54	50.9

TABLE III
COMPARISON OF HDL IMPLEMENTATION EFFORT

HDL Implementation Method	Effort / Execution Time
HDL analysis[a]	28.9 s
Dependency extraction[a]	4 s
Signal routing & Write-back[a]	24.3 s
PE unit HDL generation[a]	0.2 s
Automatic implementation total[a]	57.4 s
Manual implementation	several hours

[a]Executed on a 3.2 GHz Intel Xeon system.

TABLE IV
COMPARISON OF NUMBER OF COEFFICIENTS AND IMPACT ON EMULATION PLATFORM FOR DIFFERENT POWER MODELS

Power Emulation	Coefficients[a]	Utilization[b]
Manual model	15	0.8%
Brute force	196	4.2%
Signal corr. filter	186	4.1%
Signal and power corr. filter	54	1.6%
Functional Emulation	*n.a.*	66%

[a]Remaining non-zero coefficients after performing linear regression.
[b]Percentage of total available ALUTs on Altera Stratix II platform.

model in Table III. The table summarizes the execution times for adapting the HDL model comprising 400 files in total, $\sim$10% thereof are being modified. Note that the amount of manual effort will certainly vary largely with the designer's expertise.

D. Impact on Emulation Platform

The power emulated test system was implemented on an Altera Stratix II platform. The microcontroller itself, as used for functional emulation only, employs approximately 66% of the platform's available adaptive look-up tables (ALUTs). Table IV lists the additionally required ALUT percentage for implementing the respective power model.

The power model automatically generated using the signal and profile correlation filter requires additional 1.6% of the platform's ALUTs as opposed to the 0.8% of the manual model. Considering the overall increase in accuracy, we consider the impact of the larger power emulation hardware on the emulation platform as minor. Due to this minor impact, the system can be operated at its targeted clocking frequency of 33 MHz. Note that this relatively low clocking frequency resembles a requirement of the test system and not a limitation of the power emulation unit.

VII. CONCLUSIONS

With the increasing complexity of integrated circuits, power simulations are becoming infeasible due to extensive simulation times. Hardware-accelerated power emulation approaches promise to be a solution to this issue. However, only little attention has been awarded so far to the problem of devising a generic methodology capable of automatically enabling the power emulation of a given system-under-test.

In this paper we have outlined a method for the automatic characterization and power model creation of a given system-under-test for the purpose of power emulation. Using this method a power model has been automatically established for a smart card controller test-system that reduces the estimation error from 11.78% to 4.71% as compared to a manually derived one. Furthermore, we have illustrated how the automatic hardware integration of this power model and the required HDL model adaptation can be achieved. Both, the automated power model creation as well as its automated hardware integration, drastically decrease the overall effort for enabling the power emulation of a given system-under-test. We believe that this reduction of effort will allow the power emulation technique to prove its benefits also in industrial settings.

VIII. ACKNOWLEDGEMENTS

We would like to thank our industrial partners Infineon Technologies Austria AG and Austria Card GmbH for their enduring support in the course of this project. Furthermore, we would like to thank the Austrian Federal Ministry for Transport, Innovation, and Technology for providing us with funding for the POWERHOUSE project under the FIT-IT contract FFG 815193.

REFERENCES

[1] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *Proc. of the 9th ACM SIGOPS European workshop*, 2000.
[2] R. Joseph and M. Martonosi, "Run-time power estimation in high performance microprocessors," in *Proc. ISLPED*, 2001.
[3] G. Contreras and M. Martonosi, "Power prediction for intel XScale processors using performance monitoring unit events," in *Proc. of the ISLPED*, 2005.
[4] J. Haid, G. Kaefer, C. Steger, and R. Weiss, "A co-processor for real-time energy estimation of system-on-a-chip," in *Proc. of the 45th MWSCAS*, 2002.
[5] J. Peddersen and S. Parameswaran, "Low-impact processor for dynamic runtime power management," *Design & Test of Computers, IEEE*, vol. 25, pp. 52–62, 2008.
[6] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: a new paradigm for power estimation," in *Proc. 42nd DAC*, 2005.
[7] D. Atienza, P. G. Del Valle, G. Paci, F. Poletti, L. Benini, G. De Micheli, and J. M. Mendias, "A fast HW/SW FPGA-based thermal emulation framework for multi-processor system-on-chip," in *Proc. 43rd DAC*, 2006.
[8] M. Ghodrat, K. Lahiri, and A. Raghunathan, "Accelerating system-on-chip power analysis using hybrid power estimation," in *Proc. of the 44th DAC*, 2007.
[9] A. Bhattacharjee, G. Contreras, and M. Martonosi, "Full-system chip multiprocessor power evaluations using FPGA-based emulation," in *Proc. of the ISLPED*, 2008.
[10] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss, "An emulation-based real-time power profiling unit for embedded software," in *Systems, Architectures, Modeling, and Simulation (SAMOS)*, 2009.
[11] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid, "Accelerating embedded software power profiling using run-time power emulation," in *19th PATMOS*, 2009.
[12] L. Benini, R. Hodgson, and P. Siegel, "System-level power estimation and optimization," in *Proc. of the ISLPED*, 1998.
[13] A. Raghunathan, N. Jha, and J. Dey, *High-Level Power Analysis and Optimization*. Kluwer Academic, 1998, ch. 3, p. 43.
[14] Z. Chen, K. Roy, and T.-L. Chou, "Power sensitivity—a new method to estimate power dissipation considering uncertain specifications of primary inputs," in *Proc. of the ICCAD*, 1997.
[15] C. Lawson and R. Hanson, *Solving Least Squares Problems*. Prentice-Hall, 1974, ch. 23, p. 161.
[16] C. Pohl, C. Paiz, and M. Porrmann, "vMAGIC - Automatic code generation for VHDL," *International Journal of Reconfigurable Computing*, vol. 2009, 2009.
[17] http://www.antlr.org/.
[18] http://www.synopsys.com/.
[19] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *IEEE 4th Annual Workshop on Workload Characterization*, 2001.

A Hardware-Accelerated Estimation-Based Power Profiling Unit - Enabling Early Power-Aware Embedded Software Design and On-Chip Power Management

Andreas Genser[1], Christian Bachmann[1], Christian Steger[1], Reinhold Weiss[1], and Josef Haid[2]

[1]Institute for Technical Informatics, Graz University of Technology, Inffeldgasse 16/I, 8010 Graz, Austria
[2]Infineon Technologies Austria AG, Design Center Graz, Austria
`{andreas.genser,christian.bachmann,steger,rweiss}@tugraz.at`
`josef.haid@infineon.com`
`http://www.iti.tugraz.at`
`http://www.infineon.com`

Abstract. The power consumption of battery powered and energy scavenging devices has become a major design metric for embedded systems. Increasingly complex software applications as well as rising demands in operating times, while having restricted power budgets are main drivers of power-aware system design as well as power management techniques. Within this work, a hardware-accelerated estimation-based power profiling unit delivering real-time power information has been developed. Power consumption feedback to the designer allows for real-time power analysis of embedded systems. Power saving potential as well as power-critical events can be identified in much less time compared to power simulations. Hence, the designer can take countermeasures already at early design stages, which enhances development efficiency and decreases time-to-market. Moreover, this work forms the basis for estimation-based on-chip power management by leveraging the power information for adoptions on system frequency and supply voltage in order to enhance the power efficiency of embedded systems. Power estimation accuracies achieved for a deep sub-micron smart-card controller are above 90% compared to gate-level simulations.

1 Introduction

Rising complexity of embedded software applications and the advance in processing power available in embedded systems require power analysis techniques to identify power saving potential. Furthermore, the system stability of energy scavenging devices (e.g., contact-less smart-cards) may suffer from power critical events, such as power peaks, hence their detection and prevention is of great importance in order to ensure reliable system operation.

Among all abstraction layers the greatest power saving potential can be identified on the application layer [1]. To enable the design of power-efficient software applications, power consumption feedback to the software designer should be available already at early design stages. However, commercially available power estimation and analysis tools are often operating on low abstraction layers, which are usually not available to the software designer. Low-level power simulations lead to extensive run-times, which make power evaluations for complex designs unfeasible. Consequently, high abstraction level power information at early design stages is highly desirable from a software designer's point of view. Alongside the ability to identify power saving potential, it also offers the early detection of potential power bugs.

Functional hardware emulation by means of prototyping platforms, such as FPGA-boards, has become a widespread technique for functional verification. Power information, however, is still in many cases gathered by power simulators. In this work, which is part of the PowerHouse[3] project, we propose a hardware-accelerated estimation-based real-time power profiling approach to circumvent this limitation. A given design augmented with power estimation hardware allows for obtaining power alongside functional characteristics in real-time. Power saving potential or power peaks can hence be detected earlier in the design cycle, which normally is not feasible before the design is available in silicon and actual physical measurements can be carried out.

By coupling this approach with a software development environment, valuable power information can be transferred to the software designer. This concept is depicted in Fig. 1. The FPGA-platform collects functional verification and power characteristics information, which are transmitted to a host computer. These information can be evaluated and visualized in a software development environment.

The proposed hardware-accelerated estimation-based power profiling unit forms also the basis for estimation-based on-chip power management. By providing instantaneous power information, system frequency as well as supply voltage levels are adapted by the system autonomously to present power availability conditions. Particularly in the contact-less smart-card domain, power management techniques to smoothen the power profile can help to minimize power peaks and to prevent power bugs.

This paper is structured as follows. Section 2 provides information on previous work. Section 3 briefly shows our research contributions. In Section 4 the design of the real-time power profiling unit is discussed. Section 5 outlines a case-study applying the concepts developed in this work to a contact-less deep sub-micron smart-card controller, while Section 6 covers the concept of estimation-based power management on the basis of the proposed power profiling unit. Finally, conclusions drawn from the current work and future activities are summarized in Section 7.

[3] Project partners are Infineon Technologies Austria AG, Austria Card GmbH and TU Graz. The project is funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 815193.

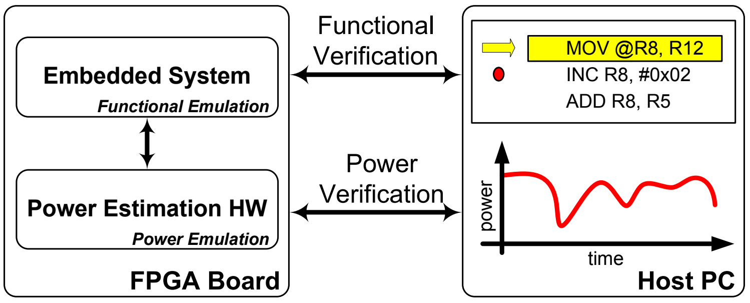


Fig. 1. Overview of the estimation-based power profiling approach for power-aware embedded software design comprising host computer interaction and visualization [2]

2 Related Work

Power profiling for embedded software can be categorized in (i) *measurement-based* and (ii) *estimation-based* methods.

Measurement-based methods are performed by taking actual physical measurements. This yields high accuracy compared to other approaches but requires additional measurement-equipment.

In contrast, power profiling by means of *estimation methods* is often based on power modeling. These techniques are usually less accurate but provide greater flexibility, since also power consumption for sub-modules of the system can be derived. In the following we compare ongoing research activities in the field of power profiling.

2.1 Measurement-Based Methods

In [3], PowerScope an energy profiling tool for mobile applications is introduced. The system's current consumption is automatically measured during run-time by a digital multimeter. Measurement data are collected for later analysis on a host computer.

An oscilloscope measurement-based profiling technique is proposed by Texas Instruments in [4]. The current drawn by a DSP system is profiled and results are visualized on a host computer in TI's software development environment.

2.2 Estimation-Based Methods

Power profiling by means of estimation techniques can be subdivided into (i) *simulation-based* and (ii) *hardware-accelerated* approaches.

Simulation-based power estimation executes programs on simulators to obtain circuit activity information. Power values are acquired based on these information. In hardware-accelerated power estimation approaches, power information is derived from power models, which are implemented in hardware.

Estimation techniques can be employed on various levels of abstraction resulting in different estimation accuracies. Moreover, the degree of abstraction influences simulation times for simulation-based approaches and hardware-effort for hardware-accelerated methods. Real-time power estimation, however, is limited to hardware-accelerated estimation techniques that operate on a high abstraction level. Commercially available power estimation tools (e.g. [5]) operate on low abstraction levels, such as gate- or register-transfer level (RTL). Achievable estimation accuracies are high, while extensive simulation times render power estimation of elaborate applications unfeasible. On top of this, low-level simulators are often not available to software designers. Therefore, attempts to estimate the system's power consumption on a higher level of abstraction are carried out.

A *simulation-based* approach employing power models for instruction-level power estimations is proposed by Tiwari et al. in [6]. It allows for power and energy consumption estimation for given applications. The underlying power model considers the power consumption during instruction execution (i.e., base costs) and power consumption during the transition between instructions (i.e., circuit state overhead costs). In [7], Sami et al. consider additional microarchitectural effects to enhance the accuracy of instruction-level power estimation based on a pipeline-aware power model for very long instruction word (VLIW) architectures. A co-simulation based power estimation technique is introduced by Lajolo et al. in [8]. This approach for system on chips (SoCs) works on multiple abstraction levels. In principle, power estimation is performed on system level, while for refinement purposes and accuracy enhancements various components are co-simulated on lower levels of abstraction. Countermeasures against high simulation times are caching, statistical sampling and macro-modeling. A simulation framework for system-level SoC power estimation is introduced by Lee et al. in [9]. This approach is based on power models developed for the processor, memories and custom IP blocks. Power values derived are provided cycle-accurately to the designer in a dedicated profile-viewer. Ahuja et al. leverage a probabilistic RTL power estimation approach on system-level. To the expense of an accuracy loss of 3-9% they achieve simulation speedups of up to 12x compared to ordinary RTL power estimations [10].

Hardware-accelerated power estimation techniques are performed by augmenting the given system with existing or dedicated power estimation hardware. A power characterization process performed beforehand determines power values, which are mapped towards corresponding power states. For example, hardware events (e.g. CPU idle/run states, memory read/write states, etc.) are representatives of such power states. Available power estimation hardware can be extended to power state counters for energy consumption accounting. In [11], Bellosa gathers information by means of hardware event counters to derive thread-specific energy information for operating systems. Joseph et al. obtain the power consumption of a system by exploiting existing hardware performance counters of a microcontroller [12]. Microprocessor performance counters are utilized for system-wide power estimations by Bircher et al. in [13]. A power macro-model based coprocessor approach for energy accounting is proposed in [14]. Energy events identified by energy sensors are tracked by a central controller. In general, the additional power estimation hardware requires extra chip area but yields also a speed-up compared to simulation-based approaches.

Power emulation represents a special case of hardware-accelerated power estimation. FPGA-boards can be used as a typical prototyping platform to emulate not only the functional system behavior but also its power consumption. A given system comprising power estimation hardware is mapped onto an FPGA-platform. Functional verification and power estimation can be performed in real-time even before the silicon implementation of the system is available.

Coburn et al. presented an overview of the power emulation principle in [15]. Run-time improvements by power estimation hardware-acceleration of about 10x to 500x over commercial power estimation tools are achieved. Strategies to minimize the hardware overhead introduced by power estimation hardware are proposed. In [16], this approach is extended to a hybrid power estimation methodology for complex SoCs. This framework combines simulation and emulation techniques, which significantly reduce power analysis times. In [17], the power consumption of processor cores is estimated employing power emulation to guide process migration between cores.

3 Contributions

Power profiling by means of physical measurements is typically very coarse-grained and limited to the entire chip due to chip integration and packaging. Moreover, the final chip is not available at early design stages.

Simulation-based power profiling techniques can be employed at the beginning of the design cycle, however they are rendered unfeasible for complex applications due to extensive simulation times. To encourage the software designer to consider power aspects at early design stages, we propose a hardware-accelerated estimation-based power profiling unit delivering power consumption information in real-time. Power information is delivered to the software designer before silicon is available by utilizing an FPGA prototyping platform comprising power estimation hardware (power emulation). Expensive redesigns caused by power bugs can be avoided, which helps to decrease time-to-market (see Fig. 2).

An automatic measure to increase system reliability is offered by on-chip power management based on our proposed hardware-accelerated estimation-based power profiling unit. The autonomous adoption of system frequency and supply voltage implies power profile smoothening that minimizes the harmfulness of power peaks. Thus, the proposed power profiling unit provides a promising approach for increasing system stability in power critical applications as well as for enhancing the system's power efficiency in general.

The main goals of this work can be defined as follows:

– Deliver power information to the designer at early design stages to allow for:
  - Power-efficient software application design
  - Power-critical event detection

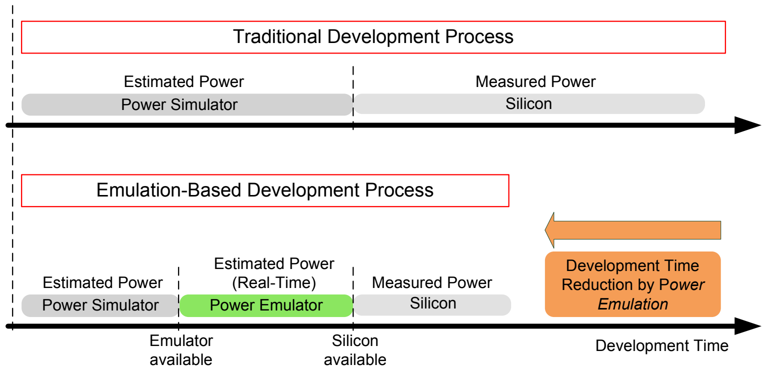


Fig. 2. Emulation-based vs. traditional power profiling approach [2]

– Enabling hardware-accelerated estimation-based on-chip power management for gaining power efficiency and to enhance the reliability of power critical embedded systems applications

4 Design of a Hardware-Accelerated Estimation-Based Power Profiling Unit

Estimation-based power profiling methods derive power information by exploiting power models. The abstraction layer on which these models are set up determines model complexity and estimation accuracy. Low-level models established on transistor- or gate-level are complex and require extensive hardware resources. Hence, they are not suitable for hardware-accelerated estimation-based power profiling. In contrast, on a higher abstraction layer only main system components (e.g. CPU, memory, coprocessor, etc.) are taken into account. This leads to more compact models, hence real-time power profiling with moderate area increases is only feasible following this approach.

4.1 Power Model

Power models on a high level of abstraction are often based on linear regression methods. Details can be obtained in [18] and implementations are discussed in [6] and [19]. A linear regression model can be defined as

$$\hat{y} = \sum_{i=0}^{n-1} c_i x_i + \epsilon. \tag{1}$$

$\mathbf{x} = [x_1, x_2, ...x_{n-1}]$ gives the vector of model parameters. x_i represent system states, such as CPU modes (e.g., idle, run) or memory accesses (e.g., read, write). The vector of model coefficients can be written as $\mathbf{c} = [c_1, c_2, ...c_{n-1}]^T$. Each model coefficient c_i contains power information and has to be determined during a power model characterization process. The linear combination of model parameters $\mathbf{x}$ and model coefficients $\mathbf{c}$ forms the power estimate $\hat{y}$. The deviation between the real power value y and its estimate $\hat{y}$ is stated by ϵ (i.e., the estimation error).

4.2 Power Characterization Process

Typically a linear regression model can be designed in three major steps.

(i) *Selection of model parameters.* The choice of model parameters directly influences the model's accuracy and is therefore of great importance. In addition, the cross-correlation between model parameters reflects the amount of redundancy in the model. This metric helps to keep a model as small as possible and thus compact in terms of hardware resources.

(ii) *Selection of the training-set.* The training-set is based on m power measurements for a number of m vectors $\mathbf{x^i}$, each of which containing n model parameters

$$\mathbf{x^i} = [x_0^i, x_1^i, ...x_{n-1}^i] \text{ for } 0 \leq i \leq m-1. \tag{2}$$

Vectors $\mathbf{x^i}$ can be combined to the matrix

$$\mathbf{X} = [\mathbf{x^0}, \mathbf{x^1}, ...\mathbf{x^{m-1}}]^\mathbf{T}. \tag{3}$$

Power values y acquired by low abstraction level power simulations (e.g., gate-level simulations) or physical measurements for corresponding vectors $\mathbf{x^i}$ can be expressed as

$$\mathbf{y} = [y_0, y_1, ...y_{m-1}]^T. \tag{4}$$

Finally, $\mathbf{X}$ and $\mathbf{y}$ define the training-set given as the tuple $\mathbf{T}$ in (5).

$$\mathbf{T} = (\mathbf{y}, \mathbf{X}) \tag{5}$$

The linear regression model given in (1) can also be written in matrix-form, which is depicted in (6).

$$\mathbf{y} = \mathbf{Xc} \tag{6}$$

Vectors $\mathbf{x^i}$ in $\mathbf{X}$ are derived from test applications (micro-benchmarks) on a given embedded system and corresponding power values $\mathbf{y}$ are determined by gate-level simulations or physical power measurements.

(iii) *Least squares fit method.* The number of elements in the training-set $\mathbf{T}$ is usually much higher than the number of model parameters $\mathbf{c}$. This implies that the number of rows in $\mathbf{y}$ and the number of columns in $\mathbf{X}$ is higher than actually required to solve the linear system of equations in (6). Hence, the system is overdetermined and no exact solution exists. To overcome this issue model parameters are determined, while minimizing the square error by using the least squares fit method.

The impact of different power model setup strategies on the estimation accuracy has been explored in our previous work [20]. Starting from (i) a *manual model*, i.e., model parameters are selected manually, accuracy enhancements could be achieved by performing (ii) a *naive brute force* method that carries out linear regression with the entire large set of model parameters of the system; (iii) an *intra-signal correlation* method that removes signals stuck at a logical level or that do not show any switching activity; and (iv) a *power-signal activity filter* that computes the cross-correlation between model parameters and the transient power profile. By means of these methods, the average estimation error could be reduced from 11.78% for the *manual model* to 4.71% for a model built upon the combination of *intra-signal correlation* and the *power-signal activity filter*.

4.3 Power Estimation Architecture

The power estimation (PE) architecture that integrates the power model in hardware is illustrated in Fig. 3.

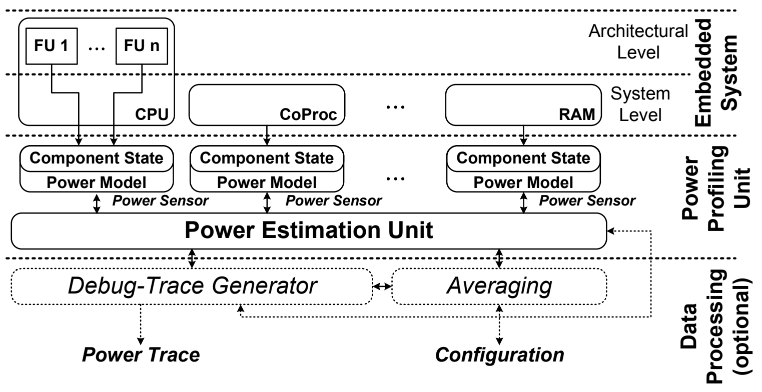


Fig. 3. Power estimation architecture, obtained and adopted from [2]

Power sensors are employed to track state information of system modules. For accuracy purposes also lower abstraction layer information can be considered (e.g., state information of functional units of the CPU). State-dependant power information are stored in a software-configurable table that is integrated in the power sensors. These state information are mapped towards power values using a table-lookup approach. Fig. 4 depicts the principle structure of a power sensor module.

Each of a number of k power sensors covers l system states and contributes to the entire power model as expressed in (7). The PE-architecture delivers power information each cycle, hence time-dependency t is introduced in the following equations to account for power values estimated at different points in time.

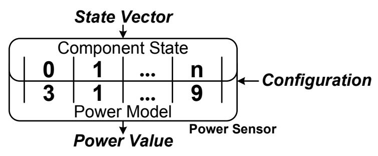


Fig. 4. Power sensor [2]

$$y_{j,i}(t) = c_i \; x_i(t) \text{ for } 0 \leq i \leq l-1 \; \wedge \; 0 \leq j \leq k-1 \tag{7}$$

16-bit registers are provided to configure the power sensors with the power coefficients information obtained from **c**. It is worth noting that this power table can also be reconfigured during program run-time. This enables the masking of system modules, allowing the tracking of the power consumption of single sub-modules.

The power estimation unit accumulates 16-bit power sensor outputs according to (8). This constitutes an instantaneous, cycle-accurate up to 32-bit wide power estimate $y(t)$ for the overall system. The entirety of power sensors comprising the power estimation unit represent the power model established in hardware (see equality in (1) and (8)).

$$y(t) = \sum_{j=0}^{k-1} \sum_{i=0}^{l-1} y_{j,i}(t) = \sum_{i=0}^{n-1} c_i \; x_i(t) \tag{8}$$

Further post-processing is applied by the averaging module, which allows for smoothing and de-noising of a sequence of power values. This is enabled by a configurable moving average filter as shown in (9). Filtering properties can be changed by adjusting N.

$$y_{avg} = \frac{1}{N} \sum_{j=0}^{N-1} y(t-j) \tag{9}$$

Finally, the debug-trace generator module assembles trace messages out of the power information that are transferred to a host computer for evaluation purposes and further processing. Note that the averaging and the debug-trace generator module are optional. They are only required for the power-aware embedded software design approach and can be omitted for on-chip power management.

4.4 Design Flow

Fig. 5 outlines the design flow of the estimation-based real-time power profiling approach. A synthesizeable RTL-model of the target system forms the basis for the characterization process. After synthesis gate-level simulations based on micro-benchmarks are performed and activity information as well as power profiles are acquired based on value change dump (VCD) files. These information is fed to a power modeling process, deriving power model coefficients.

The target system's and the PE-architecture's RTL-models are merged for the generation of a single netlist. This can either be a netlist for an FPGA-platform to allow for estimation-based power-aware embedded software design or a netlist for chip synthesis incorporating the power profiling unit for estimation-based on-chip power management. Power model coefficients determined beforehand are used to configure the power sensors for tailoring the PE-architecture to the given target system.

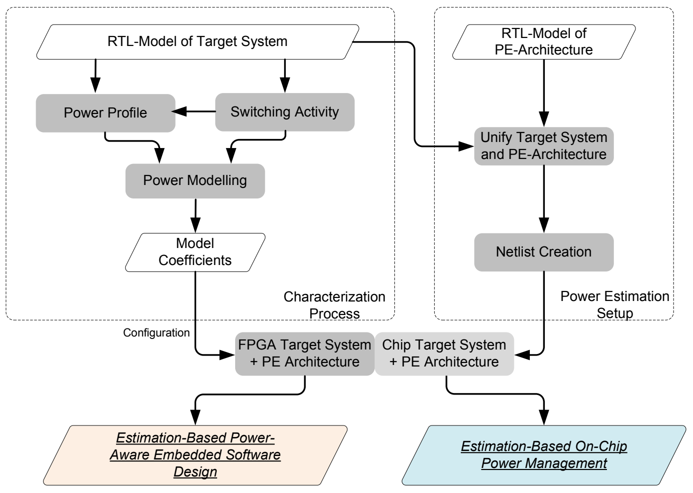

RTL-Model of Target System
Power Profile
Switching Activity
Power Modelling
Model Coefficients
Characterization Process
RTL-Model of PE-Architecture
Unify Target System and PE-Architecture
Netlist Creation
Power Estimation Setup
Configuration
FPGA Target System + PE Architecture
Chip Target System + PE Architecture
Estimation-Based Power-Aware Embedded Software Design
Estimation-Based On-Chip Power Management

Fig. 5. Design flow allowing for two possible applications: (i) estimation-based real-time power profiling for enabling power-aware embedded software design, (ii) estimation-based on-chip power management

4.5 System Set-Up

Power model coefficients obtained during the characterization process deliver configuration data for the power sensors. Listing 1.1 illustrates how to configure power sensors to tailor them to the power consumption of system modules. 16-bit registers are provided for this purpose.

```
// start of program

//configuration of power sensor1
PWRSEN0_STATE0 = 0x005A;   //CPU run mode
PWRSEN0_STATE1 = 0x0011;   //CPU halt mode
PWRSEN0_STATE2 = 0x0013;   //CPU sleep mode

//configuration of power sensor2
```

```
PWRSEN1_STATE0 = 0x0013;   //memory read
PWRSEN1_STATE1 = 0x001A;   //memory write
...

activate_power_estimation();

start_main_program();
```

Listing 1.1. Power estimation set-up, power sensor configuration

A variable number of system states for a variable number of system modules can be configured with power coefficients. Finally, power profiling is activated before normal application execution starts. The run-time overhead introduced due to power sensor configuration is negligible compared to the number of cycles executed by a typical application. A debug-trace message containing power information is automatically generated and transferred to the host computer for power information evaluation and profile visualization without the interaction of the software designer.

If power analysis of sub-modules of the system is desired, the power sensor attached to the module of interest is configured as illustrated in Listing 1.1. The configuration of the remaining modules is skipped, so they do not contribute to the overall power consumption.

For on-chip power management, the abovementioned power sensor configuration process is done in hardware by hardwiring power consumption information held by the power sensors.

5 Case Study: Profiling of Power-Critical Smart-Card Applications

Smart-card applications have been penetrating manifold market segments in the last years. Access control, electronic passport or payment are only a few out of many existing applications. Smart-cards can be categorized in (i) *contact-based* and (ii) *contact-less* derivatives. Contact-based smart-cards are powered if inserted into a reader device, while contact-less systems consume power via an RF-field generated by the reader device. Therefore, contact-less devices are subjected to stringent power limitations.

Fig. 6 illustrates the coupling of the reader device with the contact-less smart-card by a magnetic field H. A certain amount of power is transferred from the reader device to the smart-card at a time. The transferable power is limited, hence exceeding a maximum power limit due to power-peaks causes supply voltage drops that in turn can affect system stability if the supply voltage falls below a certain voltage limit $V_{DD_{critical}}$. This case-study demonstrates the capability of our estimation-based power profiling approach to support the software designer early in the development process to avoid such worst-case scenarios.

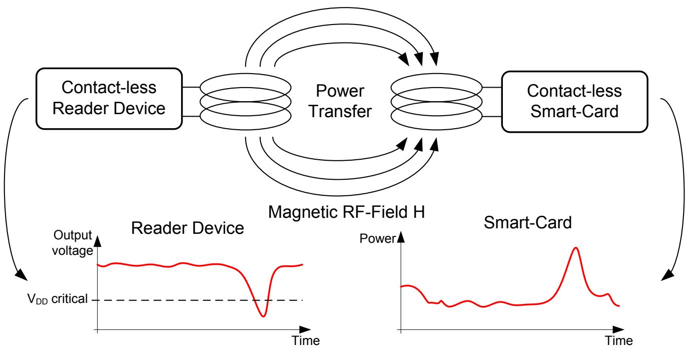


Fig. 6. Power supply of a contact-less smart card by a magnetic field generated by a reader device

5.1 Smart-Card Architecture Overview

Fig. 7 depicts a typical contact-less smart-card system. It is based on a 16-bit pipelined cache architecture comprising a memory encryption/decryption (MED) unit and volatile and non-volatile memories. Cryptographic coprocessors are included for Advanced Encryption Standard (AES), Data Encryption Standard (DES) and RSA algorithm acceleration. Moreover peripherals, such as timers or a random number generator (RNG) are provided. For communication purposes a number of interfaces (i.e., UART, I2C, contact-less interface) exist. System modules are powered by an externally generated RF-field. Energy is collected by an antenna system and power supply conditioning and stabilization by means of an analog front-end are carried out.

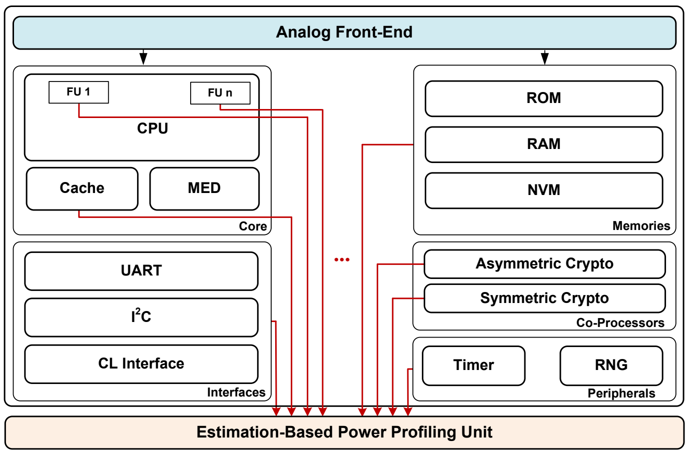


Fig. 7. Block diagram of a typical contact-less smart-card system incorporating the estimation-based power profiling unit

System modules that are major contributors to the overall power consumption are identified during a power characterization process. Moreover, available operating modes of each system module influencing the amount of power consumed are considered during this characterization process. A number of microbenchmarks in order to test many of these operating modes were applied. Based on the result of the characterization process, power model coefficients were obtained to configure the power sensors as shown in Listing 1.1. Table 1 summarizes system modules and corresponding operating modes considered for the power model.

Table 1. Operating modes of typical smart-card components considered in the power model

Unit	Mode(s)
CPU	run, halt
MED	encryption, decryption
Cache	read, write (hit, miss)
Memories	read, write
UART	read, write
Peripherals	on, off
Coproc.	encryption: AES128/192/256, (Single-, Double-, Triple-) DES
Coproc.	decryption: AES128/192/256 (Single-, Double-, Triple-) DES

5.2 Payment Application Profile Analysis

A typical application for smart-cards incorporating a symmetric cryptographic coprocessor is payment. Payment applications usually contain authentication procedures requiring cryptographic operations. Fig. 8 illustrates a power-relevant section of a typical power profile of a future payment application obtained with our estimation-based power profiling approach. The figure shows two distinct power peaks. The first one marks the power consumption of AES encryption computations, while the second peak results from AES decryption computations.

We assume that the maximum available power provided by the RF-field is 0.9 as shown in Fig. 8 (note that power values are normalized for confidentiality reasons). Hence, the payment application execution would fail due to power peaks caused by coprocessor operation.

If not obvious to the designer at this step, the source of the power peaks can be identified by decomposing the smart-card system into sub-modules. Power profiles for sub-modules are acquired by reconfiguring the power sensors. Fig. 9 depicts power profile results for the smart-card's CPU, cache (both incorporated in the core) and the coprocessors decomposed for sub-module power profile analysis. As a reference also the accumulated power profile that indicates the overall power consumption is shown.

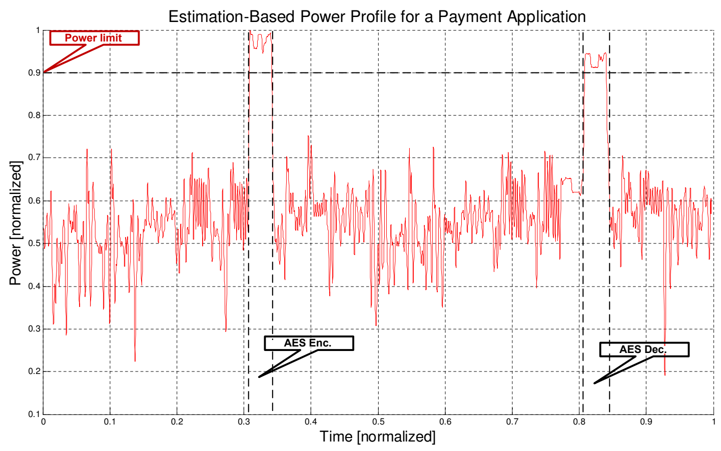


Fig. 8. Power-relevant section of a payment application power profile obtained by estimation-based power profiling (power exceeds affordable level)

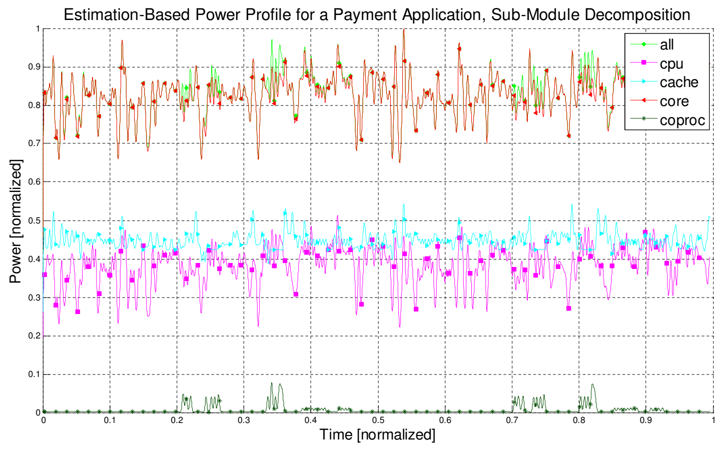


Fig. 9. Payment application power profile decomposed for sub-module power analysis

Major contributors to the overall power consumption for the given example are obviously CPU, cache and the coprocessor, when active. Peripherals, communication interfaces and also memories are inactive during the considered time frame of the payment application and therefore do not add to the system's power consumption. It can be clearly observed that the coprocessor's power consumption causes the overall power profile to exceed the absolute maximum power of 0.9.

Various countermeasures could be taken to circumvent this issue. The AES algorithm could be implemented in software to avoid using coprocessor acceleration. An alternative is also to reduce the system's clock frequency. Fig. 10 shows the power profile when scaling down the system frequency from 33 MHz to 28 MHz. Power peaks are scaled down below 0.9, hence frequency scaling resolves the power peak issue. Both remedies come at the cost of reducing the payment application's performance, but they ensure reliable operation.

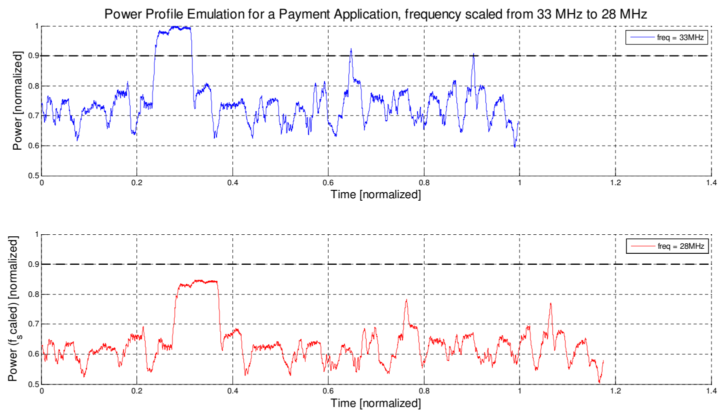


Fig. 10. Payment application power profile obtained by estimation-based power profiling (stable system operation due to frequency-scaling)

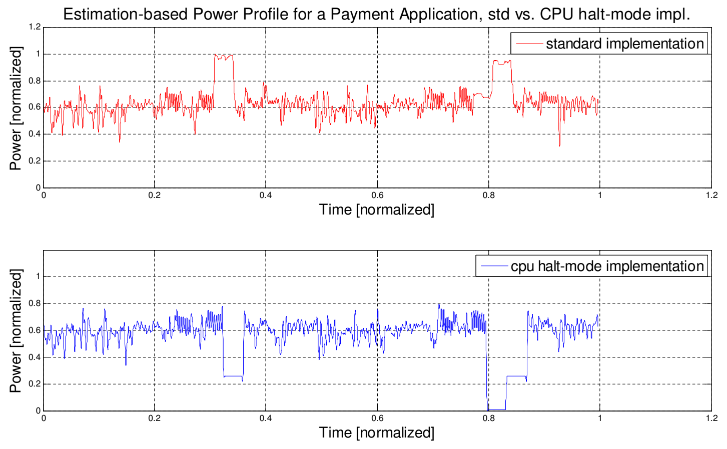


Fig. 11. Payment application power profile obtained by estimation-based power profiling (stable system operation due to CPU low-power features exploitation)

The system's power consumption can also be reduced by exploiting CPU low-power features during the critical peak phase. During coprocessor activity the CPU as a main power consumer is shut down and reactivated after the coprocessor has finished execution. Power peaks can be avoided by applying this countermeasure without degrading the system's performance. Fig. 11 illustrates the effect of the CPU low-power implementation.

5.3 Accuracy of Estimation-Based Power Profiling

Fig. 12 shows power profiles of the payment application. A comparison between the estimation-based power profiling result and gate-level power simulation profiles obtained with *Magma Blastfusion 5.2.2* [21] is given.

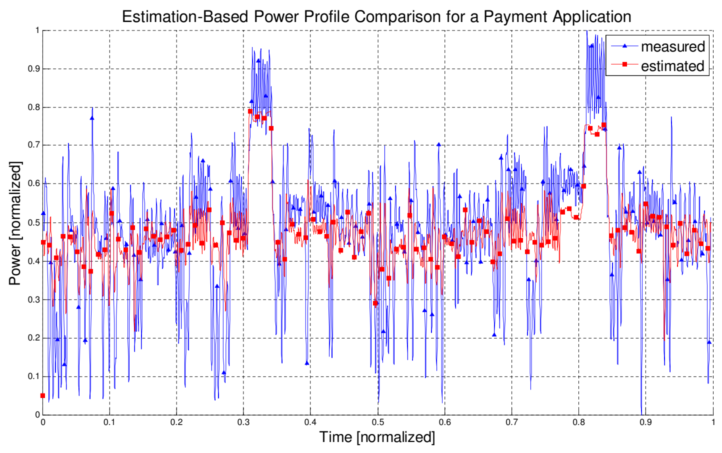


Fig. 12. Power profile comparison, gate-level power simulation profile vs. estimation-based power profile

Accuracy considerations for other executed test applications are summarized in Table 2. The relative power error on average and the corresponding variance are given. For all tested applications relative average power errors are less than 10%.

5.4 Performance Evaluation

One of the major advantages of the hardware-accelerated estimation-based power profiling approach is the capability to acquire power profiles in real-time. Power estimation takes no longer than application execution on the target-system. Hence, the power profile is available immediately after execution. Table 3 shows execution times of power profile simulations on a gate-level basis using *Magma Blastfusion* compared to our estimation-based approach. Simulation times are depicted compared to hardware-accelerated power estimation run-times. Power estimation of the payment application takes 136 microseconds, whereas the

Table 2. Power estimation accuracy comparison for a number of executed micro-benchmarks

Algorithm	Duration (μs)	# Cycles	Error (%)		
			Power		Energy
			avg.	σ^2	
ALU	69.5	2293	4.9	0.9	3.1
Cache	120.5	3978	2.2	2	0.5
Dhrystone	126.5	4176	3.9	2.7	1.4
Memory	52.1	1722	-6.1	1.3	-7.8
Payment (Coproc., CPU halt)	136	4510	5.7	3.8	1.6
Payment (Coproc., CPU run)	146.6	4837	2.2	2.9	-0.5
DES (Coproc., CPU halt)	87.8	2899	5.3	4	1.8
DES (Coproc., CPU run)	93.1	3072	4.5	3.9	1.1

power simulation is about 17 hours. Hence, our approach shows that extensively high simulation times can be reduced enormously by means of the hardware-accelerated estimation-based power profiling unit.

Table 3. Performance comparison of power profiling, simulation vs. hardware-accelerated estimation-based approach

Algorithm	# Cycles	Duration	
		Simulation [4] (hours)	Estimation-Based (microseconds)
ALU	2293	4.1	69.5
Cache	3978	14.9	120.5
Dhrystone	4176	18.6	126.5
Memory	1722	5.9	52.1
Payment (Coproc., CPU halt)	4510	17.8	136
Payment (Coproc., CPU run)	4837	23.8	146.6
DES (Coproc., CPU halt)	2899	9.5	87.8
DES (Coproc., CPU run)	3072	10.2	93.1

[4] Simulation has been performed on a state of the art server system. The sampling rate at which power simulations have been performed is 33 MHz, which is the clock frequency of the smart-card system.

It is obvious that power simulation of complex applications is rendered unfeasible due to far too extensive simulation times. Therefore, power saving potential or power peaks leading to system failure cannot be detected at an early design stage. By means of the hardware-accelerated estimation-based power profiling approach, power estimates are available immediately and already when working with FPGA prototyping platforms. Countermeasures to circumvent power peaks can be taken before the device is available in silicon and physical measurements can be performed.

5.5 Resource Allocation

The smart-card system incorporating the PE-architecture has been synthesized on an *Altera Stratix II FPGA* [22]. For power estimation purposes additional 1.5% of hardware is added to the overall system for a power model consisting of 40 model parameters. In Table 4, the resource allocation of different units of the PE-architecture is shown. The largest portion of resources is occupied by the system bus interface together with the power tables load mechanism as well as by the power sensors and the power estimation unit. For a more complex power model, power sensor and power estimation resources are expected to grow linearly with the number of model parameters.

Table 4. Composition of the amount of combinational and register resources allocated by the PE-architecture on an *Altera Stratix II FPGA*

	Combinational Logic (%)	Registers (%)
Bus IF, Load Mechanism [5]	32.6	7.2
Pwr. Sensors, PE Unit	58.1	78.5
Averaging	5.4	10.7
Debug-Trace Gen.	3.9	3.6

[5] The PE-architecture is interfaced to the system bus allowing for memory-mapped power sensor configuration via register accesses

6 Concept for Estimation-Based On-Chip Power Management

Power management has emerged as an effective technique to gain power and energy efficiency in embedded systems. The system is dynamically reconfigured during run-time to provide the requested system performance under the consideration of given power constraints. System reconfiguration can either be accomplished by switching off unused system components or by dynamic voltage and frequency scaling (DVFS). In the latter case, system adoption is carried out by dynamically changing system frequency and supply voltage levels.

The concept of estimation-based power management is illustrated in Fig. 13 by means of a smart-card use-case. The hardware-accelerated estimation-based power profiling unit is integrated on-chip together with the smart-card system. Power consumption information is provided cycle-accurately and in real-time to a power manager that autonomously adopts system frequency and supply voltage

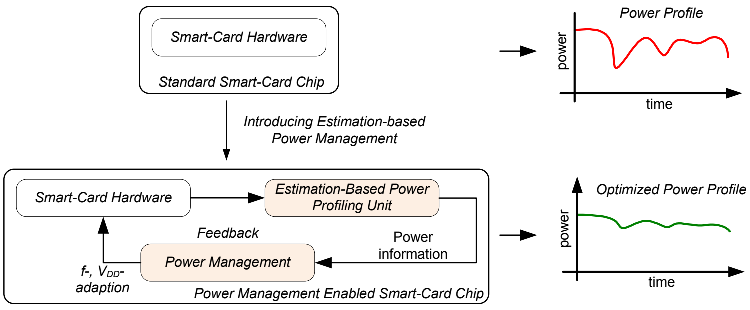


Fig. 13. Power management principle based on the hardware-accelerated estimation-based power profiling unit

in order to smoothen the power profile. Alongside making a system more power-efficient, we assume that DVFS power management can be an effective measure for energy scavenging devices to make them more robust against power peaks.

7 Conclusions and Future Work

Extensive run-times of power simulators render power analysis of increasingly complex embedded systems unfeasible. The hardware-accelerated estimation-based power profiling approach proposed in this work, delivers power information to the software designer in real-time. Moreover, by employing FPGA prototyping platforms, these power information is available already at early design stages. The proposed power profiling unit has proven to be an effective option to estimate the system's power consumption and to deliver power information with accuracies of above 90% on average. This paves the way for power-efficient embedded software design and the capability to cope with and to avoid power critical events during the design phase more efficiently.

In our future work, we will focus on power management by leveraging power consumption information of our developed power profiling unit. Power management policies enhancing the system's power efficiency as well as reducing the risk of malfunctions due to power peaks is going to be the focus of our prospective research.

References

1. E. Macii and M. Poncino. *Power Macro-Models for High-Level Power Estimation*, chapter 39 Low-Power Electronics Design, pages 39–1—39–18. CRC Press, edited by C. Piguet, 2005.
2. A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss. An emulation-based real-time power profiling unit for embedded software. In *SAMOS*, pages 67 –73, 2009.

3. J. Flinn and M. Satyanarayanan. PowerScope: a tool for profiling the energy usage of mobile applications. In *WMCSA*, pages 2–10, 1999.
4. Texas Instruments. *Analyzing Target System Energy Consumption in Code Composer Studio^{TM} IDE*, 2002.
5. J. Flynn and B. Waldo. Power Management in Complex SoC Design. Technical report, Synopsys Inc. White Paper, 2005.
6. V. Tiwari, S. Malik, and A. Wolfe. Power Analysis Of Embedded Software: A First Step Towards Software Power Minimization. In *ICCAD*, pages 384–390, 1994.
7. M. Sami, D. Sciuto, C. Silvano, and V. Zaccaria. An instruction-level energy model for embedded VLIW architectures. In *IEEE Trans. on CAD of Integrated Circuits and Systems*, volume 21, pages 998–1010, 2002.
8. M. Lajolo, A. Raghunathan, S. Dey, and L. Lavagno. Cosimulation-based power estimation for system-on-chip design. In *IEEE Trans. on Very Large Scale Integration Systems*, volume 10, pages 253–266, 2002.
9. I. Lee, H. Kim, P. Yang, S. Yoo, E. Chung, K. Choi, J. Kong, and S. Eo. PowerViP: SoC Power Estimation Framework at Transaction Level. In *ASP-DAC*, pages 551–558, 2006.
10. S. Ahuja, D. A. Mathaikutty, G. Singh, J. Stetzer, S. K. Shukla, and A. Dingankar. Power estimation methodology for a high-level synthesis framework. In *ISQED*, pages 541–546, Washington, DC, USA, 2009. IEEE Computer Society.
11. F. Bellosa. The benefits of event: driven energy accounting in power-sensitive systems. In *SIGOPS European Workshop*, pages 37–42, 2000.
12. R. Joseph and M. Martonosi. Run-time power estimation in high performance microprocessors. In *ISLPED*, pages 135–140, 2001.
13. W. L. Bircher and L. K. John. Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events. In *ISPASS*, pages 158–168, April 2007.
14. J. Haid, G. Kaefer, Ch. Steger, and R. Weiss. Run-time energy estimation in system-on-a-chip designs. In *ASP-DAC*, pages 595–599, 2003.
15. J. Coburn, S. Ravi, and A. Raghunathan. Power emulation: a new paradigm for power estimation. In *DAC*, pages 700–705, 2005.
16. M.A. Ghodrat, K. Lahiri, and A. Raghunathan. Accelerating System-on-Chip Power Analysis Using Hybrid Power Estimation. In *DAC*, pages 883–886, 2007.
17. A. Bhattacharjee, G. Contreras, and M. Martonosi. Full-System Chip Multiprocessor Power Evaluations Using FPGA-Based Emulation. In *ISLPED*, pages 335–340, 2008.
18. A. Bogliolo, L. Benini, and G. De Micheli. Regression-based RTL power modeling. In *ACM Trans. on Design Automation of Electronic Systems*, volume 5, pages 337–372, 2000.
19. C. Krintz and S. Gurun. A run-time, feedback-based energy estimation model for embedded devices. In *CODES+ISSS*, pages 28–33, 2006.
20. C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid. Automated Power Characterization for Run-Time Power Emulation of SoC Designs. In *DSD, In press.*, 2010.
21. Magma Design Automation Inc., Blastfusion. (`http://www.magma-da.com/`), July 2010.
22. Altera Coroporation, Stratix II. (`http://www.altera.com/`), July 2010.

Power Emulation Based DVFS Efficiency Investigations for Embedded Systems

Andreas Genser[1], Christian Bachmann[1], Christian Steger[1], Reinhold Weiss[1], Josef Haid[2]
[1]Institute for Technical Informatics, Graz University of Technology
{andreas.genser, christian.bachmann, steger, rweiss}@tugraz.at
[2]Design Center Graz, Infineon Technologies AG
josef.haid@infineon.com

***Abstract*— Power consumption has become a major design constraint in the embedded systems domain and techniques such as dynamic voltage and frequency scaling (DVFS) have emerged to enhance the system's power and energy efficiency. DVFS-enabling voltage regulators influence the performance, power and energy efficiency of such systems, however, this impact is often neglected or considered late in the design process. In this work, we propose DVFS hardware extensions to a power emulation approach for modeling the voltage regulator behavior, which allows for performance, power and energy efficiency investigations of DVFS-enabled embedded systems. The power emulation approach delivers real-time power information in an early design phase, which allows for the exploration of DVFS efficiency before silicon is available. This offers greater freedom to designers to determine the most apt voltage regulator yielding a system that meets performance, power and energy constraints.**

I. Introduction

A. Overview

In recent years, an emerging gap between the available power and the complexity of mobile devices can be observed. However, embedded system run-times are still required to be prolonged despite of increasingly restricted energy budgets. Hence, power- and energy efficiency have become a major design target in the embedded systems domain.

In order to gain system efficiency, various low-power design techniques as well as run-time power-aware measures have been introduced. Low-power design techniques have been developed on various levels of abstraction, from the transistor-level up to the software-level. Alternatively, run-time power-aware measures (i.e., power management techniques) aim at the minimization of the power and energy consumption of a system during run-time by adapting internal system states. First, dynamic power management techniques focus on the minimization of the number of active system components by switching them off when idle. Alternatively, dynamic frequency scaling (DFS) and dynamic voltage and frequency scaling (DVFS) have emerged as effective dynamic power management techniques. Instead of completely switching off system components, energy savings are achieved by varying the frequency and the supply voltage of the system.

Many of today's embedded systems are featured with voltage regulators allowing for the switching of the supply voltage. Ideally, the switching can be carried out in zero time and to any voltage level. However, voltage regulators are subject to constraints that impact on the system performance as well as power and energy consumption. First, the number of supported voltage levels of the voltage regulator is limited. Moreover, the switching between two voltage levels is not feasible in an infinitely small amount of time. Although many DVFS works have been proposed in the past, most of them lack the consideration of the real voltage regulator behavior and its impact on system performance as well as the system's power and energy efficiency. The execution time-penalty induced by the switching between different voltage levels can cause timing deadlines of executed tasks to be violated. Moreover, power and energy consumption computations made by assuming ideal voltage regulator behavior might yield results that deviate from the real system's power and energy consumption, respectively. This is particularly critical in the domain of prohibitively power-constrained systems, such as embedded systems that are powered by energy-scavenging devices, where wrong power consumption information during the design phase can lead to malfunctions during run-time. Functional and power simulations of the system allow for the early design phase investigation of performance, power and energy efficiency. However, state of the art power simulations that are frequently performed on a gate-level basis lead to extensive simulation times, which render power simulations of complex applications unfeasible.

In this work, which is part of the PowerHouse[3] project, we propose an FPGA-based prototyping power emulation platform allowing for real-time performance, power and energy-efficiency investigations of DVFS-enabled embedded systems. The embedded system is functionally emulated on an FPGA-platform and power consumption information is derived from power estimation hardware [1]. The power estimation hardware allows for the real-time acquiration of cycle-accurate power consumption information, which is superior in terms of speed over power simulations. Moreover, we introduce configurable DVFS hardware extensions to model voltage regulator behavior. This approach allows for the early design phase investigation of performance, power and energy consumption of DVFS-enabled embedded systems.

[3]Project partners are Infineon Technologies Austria AG, Austria Card GmbH and TU Graz. The project is funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 815193.

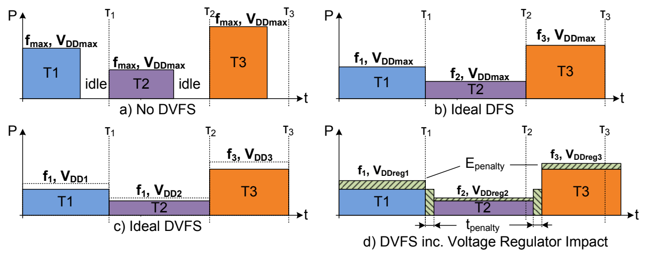


Fig. 1. DFS and DVFS optimization potential incorporating time- and energy-penalty induced by the voltage regulator behavior

B. Motivational Example

The goal of DVFS is to minimize the power and energy consumption of an embedded system by still fulfilling given timing constraints. We assume a set of three tasks T=(T_1, T_2, T_3) that have been scheduled according to an earliest-deadline-first (EDF) policy.

Fig. 1 illustrates energy optimization results of different optimization strategies to not violate the task deadlines τ=(τ_1, τ_2, τ_3). In Fig. 1a all three tasks are executed at the maximum system frequency f_{max} and the maximum supply voltage $V_{DD_{max}}$, thus finishing tasks early. The remaining period of time, the system is idle. By introducing dynamic frequency scaling (DFS), the system frequency is adjusted to f=(f_1, f_2, f_3) for T_i in a way that task deadlines τ_i are still met. This is depicted in Fig. 1b. DFS does not influence the energy consumption of the system but reduces the power consumption. When applying DVFS, we assume the embedded system to be equipped with a voltage regulator supporting three different supply voltage levels V_{DD}=(V_{DD_1}, V_{DD_2}, V_{DD_3}). Thus, DVFS methods can be applied to reduce power and energy consumption, which is illustrated in Fig. 1c.

In general, voltage regulator switching between different voltage levels is not feasible in zero time, which causes a time-penalty that can violate task deadlines τ_i. Furthermore, the chosen voltage regulator might not support the ideal supply voltage levels V_{DD_i}. If so, it has to be adjusted to the closest next higher supported voltage level V_{DDreg_i} to reliably operate the system. This is illustrated in Fig. 1d yielding a higher power and energy consumption compared to the ideal case shown in Fig. 1c. Often the real behavior of DVFS-enabled systems as shown in Fig. 1d is neglected and simplified to the case depicted in Fig. 1c.

II. Related Work

In order to determine the impact of voltage regulators on the power and energy consumption of a system, power profiling is required. Basically, two groups of power profiling methods can be distinguished, (i) *physical power profiling* and (ii) *estimation-based power profiling*.

Physical power profiling gathers power information by carrying out actual physical measurements. However, expensive measurement equipment is required and power information is not available before the silicon implementation of the system has been manufactured [2]. Hence, physical power profiling is only applicable in a late design phase.

Early design phase power profiling can be conducted by *estimation-based* methods that derive power information from power models that are evaluated by simulations or in hardware. *Simulation-based* methods as proposed in [3,4] often suffer from extensive simulation times, which hinders power profiling of complex applications. To circumvent this limitation *hardware-accelerated* approaches were introduced that shift power model computations from software to hardware. This yields high speedups over simulation-based approaches [1].

Power emulation is a special form of hardware-accelerated power estimation. A prototyping platform, such as FPGA-boards, is used to emulate the power consumption of a system alongside performing functional emulation. The level of abstraction power emulation is carried out impacts on the power model evaluation performance and estimation accuracy. On lower abstraction levels, such as register transfer level (RTL), high accuracies are achievable. However, power emulation on these abstraction levels suffers from a high demand of hardware resources. Hence, current efforts tend to move to higher levels of abstraction (e.g., system-level). Coburn et al. first presented an RTL power emulation approach in [5] yielding run-time improvements of 10x to 500x compared to commercial power estimation tools. However, this comes at the cost of hardware overhead of up to 3x compared to the original design. In recent work, we introduced a system-level power emulation approach providing real-time capability and less hardware overhead [1]. This forms the basis for the proposed early design phase power and energy efficiency investigations for DVFS-enabled systems.

Many DVFS works have been published in the past that aim at power and energy efficiency enhancements. Pillai et al. proposed different DVFS algorithms for embedded systems in order to guarantee real-time behavior of operating systems [6]. They evaluate energy efficiency gains by simulating the system within a C++-simulator. For each executed cycle, they assume a constant amount of energy consumed. This is a simplified assumption, since power and energy consumption might vary greatly during algorithm execution. Moreover, the voltage regulator impact on performance, power and energy consumption is neglected.

Calhoun et al. presented a 90nm test chip to permit DVFS with the purpose of energy reduction for fixed throughput systems [7]. In [8], Lee et al. developed a hardware-software power control scheme to reduce the power consumption of a commercial processor by applying frequency and voltage adaptions. These adaptions are performed by a simple power control chip. Hotta et al. proposed a profile-based power performance optimization system based on DVFS adaptions for PC clusters [9]. The voltage regulator impact is implicitly considered by performing physical measurements on the final chip. However, these information is delivered late in the design process.

The combination of the power emulation approach together with DVFS hardware extensions, as presented in this paper, provides a promising method to perform early design phase performance, power and energy investigations for DVFS-enabled systems. The contributions of this work can be summarized as follows:

- Modeling of the DVFS time- and energy-penalty in hardware allowing for rapid DVFS investigations of complex applications (e.g., operating systems)
- Early design phase applicability due to the power emulation approach
- Voltage regulator impact on performance, power and energy consumption is quantified before the final chip is available

III. DVFS Preliminaries

A. System Frequency and Supply Voltage Relation

DVFS techniques adapt the system frequency f and supply voltage V_{DD} to meet performance demands at the highest possible energy efficiency. For DVFS energy efficiency investigations, we assume a set of tasks $T = (T_1, T_2, ..., T_{n-1})$ incorporating a set of corresponding task deadlines $\tau = (\tau_1, \tau_2, ..., \tau_{n-1})$. The well known relation for the energy consumption of a CMOS-based embedded system $E = \alpha\ C\ V_{DD}^2$ for a certain task T_i yields

$$E^{T_i} = \alpha_i\ C_{eff}\ V_{DD_i}^2\ N_c^{T_i}, \tag{1}$$

where α_i corresponds to the switching activity during task execution and V_{DD_i} gives the supply voltage transferring energy E^{T_i} to the effective capacitance C_{eff}. The number of executed cycles for a task T_i can be written as $N_c^{T_i} = f_i\ \tau_i$. Energy efficiency gains for a task T_i can be achieved by the reduction of the supply voltage V_{DD_i}. However, reducing V_{DD_i} increases the circuit delay d, which is inversely proportional to the system frequency f as shown in (2).

$$d \propto 1/f \approx k_d\ \frac{V_{DD}}{(V_{DD} - V_{th})^2} \tag{2}$$

k_d represents a CMOS device constant and V_{th} gives the CMOS transistor's threshold voltage [10]. Equ. (1) and (2) imply that the energy efficiency is increased when V_{DD} is reduced, but execution speed is degraded. Consequently, each task deadline τ_i requires a certain system frequency f_i to not violate the deadline. In turn, a minimum supply voltage level V_{DD_i} is required to reliably operate the system at the system frequency f_i.

B. Voltage Regulator Properties

DVFS is enabled by voltage regulators that perform the switching of the supply voltage between different voltage levels. Voltage regulator properties impact on the system performance and its power and energy consumption. In [11], Burd et al. proposed a metric giving the *time-penalty* introduced by voltage regulators. This is stated in (3).

$$t_p^{T_{i,j}} \propto \frac{C}{I_{max}}\ |V_{DD_i} - V_{DD_j}| \tag{3}$$

I_{max} gives the maximum current that can be drawn from the voltage regulator and C determines the voltage regulator's capacitance. Both terms as well as the difference between the present supply voltage V_{DD_i} and the future supply voltage level V_{DD_j} required to meet the task deadlines for T_i and T_j, respectively, impact on $t_p^{T_{i,j}}$.

Moreover, we introduce a metric representing the *energy-penalty* if the minimum supply voltage level V_{DD_i} is not supported by the voltage regulator. The system is then operated at the next higher supported supply voltage level V_{DDreg_i}. This yields an energy consumption overhead as stated in (4).

$$E_p^{T_i} = \alpha_i\ C_{eff}\ N_c^{T_i}\ \left|V_{DDreg_i}^2 - V_{DD_i}^2\right| \tag{4}$$

IV. Power Emulation Hardware and DVFS Extensions

Functional and power emulation form the basis for performance, power and energy evaluations of DVFS-enabled embedded systems. The power estimation hardware provides real-time power consumption information based on the evaluation of a power model that is set up by a power characterization process. Both, the power estimation hardware as well as the characterization process have been presented in previous work [1,12]. We extend this system with DVFS hardware extensions for modeling the voltage regulator impact on performance, power and energy consumption.

A. Power Model and Characterization

Linear regression methods as proposed in [13] are frequently used to set up system-level power models. Their simple structure makes them suitable to be implemented in hardware. A linear regression model can be given as

$$\hat{y} = \sum_{i=1}^{n} c_i x_i + \epsilon, \tag{5}$$

where x_i mark model parameters covering system states, such as CPU operating modes (e.g., run, idle) or memory accesses (e.g., read, write). c_i represent model coefficients and ϵ gives the estimation error. The evaluation of (5) yields the power estimate $\hat{y}$. Power model coefficients c_i are to be determined during a power characterization process. In [12], we presented the underlying characterization methodology to set up an accurate system-level power model. The quality of the characterization process determines the accuracy and hardware effort of the power model and can be set up by the following steps.

- *Selection of model parameters*. This step seeks for power-relevant model parameters. The potentially huge number of model parameters is minimized by computing cross-correlations to determine redundancies.
- *Training-set execution*. A number of micro-benchmarks is executed on the embedded system and corresponding power profiles are acquired based on gate-level power simulations.
- *Least squares fit method*. Model coefficients c_i are determined based on model parameters x_i and the obtained

power profiles. The least squares fit method minimizes the square error for each model coefficient c_i.

B. Power Estimation Hardware

The established power model in (5) is incorporated in the power estimation hardware as shown in Fig. 2. Power sensors hold power model coefficients c_i and track state changes of system modules. Based on these state information, each power sensor provides power information for a certain system module, which is fed to the power estimation unit for power information accumulation. Finally, a debug-trace generator assembles functional and power messages to be transferred to a host computer. Optionally, an averaging module is provided for power profile denoising and smoothing purposes.

C. DVFS Hardware Extensions

1) DVFS Adaption Modeling: The modeling of the time-penalty $t_p^{T_{i,j}}$ of the voltage regulator when switching between two supply voltage levels as stated in (3) is shown in Fig. 3.

If the system frequency f is required to be increased from f=f_n to f=f_{n+1} at t=t_n, the time overhead introduced by the voltage regulator until the output voltage has stabilized from V_{DD}=V_{DDreg_n} to V_{DD}=$V_{DDreg_{n+1}}$ leads to a delay of $t_p^{T_{i,j}}$. To ensure reliable system operation according to (2) also the system frequency f is switched time-delayed at t=t_n+$t_p^{T_{i,j}}$. This is stated in (6) and (7).

$$f = \begin{cases} f_n, & \text{if } t \leq t_n + t_p^{T_{i,j}} \\ f_{n+1}, & \text{else.} \end{cases} \tag{6}$$

If the system frequency f and the supply voltage V_{DD} are required to be reduced, f can be reduced from f=f_{n+1}

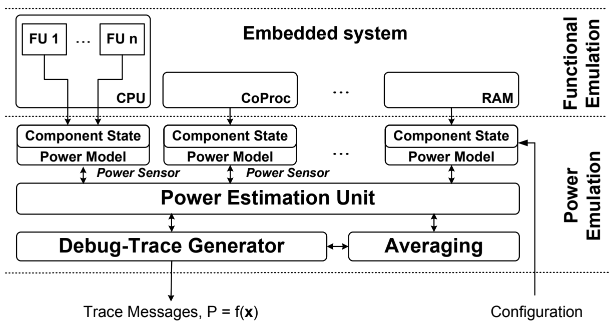


Fig. 2. Power estimation hardware deriving power information from system activity, obtained with changes from [1]

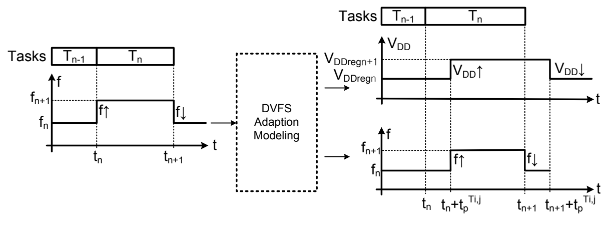


Fig. 3. DVFS adaption modeling emulating the voltage regulator behavior

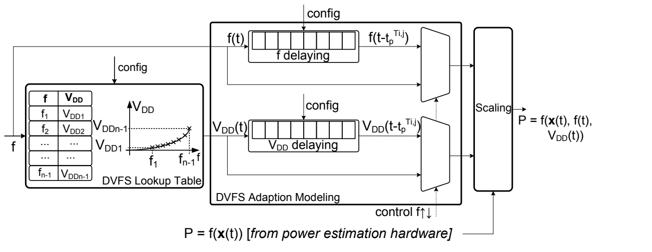


Fig. 4. DVFS hardware extensions, voltage regulator modeling by DVFS lookup tables, delay modeling and scaling

to f=f_n immediately at t=t_{n+1} without threatening system stability, while reducing V_{DD} from V_{DD}=$V_{DDreg_{n+1}}$ to V_{DD}=V_{DDreg_n} is performed time-delayed at t=t_{n+1}+$t_p^{T_{i,j}}$.

$$V_{DD} = \begin{cases} V_{DDreg_n}, & \text{if } t \leq t_n + t_p^{T_{i,j}} \\ V_{DDreg_{n+1}}, & \text{if } t_n + t_p^{T_{i,j}} < t \leq t_{n+1} + t_p^{T_{i,j}}. \end{cases} \tag{7}$$

The DVFS hardware extensions modeling the voltage regulator behavior are depicted in Fig. 4. Two configurable delay lines model the time-penalty $t_p^{T_{i,j}}$ for both f and V_{DD}.

2) DVFS Lookup Table: The maximum system frequency is determined by the underlying process technology, the system's complexity and the supply voltage V_{DD}. Based on that, a number of (f, V_{DD})-pairs can be derived for each supply voltage level supported by the voltage regulator. We extract these pairs from post-synthesis system information, which are then stored in a DVFS lookup table as depicted in Fig. 4. The number of table pairs and their contents are configurable via register accesses.

3) Scaling Unit: The power estimation hardware derives power consumption information $P = f(\mathbf{x}(t))$ based on system module states $\mathbf{x} = [x_1, x_2, ...x_{n-1}]$. The obtained power information is state-dependant, but independant from the system frequency f and the supply voltage V_{DD}. For DVFS efficiency investigations, a scaling unit scales power consumption information within power trace messages with present f and V_{DD} settings yielding $P = f(\mathbf{x}(t), f(t), V_{DD}(t))$.

D. Emulated System

We emulate an HDL-model of an RF-powered smart-card SoC, based on a 16-bit pipelined cache architecture. The system incorporates volatile and non-volatile memories, symmetric and asymmetric coprocessors to accelerate cryptographic operations. Moreover, typical system modules, such as timers, random number generators, are present as well as communication interfaces (e.g., UART). The power estimation hardware traces power-relevant system states of all system modules yielding average relative estimation errors of less than 10% compared to gate-level power simulations [12].

The system is emulated on an Altera Stratix II FPGA. The emulated 16-bit test system occupies 67% of the available ALUTs, while power estimation hardware and DVFS extensions contribute only to about 2% to the system area. The power emulations's real-time capability offers speedups of several orders of magnitude compared to gate-level power simulations [1].

E. Emulation Design Flow

Fig. 5 outlines the design flow illustrating the steps from the system's HDL-model to the proposed emulation platform allowing for performance, power and energy investigations. Starting from the HDL-model of the smart-card SoC, the power model is set up by deriving power model coefficients within the power characterization process for power estimation hardware configuration. Furthermore, the merging of the HDL models of the power estimation hardware and DVFS hardware extensions with the smart-card SoC is performed, which forms the basis for an FPGA downloadable netlist. After synthesis, (f, V_{DD})-pairs according to the number of voltage levels supported by the voltage regulator are derived. Moreover, depending on the voltage regulator's characteristics C and I_{max}, $t_p^{T_{i,j}}$ is provided. f, V_{DD} and $t_p^{T_{i,j}}$ are exploited to configure the DVFS lookup table and the delay lines given in Fig. 4. If downloaded on the FPGA, functional and power consumption information from task execution are transferred to a host computer, where further performance, power and energy consumption evaluations can be carried out.

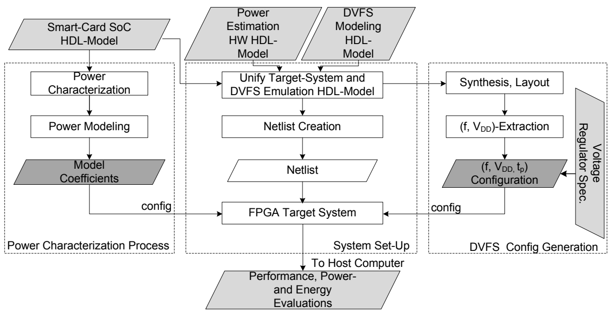


Fig. 5. Emulation design flow for DVFS performance, power and energy efficiency investigations

V. Experimental Results

We demonstrate profiling results gathered by means of the emulation platform by benchmarking a set of five tasks T=(T_1, T_2, T_3, T_4, T_5) on the emulated 16-bit smart-card SoC system according to an EDF scheduling policy. Moreover, we assume that system frequencies f=(f_1, f_2, f_3, f_4, f_5) to meet task deadlines τ_i for each task T_i are given. The task set T consists of micro-benchmarks ranging from computationally-intensive (ALU, CPU) to memory-intensive (Cache, RAM) applications. The following DVFS scenarios have been investigated: 1.) Smart-card SoC w/o DVFS operating at f_{max} and $V_{DD_{max}}$; 2.) Ideal DVFS, assuming a voltage regulator supporting an infinite amount of supply voltage levels and $t_p^{T_{i,j}} = 0$; 3.) Voltage regulator C=1μF, I_{max}=10mA, V_{DD} levels: 0.6V-1.4V in 0.1V steps leading to $t_p^{T_{i,j}}$=10μs for a 0.1V step; 4.) Voltage regulator C=1μF, I_{max}=5mA, V_{DD} levels: 0.6V-1.4V in 0.2V steps leading to $t_p^{T_{i,j}}$=40μs.

In the following, power profiling results, energy comparisons and performance evaluations for these scenarios are given. The power profiling has been executed on the emulation platform, while energy and performance comparisons are computed on the host computer based on power trace messages.

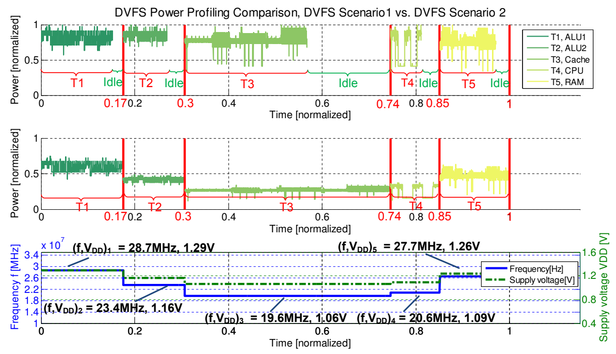


Fig. 6. DVFS power profiling result comparison, DVFS scenario 1 vs. 2

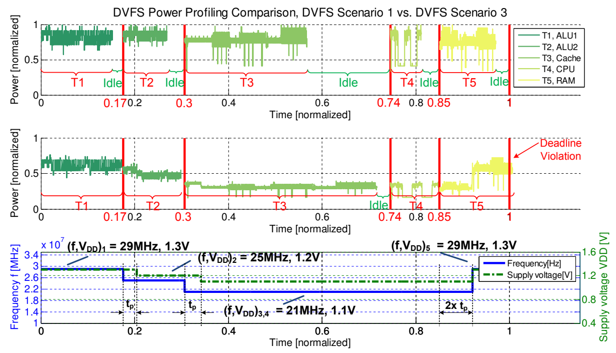


Fig. 7. DVFS power profiling result comparison, DVFS scenario 1 vs. 3

A. Profiling and Evaluation Results

Fig. 6 illustrates a power profiling result of the execution of the task set T. First, DVFS scenario 1 is shown, that finishes all tasks before their deadlines τ. When the execution of a task T_i has finished, the system is switched to idle mode. Second, Fig. 6 shows the ideal DVFS case, executing tasks at exactly the required f and V_{DD} in order to meet deadlines. No voltage regulator delay is introduced. Ideal f and V_{DD} settings are depicted at the bottom of Fig. 6.

In DVFS scenario 3, the smart-card SoC is profiled, incorporating a voltage regulator supporting supply voltage steps of 0.1V that allows for frequency adjustments in 4MHz steps. The power profiling result of the task set T is illustrated in Fig. 7 compared to DVFS scenario 1. Each task is executed at the closest next higher voltage level compared to the ideal ones in Fig. 6. Tasks T_1-T_4 finish before their deadlines. The reduction of f to each of f_1-f_3 is carried out immediately, while the voltage level switching is delayed by $t_p^{T_{i,j}}$ for each of V_{DD_1}-V_{DD_3}. No switching is required from V_{DD_3} to V_{DD_4}. In contrast, when f_5 is increased to meet τ_5 for T_5, both f and V_{DD} are delayed, which models the voltage regulator behavior for the supply voltage transition. The delayed increase of f_5 yields a deadline violation of T_5, which is shown in Fig. 7.

In DVFS scenario 4 illustrated in Fig. 8, a similar power profiling result for a voltage regulator supporting supply

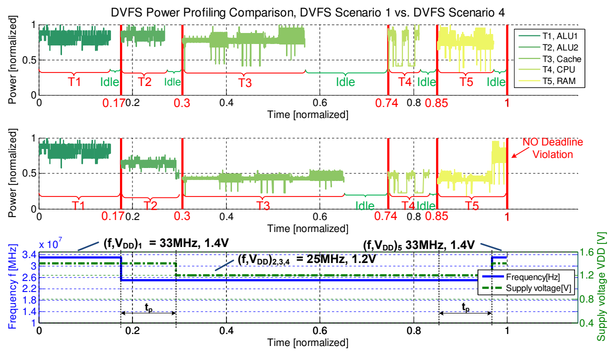


Fig. 8. DVFS power profiling result comparison, DVFS scenario 1 vs. 4

voltage steps of 0.2V to conduct frequency steps of 8MHz is shown. The smaller I_{max} compared to DVFS Scenario 3 causes higher time-penalties. Additionally, voltage levels are more coarse-grained leading to higher energy-penalties. Note that although DVFS scenario 4 introduces a higher time-penalty $t_p^{T_{4,5}}$ compared to DVFS scenario 3, it still meets the task deadline τ_5. This is mainly due to the reason that f_4=25Mhz in DVFS scenario 4 compared to f_4=21MHz in DVFS scenario 3. The higher frequency f_4 compensates the higher time-penalty $t_p^{T_{4,5}}$ in DVFS scenario 4 compared to DVFS Scenario 3.

Normalized performance results for all DVFS scenarios are given in Tab. I compared to given task deadlines τ. DVFS scenario 1 finishes first, while DVFS scenario 2 as a matter of course (ideal DVFS) finishes exactly at the required deadlines τ. All task deadlines are also met for DVFS scenario 4. In DVFS scenario 3, despite a more fine-grained and faster voltage regulator, a deadline violation occurs due to $t_p^{T_{4,5}}$.

From an energy point of view, we can observe that the system without DVFS performs worst, while the ideal DVFS implementation in DVFS scenario 2 yields lowest energy consumption results. The more fine-grained voltage regulator in DVFS scenario 3 that supports voltage levels closer to the ideal ones turns out to be superior to DVFS scenario 4 in terms of energy consumption.

TABLE I
PERFORMANCE EVALUATIONS RESULT (COMPARED TO TASK DEADLINES) FOR DIFFERENT DVFS SCENARIOS (NORMALIZED TIME VALUES)

	T1	T2	T3	T4	T5
Deadlines	0.175	0.307	0.746	0.851	1.000
DVFS Scen. 1	0.153	0.269	0.568	0.812	0.971
DVFS Scen. 2	0.175	0.307	0.746	0.851	1.000
DVFS Scen. 3	0.174	0.299	0.717	0.849	**1.006**
DVFS Scen. 4	0.153	0.299	0.651	0.833	0.999

The configuration ability of our approach allows for the profiling of different voltage regulators and the determination of the most apt one. Emulation profiling results have shown that the voltage regulator in DVFS scenario 3 is superior for an energy-driven optimization strategy, while the system configuration in DVFS scenario 4 is more reliable to not violate task deadlines. The early design phase applicability of our approach offers much room for designers to explore various voltage regulator configurations and take timely countermeasures if energy or timing constraints are violated.

TABLE II
ENERGY EVALUATIONS RESULT FOR DIFFERENT DVFS SCENARIOS (NORMALIZED ENERGY VALUES)

	T1	T2	T3	T4	T5	Sum
DVFS Scen. 1	0.205	0.139	0.396	0.075	0.186	1.000
DVFS Scen. 2	0.175	0.095	0.230	0.045	0.145	0.690
DVFS Scen. 3	0.177	0.106	0.248	0.046	0.143	0.720
DVFS Scen. 4	0.205	0.137	0.291	0.055	0.149	0.837

VI. CONCLUSION

Numerous works in the past have dealt with DVFS and its energy saving potential for embedded systems. However, not so much attention has been paid to the non-ideal behavior of voltage regulators. The limited number of voltage levels and the switching delay between these levels impacts on the energy efficiency of the system. In this work, we provide an emulation platform for DVFS efficiency investigations based on power estimation hardware and DVFS hardware extensions. Performance, power and energy efficiency investigations based on the emulation approach can be carried out at speedups of several orders of magnitude over power simulations. Moreover, the early design phase applicability offers more freedom to the designer to explore various voltage regulators and to determine the most suitable one before silicon of the system is available.

REFERENCES

[1] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss, "An emulation-based real-time power profiling unit for embedded software," in *SAMOS*, 2009.
[2] J. Flinn and M. Satyanarayanan, "PowerScope: a tool for profiling the energy usage of mobile applications," in *WMCSA*, 1999.
[3] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis Of Embedded Software: A First Step Towards Software Power Minimization," in *ICCAD*, 1994.
[4] M. Lajolo, A. Raghunathan, S. Dey, and L. Lavagno, "Cosimulation-based power estimation for system-on-chip design," in *IEEE Trans. on VLSI Systems*, 2002.
[5] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: a new paradigm for power estimation," in *DAC*, 2005.
[6] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *SOSP*, 2001.
[7] B. H. Calhoun and A. P. Chandrakasan, "Ultra-dynamic voltage scaling (UDVS) using sub-threshold operation and local voltage dithering," in *IEEE Journal of Solid-State Circuits*, 2006.
[8] S. Lee and T. Sakurai, "Run-time power control scheme using software feedback loop forlow-power real-time applications," in *ASP-DAC*, 2000.
[9] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi, "Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster," in *IPDPS*, 2006.
[10] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, *System-Level Design Techniques for Energy-Efficient Embedded Systems*. Kluwer Academic Publishers, 2004.
[11] T. Burd and R. Brodersen, "Design issues for dynamic voltage scaling," in *ISLPED*, 2000.
[12] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid, "Automated Power Characterization for Run-Time Power Emulation of SoC Designs," in *DSD, In press.*, 2010.
[13] A. Bogliolo, L. Benini, and G. D. Micheli, "Regression-based RTL power modeling," in *ACM Trans. on Design Automation of Electronic Systems*, vol. 5, 2000.

An Automated Framework for Power-Critical Code Region Detection and Power Peak Optimization of Embedded Software

Christian Bachmann[1], Andreas Genser[1],
Christian Steger[1], Reinhold Weiß[1] and Josef Haid[2]

[1] Institute for Technical Informatics, Graz University of Technology, Austria
[2] Infineon Technologies Austria AG, Design Center Graz, Austria

Abstract. In power-constrained mobile systems such as RF-powered smart-cards, power consumption peaks can lead to supply voltage drops threatening the reliability of these systems. In this paper we focus on the automated detection and reduction of power consumption peaks caused by embedded software. We propose a complete framework for automatically profiling embedded software applications by means of the power emulation technique and for identifying the power-critical software source code regions causing power peaks. Depending on the power management features available on the given device, an optimization strategy is chosen and automatically applied to the source code. In comparison to the manual optimization of power peaks, the automatic approach decreases the execution time overhead while only slightly increasing the required code size.

1 Introduction

The power consumption of embedded systems is increasingly dependent on software applications determining the utilization of system components and peripherals. Furthermore, the embedded software actuates power management features such as voltage and frequency scaling as well as dedicated sleep or hibernation states. Hence, software applications impact the average as well as the peak power consumption that is in turn affecting the reliability, stability and security of embedded systems. Especially for RF-powered devices such as contactless smart-cards, power peaks threaten the system reliability by impacting the power supply circuit and leading to supply voltage drops [1]. These supply voltage drops can in turn result in system resets or, even worse, in erroneous system states. Therefore, power peak reduction and elimination methods for embedded software have been proposed [2–4]. Furthermore, power peak reduction techniques have been studied for the purpose of power profile flattening in hardware implementations [5–7]. For security applications, the profile flattening resembles a countermeasure against power analysis attacks.

In this paper we propose an automated methodology for profiling a software application's power consumption and deriving a power peak optimized implementation. Based on an integrated supply voltage simulation, critical code regions are detected and optimized. While existing software optimization methods employ either instruction-level power simulators [2–4] or physical on-chip power measurements [5–7] to obtain power profiles, our approach utilizes a high-level power emulation technique previously introduced in [8]. Using this technique, cycle-accurate run-time power estimates are derived from the system-under-test's functional emulation. In comparison to measurement-based approaches, the joint functional and power emulation offers the advantage of inherent power profile to functional execution trace correspondence, i.e., a power consumption value can be determined for each executed instruction. Furthermore, the emulation is cycle-accurate while still allowing for rapid profiling of long program sequences. This constitutes an advantage over simulation-based approaches that are either lacking simulation detail and hence accuracy or simulation speed.

In contrast to hardware power profile flattening approaches, no additional on-chip measurement and control hardware is required. Furthermore, opposed to power peak reduction methods modifying intermediate language representations of the given software application [2, 3], our approach operates on and modifies the original C or assembler source code. The resulting power peak optimized source code can afterwards still be manually modified by the software engineer if required. In the context of embedded software power peak optimization, the novel contributions of this paper are as follows:

– We present a framework for detecting source code regions causing power peaks by analyzing the power consumption as well as the functional debug information obtained during software execution.
– We derive an optimization algorithm, actuating power management features for these power-critical source code regions and hence reducing the number of power peaks.
– Finally, we illustrate the feasibility of our approach on a power-constrained deep-submicron smart-card controller system.

This paper is structured as follows. In Section 2 we discuss related work on power peak optimization and power profile flattening. Section 3 presents our automated framework for power-critical code region detection and optimization. We illustrate the effectiveness of our approach in Section 4. Finally, conclusions drawn from our current work are summarized in Section 5.

2 Related Work

Due to the large influence of software on both average as well as peak power consumption of embedded systems, numerous works have studied power- and energy-aware software optimization methods. With regard to power-constrained devices, the power profile flattening and the optimization of power consumption peaks, is of increased interest. These power peaks are often caused due to the occurrence of power-critical events during software execution. Especially in battery- and RF-powered devices these peaks can severely impact the power supply circuit and can lead to supply voltage drops [1]. These supply voltage drops seriously jeopardize the stability and hence the reliability of the given system. Power profile flattening hardware implementations have been studied in the context of security-related applications. In the security domain, the reduction of profile variability is of increased interest as a countermeasure against power analysis attacks [9].

For the purpose of reliability enhancements, the reduction of power peaks has been investigated in [3] by means of a simulation-based peak elimination framework using iterative compilation. Other attempts on power peak reduction have focused on instruction reordering to minimize the switching activity due to circuit state changes [2] as well as non-functional instruction (NFI) insertion [4].

Power profile flattening in security applications, aiming at hindering power analysis attacks by means of NFI insertion, was studied in [5]. Both software and hardware implementations were shown. In [6] a current-injection-based real-time flattening method has been proposed. This approach has been extended in [7] by a voltage scaling capability for improved flattening performance.

3 Automated Power-Critical Code Region Detection and Power Peak Optimization of Embedded Software

Our automated power profiling and power-critical code region detection methodology as depicted in Figure 1 builds upon a *standard software development flow* (A) and our *run-time power profiling* approach (B). The power estimates, alongside with the functional traces are being analyzed to *detect power-critical code regions* (C). After these regions have been detected, an *optimization algorithm* is used to reduce the power consumption and hence the power peaks during these critical code regions (D).

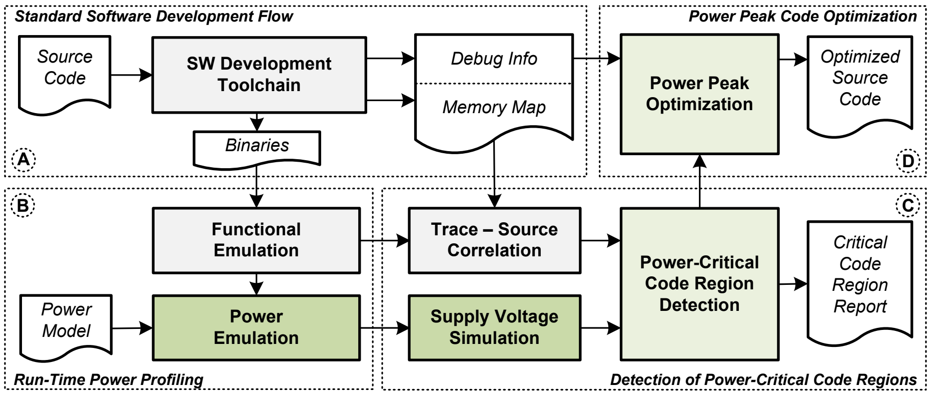

Standard Software Development Flow
Source Code
SW Development Toolchain
Debug Info
Memory Map
Binaries
A
Power Peak Code Optimization
Power Peak Optimization
Optimized Source Code
D
B
Functional Emulation
Power Model
Power Emulation
Run-Time Power Profiling
C
Trace – Source Correlation
Supply Voltage Simulation
Power-Critical Code Region Detection
Critical Code Region Report
Detection of Power-Critical Code Regions

Fig. 1. Automated flow for power profiling, power-critical code region detection and optimization

3.1 Run-Time Power Profiling Based on Power Emulation

For the purpose of detecting power-critical code regions, power profiling of the given software application has to be performed in the first place. In contrast to existing software power peak optimization approaches, we employ the power emulation technique previously introduced in [8] to obtain power profiles for the software application's execution. The principle of power emulation as depicted in Figure 2, is to augment the functionally emulated system-under-test with special power estimation hardware. This power estimation hardware monitors the state of the system and its subcomponents. Based on these state data, the power estimator derives cycle-accurate run-time power estimates according to an integrated high-level power model.

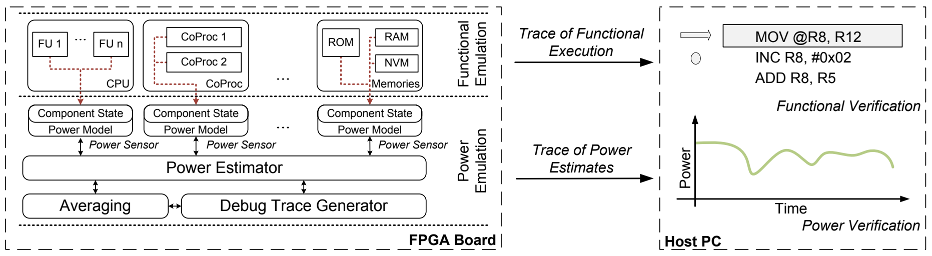


Fig. 2. Embedded software power profiling utilizing power emulation: Run-time power estimation and functional execution trace generation (adapted from [8])

As compared to low-level simulation-based power profiling, the power emulation technique largely reduces profiling time. This allows for the profiling of complex software applications and elaborate program sequences, such as the booting process of an operating system. In contrast to high-level simulators, power emulation offers the benefit of cycle-accuracy that instruction- or system-level-simulators fail to deliver. Furthermore, power emulation offers the advantage of inherent power profile to functional execution trace correspondence as compared to measurement-based approaches.

3.2 Power-Critical Code Region Detection

Our power critical code region detection approach as depicted in Figure 1 consists of multiple stages. First, the functional execution trace obtained in the joint functional and power emulation step is used to establish the source code correlation, i.e., identifying the source code region corresponding to each execution trace message. Second, using the power emulation trace as input data, a supply voltage simulation employing a numerical model of the RF-supply is performed[3]. Third, the resulting supply voltage profile is utilized to identify

[3] Due to the limited computational complexity of the numerical RF-supply model, a simulation-based implementation is adequate.

power peaks leading to critical voltage drops and finding the source code regions causing these drops.

Figure 3 depicts the inductively coupled power supply of a contact-less smart-card device. The impact of power peaks on the supply voltage level, however, is dependent on the duration, power level and rate of these peaks as shown in Figure 4. We define power-critical source code regions as parts of an embedded software application resulting in power peaks that lead to supply voltage drops below a critical limit. These peaks can be caused by, e.g., phases of high processor activity, a number of consecutive memory read or write accesses and co-processor as well power-intensive peripheral activity. In order to identify power peaks that actually lead to critical supply voltage drops on the given system, a supply voltage simulation based on the emulated power profile is performed.

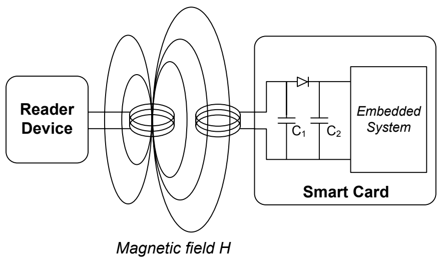


Fig. 3. Inductively coupled power supply of RF-powered smart-card embedded system (adapted from [10])

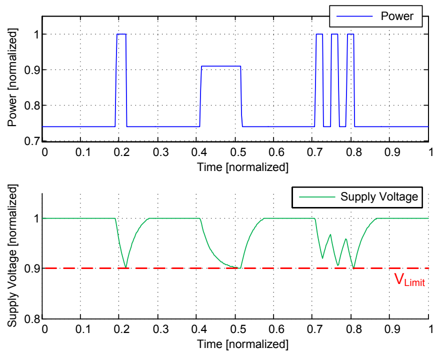


Fig. 4. Impact of different power peaks on the supply voltage (voltage drops)

3.3 Optimization of Power-Critical Source Code Regions

The subsequent power-critical code region optimization algorithm as shown in Algorithm 1 aims at applying code modifications for power peak reduction to the original C or assembler source code. Depending on the power management features available on the given system, the frequency scaling and the NFI insertion techniques are applied to these power-critical regions. Listing 1.1 illustrates the insertion of frequency scaling control instructions around the call-site[4] of a function causing power peaks, whereas Listing 1.2 shows the use of NFI insertion within a loop causing short power peaks.

The algorithm operates in three major stages: (1) The power-critical code regions for each function are determined. If a large part of a function constitutes the power-critical code region, the algorithm chooses to optimize the entire function.

[4] The source code line calling a particular function.

```
start_f_scaling();

power_critical_function();

stop_f_scaling();
```

Listing 1.1. f-scaling example

```
while(loop_condition)
{
   short_loop_instruction;
   nop(); //NFI
}
```

Listing 1.2. NFI insertion example

In this case the call-sites of the function are searched and marked for modification instead of the function itself. (2) Consecutive source code lines marked for modification are grouped into modification clusters. For each of those clusters, the algorithm chooses an optimization strategy based on the cluster's number of power peaks and their respective duration: Short power peaks are likely to be resolved by NFI insertion, longer power peaks or longer groups of peaks can be reduced by applying frequency scaling. (3) Each of the found source code clusters is then modified in the chosen way and the modified code is written back to the source files.

Algorithm 1: Power-Critical Source Code Region Optimization

```
Input: Set of application source code $S$, List of power-critical code regions $L$,
       Threshold of max. percentage of power-critical lines per function $Th_{clpf}$,
       Threshold of f-scaling time penalty $Th_{f-scale}$
Output: Set of optimized application source code $S_o$
Step 1, group by function:
List of affected source code lines $L_{sl} := \{\}$
foreach Function $f$ in $S$ do
    Find source code lines of $f$ in $L$
    if Found source code lines $> 0$ then
        Calculate percentage of power-critical code region in function
        if Percentage $> Th_{clpf}$ then
            Find call-sites of function $f$, add source code lines of call-sites to $L_{sl}$
        else
            Add source code lines to $L_{sl}$
Step 2, cluster lines to modify & choose optimization strategy:
$L_{slc}$ := Cluster consecutive source code lines in $L_{sl}$
foreach Source code cluster $C$ in $L_{slc}$ do
    if Duration $C > Th_{f-scale}$ then
        Mark cluster $C$ for f-scaling
    else
        Mark cluster $C$ for NFI insertion
Step 3, perform modification:
$S_o := S$
foreach Source code cluster $C$ in $L_{slc}$ do
    Modify $S_o$ by inserting selected optimization instructions
```

4 Experimental Results

For evaluating our framework, a smart-card microcontroller test-system supplied by our industrial partner was employed. For different benchmarking applications, power profiles were recorded using the power emulation technique. Afterwards, these benchmarks were optimized both in a manual as well as in an automated way utilizing the presented framework. This allows for evaluating the effectiveness of our method.

4.1 Test System for Power Peak Optimization

The used smart-card microcontroller test system consists of a 16-bit pipelined cache architecture. It comprises volatile and non-volatile memories as well as a number of peripherals, e.g., cryptographic coprocessors, timers, and random number generators. The system has been augmented with a power emulation unit as depicted in Figure 5 to allow for the generation of run-time power estimates.

For detecting power peaks leading to problematic supply voltage drops, we have implemented an RF power supply equivalent circuit model as proposed in [1] and depicted in Figure 6. Based on power consumption changes in the microcontroller test-system, the load current $i_l(t)$ changes and affects the load voltage $v_l(t)$. In phases of high power consumption and thus high load currents when the required load current is higher than the supplied source current $i_s(t)$, the energy storage capacitor delivers the missing fraction $i_c(t)$. However, for longer power peaks or a longer series of short power peaks, the capacitor fails to deliver the required current resulting in a critical supply voltage drop.

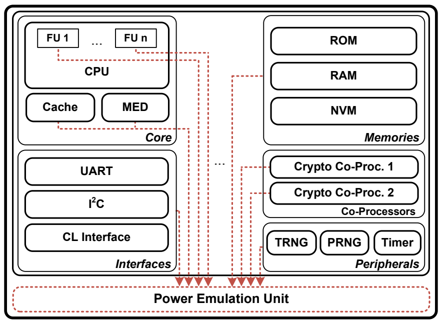


Fig. 5. 16-bit smart-card microcontroller test system augmented by power emulation unit (adapted from [11])

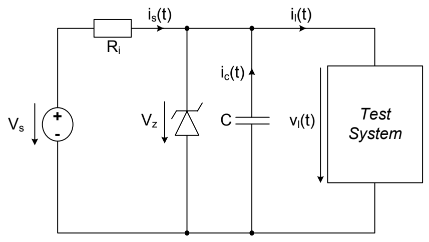


Fig. 6. Equivalent circuit of the RF power supply of the test system (adapted from [1])

4.2 Comparison of Original and Optimized Power Consumption and Supply Voltage Profiles

We illustrate the optimization result by comparing the power consumption and the respective supply voltage profiles of a given software application. Figure 7 resembles the results obtained during profiling of the original application. After the power-critical code region detection and optimization, the power profiling and supply voltage simulation was repeated yielding the profiles depicted in Figure 8.

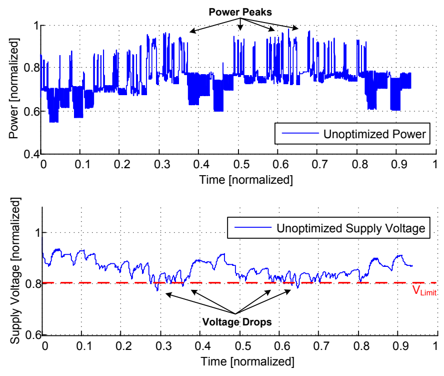


Fig. 7. Unoptimized power consumption and resulting supply voltage profiles of authentication benchmarking application[1]

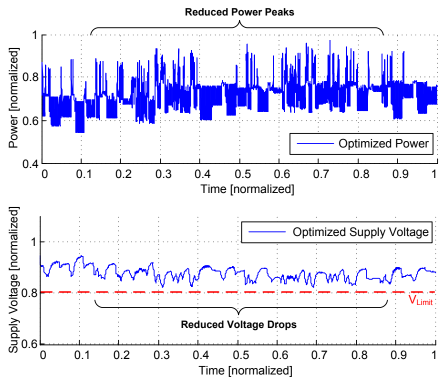


Fig. 8. Optimized power consumption and resulting supply voltage profiles of authentication benchmarking application[1]

The results illustrate how a number of power peaks result in supply voltage drops below the critical limit. By applying frequency scaling and NFI insertion to the code regions causing these peaks, their power consumption and hence their supply voltage impact can be diminished. Note that this modification, while improving system stability and reliability, comes at the cost of a slightly increased execution time. However, as illustrated in the subsequent section, the additionally required execution time is smaller for the automatically than for the manually optimized version because the frequency scaling and the NFI insertion are applied more selectively.

4.3 Impact of Power Peak Optimization on Execution Time and Code Size

We have applied the power peak optimization algorithm to various benchmarking applications in order to evaluate its impact on the execution time and the code

[1] Data normalized due to existing NDA.

size. For comparison we have also manually optimized the given benchmarking applications by applying frequency scaling to the entire benchmark. For both the manual and the automatic approach, all power peaks resulting in critical supply voltage drops have been eliminated. Figure 9 illustrates these results for two general purpose microcontroller benchmarks (Coremark [12] and Dhrystone) as well as for two domain-specific ones (Authenthication and Crypto).

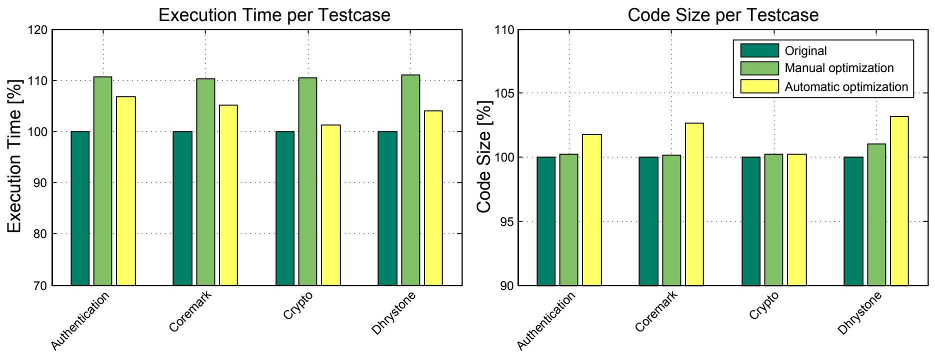


Fig. 9. Execution time and code size of original, manually as well as automatically modified benchmarks[1]

The results show that in terms of execution time the automatic approach outperforms the manual optimization due to the finer granularity of code modifications. For the manual optimization approach the execution time increases by ∼10% due to the minimally required frequency reduction of ∼10% for eliminating all critical supply voltage drops. However, for the automatic approach this increase is in the range of only 1.2% (Crypto) up to 6.8% (Authentication) depending on the number and duration of power peaks. Note that the increase in execution time also depends on the ratio of code regions affected by power peaks that need to be optimized to regions requiring no optimization.

Furthermore, we compare the increase in code size caused by the insertion of frequency scaling control instructions and NFIs. This increase is almost negligible for the manual approach (smaller than or ∼1% for all testcases). For the automatic approach, the increase is slightly higher and in the range of 0.2% (Crypto) up to 3.2% (Dhrystone).

5 Conclusions

The power consumption of embedded systems is to a large extent determined by software applications, actuating power management features as well as controlling the overall system activity. Power peaks, caused by power-critical software

[1] Data normalized due to existing NDA.

events, can seriously impact the supply voltage and lead to critical supply voltage drops. These voltage drops pose a threat to the reliability of power-constrained mobile devices such as RF-powered smart cards.

In this paper we have outlined an automated framework aimed at the power peak detection utilizing the emulation-based power profiling of given embedded software applications. By identifying the software code regions causing power peaks, the framework is able to selectively apply power reduction strategies, such as frequency scaling and non-functional instruction insertion, to the affected regions. Furthermore, we have evaluated the effectiveness of this automated power peak optimization framework on a number of benchmarking applications. For these benchmarks the inherent execution time increase is in the range of only 1.2% up to 6.8% for the automatic modifications as compared to $\sim$10% for the manual ones.

6 Acknowledgements

We would like to thank the Austrian Federal Ministry for Transport, Innovation, and Technology for providing us with funding for the POWERHOUSE project under FIT-IT contract FFG 815193, as well as our industrial partners Infineon Technologies Austria AG and Austria Card GmbH for their enduring support.

References

1. Haid, J., Kargl, W., Leutgeb, T., Scheiblhofer, D.: Power management for RF-powered vs. battery-powered devices. In: TMCS. (2005)
2. Grumer, M., Wendt, M., Steger, C., Weiss, R., Neffe, U., Muehlberger, A.: Automated software power optimization for smart card systems with focus on peak reduction. AICCSA (2007)
3. Grumer, M., Wendt, M., Lickl, S., Steger, C., Weiss, R., Neffe, U., Muehlberger, A.: Software power peak reduction on smart card systems based on iterative compiling. Emerging Directions in Embedded and Ubiquitous Computing (2007)
4. Wendt, M., Grumer, M., Steger, C., Weiss, R., Neffe, U., Muehlberger, A.: System level power profile analysis and optimization for smart cards and mobile devices. In: SAC. (2008)
5. Muresan, R., Gebotys, C.: Current flattening in software and hardware for security applications. In: CODES+ISSS. (2004)
6. Li, X., Vahedi, H., Muresan, R., Gregori, S.: An integrated current flattening module for embedded cryptosystems. In: ISCAS. (2005)
7. Vahedi, H., Muresan, R., Gregori, S.: On-chip current flattening circuit with dynamic voltage scaling. In: ISCAS. (2006)
8. Genser, A., Bachmann, C., Haid, J., Steger, C., Weiss, R.: An emulation-based real-time power profiling unit for embedded software. In: SAMOS. (2009)
9. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: CRYPTO. (1999)
10. Finkenzeller, K.: RFID Handbook. John Wiley & Sons Ltd. (2003)
11. Bachmann, C., Genser, A., Steger, C., Weiss, R., Haid, J.: Automated power characterization for run-time power emulation of soc designs. In: 13th Euromicro DSD. In press. (2010)
12. http://www.coremark.org/

Estimation-Based Run-Time Power Profile Flattening for RF-Powered Smart-Card Systems

Andreas Genser[1], Christian Bachmann[1], Christian Steger[1], Reinhold Weiss[1], Josef Haid[2]
[1]Institute for Technical Informatics, Graz University of Technology
{andreas.genser, christian.bachmann, steger, rweiss}@tugraz.at
[2]Design Center Graz, Infineon Technologies AG
josef.haid@infineon.com

***Abstract*—Power-constrained systems, such as RF-powered smart-cards are gaining increased significance in the embedded system's domain. The high susceptibility of these systems to supply voltage drops caused by power peak regions impacts on the system stability.**

In this paper we present a hardware power profile flattening approach by employing system-level DVFS adaptions coupled with a hardware power estimation architecture. The exploitation of hardware-accelerated real-time power estimation techniques replaces costly analog on-chip power measurements and enables the dynamic control of the system's power consumption in a purely digital manner. We demonstrate the effectiveness of our approach by conducting power profiling and voltage drop analysis of a deep-submicron RF-powered smart-card system.

I. Introduction

RF-powered smart-cards acquire power via an externally generated RF-field that provides only a limited amount of power dependant on the field strength and the distance between the smart-card and the RF-field generating reader device. Power peaks (i.e., regions that exceed the power limit) that are often caused by large power consuming system modules (e.g., crypographic coprocessors) can cause the supply voltage of the system to drop below a critical limit, in a way that stable system operation can not be guaranteed. A number of countermeasures to ensure reliable system operation despite occurring power peaks have been proposed. These works can be grouped in two categories: (i) *software power profile flattening* and (ii) *hardware power profile flattening*.

Software power profile flattening techniques aim at the insertion of non functional instructions (NFIs) by modifying program code sections causing power peak regions [1], [2]. An automated software power peak optimization method has been proposed in [3] by performing optimizations on the compiler level and by instruction reordering. In [4], an automated framework for power peak optimization has been presented based on NFI insertion and frequency scaling. However, the number of NFIs, their locations or the required system frequency to avoid power peaks is not known during compile time, which might require multiple optimization iterations until satisfying results are achieved.

With the advent of *hardware power profile flattening techniques*, these limitations are circumvented by utilizing dedicated flattening hardware that performs run-time NFI insertion based on power profile analysis. Power profiles are obtained by on-chip analog power measurements, while power constraints are held in registers in a digital manner. This requires D/A converters to perform the decision making whether NFI insertion is necessary [1].

A second variant of *hardware power profile flattening* is done based on *current injection methods* as proposed in [5]. Digital and analog solutions are proposed. Power profile flattening is achieved by forcing redundant switching activity or by adjusting a dedicated current sink depending on the present system's power consumption. In [6], a *current injection method* combined with the scaling of the supply voltage is presented. All these methods have in common that they require a current reserve that is dissipated during phases of low power consumption to keep the overall power consumption constant, which reduces system efficiency.

In this paper, we present a hardware power profile flattening technique based on a power estimation architecture that has been proposed in [7]. The availability of real-time power estimates in a purely digital manner replaces costly on-chip analog measurements and D/A conversions, which simplifies the overall system design.

We implement dynamic voltage and frequency scaling (DVFS) as a promising power profile flattening alternative to NFI insertion and current injection methods. By varying the system frequency and the supply voltage, the provided power can be exploited more efficiently compared to current injection methods that waste a portion of the available power. We demonstrate the effectiveness of our approach for a typical RF-powered smart-card system. The system-level implementation of the estimation-based power profile flattening technique accounts only to 2.8% to the total area and to approx. 2% to the overall power consumption of the smart-card system.

II. Hardware-Accelerated Power Estimation Prerequisites

Hardware-accelerated power estimation allows to derive cycle-accurate power consumption information in real-time in a purely digital way. In our previous work, we proposed a hardware-accelerated power estimation approach that is implemented on system-level to keep hardware and power overhead low, while still providing high estimation accuracies [7]. The basic essentials of this work are discussed within this section.

A. Power Model

Power models intended for hardware integration typically operate on high abstraction layers (e.g., system-level) and are frequently based on linear regression methods [8]. A linear regression model can be established by considering a linear combination of a number of model parameters x_i and model coefficients c_i yielding the estimate $\hat{y}$. This can be given as

$$\hat{y} = c_0 + \sum_{i=1}^{n} c_i x_i + \epsilon. \quad (1)$$

$\mathbf{x} = [x_1, x_2, ...x_n]$ gives the vector of model parameters, each of which representing system states, such as CPU operating modes, memory accesses or coprocessor activity, etc. The entirety of model coefficients can be written as $\mathbf{c} = [c_1, c_2, ...c_n]^T$. Each of the model coefficients c_i contain power consumption information corresponding to a system state x_i. c_0 resembles a special model coefficient that represents the leakage power consumption of the system. All model coefficients in $\mathbf{c}$ are determined by a power model characterization process [9]. The evaluation of (1) finally gives the power estimate $\hat{y}$, which deviates from the true power value y by the estimation error ϵ. If computed in hardware, the model can be evaluated on a clock cycle basis allowing for real-time power consumption information derivation.

B. Power Estimation Architecture

The power estimation architecture as illustrated in Fig. 1 implements the power model established according to (1) and holds power model coefficients $\mathbf{c}$ determined during the power characterization process. Power sensors implement lookup tables that map system state information x_i towards model coefficients c_i. Each of the sensors observes system states of a system component and derives its power consumption. The power accumulation unit sums up power consumption information from the power sensors and assembles cycle-accurate power samples $P(\mathbf{x})$.

Power model evaluations in hardware benefit from a huge speed up over low-level power simulations (e.g., gate-level simulations) in terms of simulation time. As reported in [7], speedups of several orders of magnitude could be achieved compared to gate-level simulations.

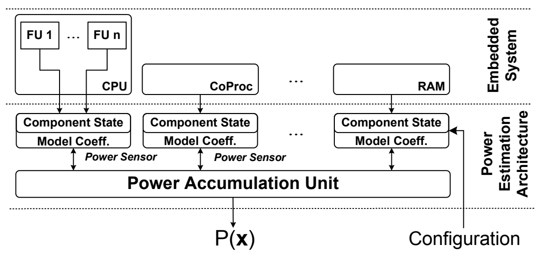


Fig. 1. Power estimation architecture, power consumption information are derived from system state information by power sensors. Obtained and modified from [7]

III. Estimation-Based Power Profile Flattening

A system overview of our proposed estimation-based hardware power profile flattening architecture is given in Fig. 2.

A. DVFS Power Scaling Extensions

The DVFS power scaling extensions block solves the issue that power consumption information $P(\mathbf{x})$ lacks the dependency from f and V_{DD}. $P(\mathbf{x})$ provides power consumption information for a reference system frequency and a reference supply voltage at which the power characterization process has been performed. DVFS power scaling extensions as depicted in Fig. 3, extend $P(\mathbf{x})$ with f and V_{DD} information yielding the real power consumption of the system $P(\mathbf{x}, f, V_{DD}^2)$. The DVFS power scaling extensions block consists of a configurable lookup table (LUT) providing V_{DD} information depending on the system frequency f. These (f, V_{DD})-pairs correspond to the supported f and V_{DD} settings of the considered system. The scaling unit conducts the multiplication of $P(\mathbf{x})$, f and V_{DD} according to (2).

$$DVFS-Scaling: P(\mathbf{x}, f, V_{DD}) = P(\mathbf{x}) \cdot f \cdot LUT^2(f) \quad (2)$$

B. Power Profile Flattening Controller

The actual power profile flattening is performed by the power profile flattening controller. The present power consumption $P_t = P(\mathbf{x}, f, V_{DD})$ is compared to given power constraints P_c. The discrepancy between P_c and P_t is given as P_ϵ, which determines the decision criteria whether the system frequency f is to be increased or decreased. By performing DVFS adaptions, the employed power profile flattening policy aims at minimizing P_ϵ according to (3).

$$FlatteningPolicy: \min\{P_\epsilon\} = \min\{\|P_t - P_c\|\} \quad (3)$$

Fig. 4 depicts the state-machine of a greedy power profile flattening policy. Starting from the $Init$-state, a state transition to f_{dec} or f_{inc} depending on the present value of f is

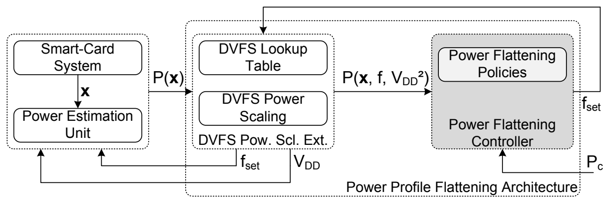


Fig. 2. Power profile flattening concept based on the power estimation architecture, the power profile flattening architecture incorporating the power profile flattening controller and DVFS power scaling extensions

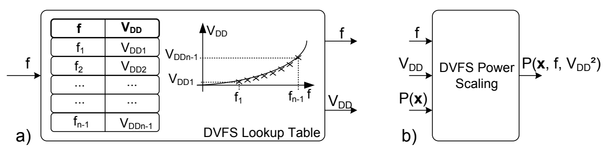


Fig. 3. DVFS power scaling extensions that incorporate a) system frequency and supply voltage dependency and b) a power scaling unit

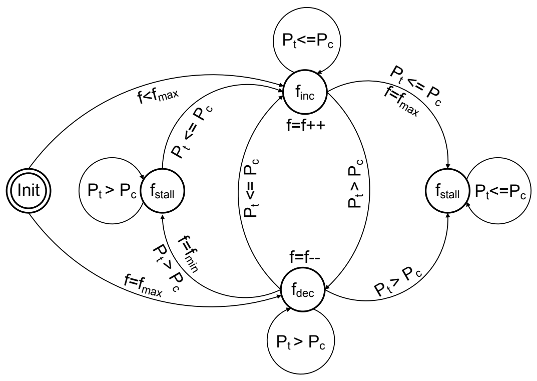


Fig. 4. System-level power profile flattening state-machine based on a greedy flattening policy

performed. In state f_{inc}, f is increased as long as the present power consumption P_t is less than or equal to P_c, else the state-machine enters state f_{dec} and f is decreased to lower the system's power consumption. If the minimum frequency f_{min} or the maximum frequency f_{max} is reached, two f_{stall} states exist to prevent f from running out of bounds.
Note that each f-adaption outputs a new frequency set value f_{set} at the power profile flattening controller causing an according lookup in the DVFS lookup table implemented in the DVFS scaling extensions block. Both, f_{set} and the corresponding supply voltage V_{DD} steer the system's operating point in a way that the power profile is flattened.

IV. Case Study: Power Profile Flattening for an RF-powered Smart-Card System

We demonstrate the results of the proposed estimation-based power profile flattening approach based on an RF-powered smart-card system that is highly susceptible to power peaks. If power peaks occur, the source voltage of the RF-energy source might drop below a critical limit, which leads to power shortages at the smart-card system.

To reproduce the behaviour of the RF-energy source, we exploit SPICE simulations of *LTSpice IV* [10] based on an RF-energy source equivalent circuit [11]. By means of a typical payment application, results of the power profile flattening approach are illustrated.

A. System Setup

Fig. 5 illustrates the coupling of the reader device to a contact-less smart-card via an RF-field. The available power at the smart-card device is dependent on the strength of the magnetic RF-field H as well as the distance between the reader and the smart-card. Moreover, Fig. 5 illustrates the structure of a typical contact-less smart-card system. It is based on a 16-bit pipelined cache architecture incorporating a memory encryption-decryption (MED) unit and volatile and non-volatile memories. Communication interfaces, such as UART, I2C or a contact-less interface are provided. Moreover, coprocessors to accelerate symmetric and asymmetric cryptographic algorithms as well as peripherals such as random number generators (RNG) or timers are built in. An analog

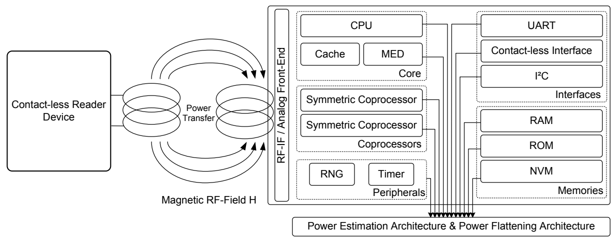


Fig. 5. Block diagram of a 16-bit contact-less smart-card system incorporating coprocessors to enhance system performance for cryptographic algorithms

front-end powers system components with energy gathered from the RF-field. System state information are provided to the power estimation architecture by routing power-relevant signals that have been determined during the power characterization process.

In order to apply DVFS adaptions for power profile flattening, the lookup table in the DVFS power scaling extensions block has to be set up based on the frequency-supply voltage relationship of the given smart-card system. The supported (f, V_{DD})-pairs are in the range of 1-38MHz and 0.5-1.7V, and frequency and supply voltage steps are 1MHz and 0.1V, respectively. The smart-card system incorporating the power estimation architecture and the power profile flattening architecture has been synthesized on an *Altera StratixII FPGA* for functional emulation.

B. Power Profile Flattening Results

A typical payment application to obtain system power profiles has been executed for the following scenarios:

- Scenario 1: Standard configuration without power profile flattening hardware
- Scenario 2: Power profile flattening configuration incorporating the power estimation and the power profile flattening architecture exploiting a *greedy flattening policy*. Power from the RF-field is constraint to $P_c = 0.4$.

Fig. 6 and 7 show power profile flattening results of Scenario 1 and 2, respectively. The power profile of an encryption and decryption procedure within the payment application is depicted for Scenario 1 in the upper subplot of Fig. 6. During

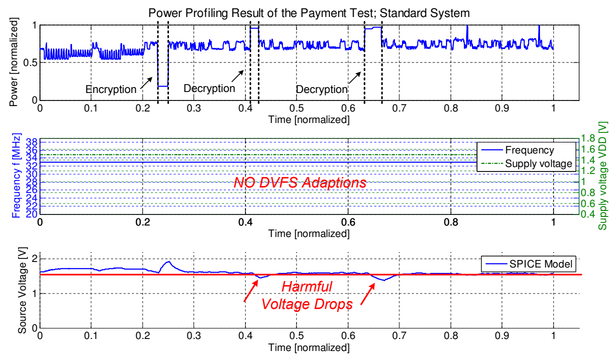


Fig. 6. Scenario 1: Payment application power profile, f and V_{DD}, as well as the behavior of the RF-energy source voltage without power profile flattening (data have been normalized for confidentiality reasons)

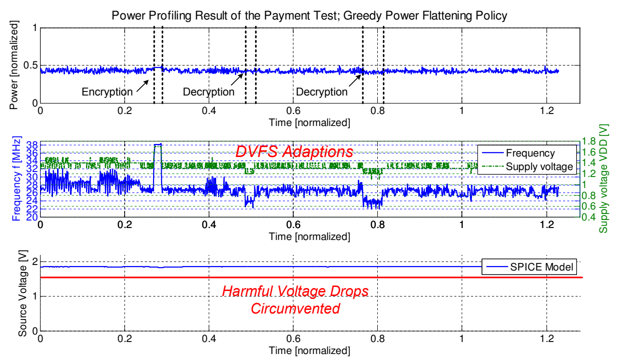


Fig. 7. Scenario 2: Payment application power profile, f and V_{DD}, as well as the behavior of the RF-energy source voltage with power profile flattening by the *greedy policy* (data have been normalized for confidentiality reasons)

decryption the CPU and the coprocessor are working simultaneously, while during encryption the CPU is put to sleep mode. The standard implementation executes the payment algorithm at a system frequency of f=33MHz and a supply voltage of V_{DD}=1.5V (see second subplot in Fig. 6). The bottom subplot of Fig. 6 illustrates the source voltage drop simulated by *LTSpice IV*. It can be clearly observed that power peaks caused by coprocessor activity make the source voltage to drop below the critical limit of V_{DD}=1.5V.
The achieved power profile flattening results of Scenario 2 are illustrated in Fig. 7. Power peaks caused by the coprocessor are flattened by adapting the system frequency f and the supply voltage V_{DD} (see middle subplot of Fig. 7). As a consequence, the source voltage stays above the critical voltage limit. Moreover, the 'negative' power peak during the encryption phase is flattened by speeding up the system to f=38MHz and V_{DD}=1.7V. This behavior exploits the available power more efficiently compared to previously proposed works based on NFI insertion or current injection methods that can not accelerate system execution.

Tab. I gives a general overview of the average to peak power (A2P) and algorithm execution time (ET) for different applications. A comparison between Scenario 1 and Scenario 2 shows that the average to peak power is reduced between 42% and 71% for the given applications, which on the other hand comes at the cost of an execution time increase of 17-23%.

TABLE I
COMPARISON OF AVERAGE TO PEAK POWER (A2P) IN % AS WELL AS EXECUTION TIME (ET) IN % FOR DIFFERENT EXECUTED ALGORITHMS (ALL VALUES NORMALIZED FOR CONFIDENTIALITY REASONS)

	Payment		Dhrystone		CPU	
	A2P	ET	A2P	ET	A2P	ET
Scen.1	100	100	100	100	100	100
Scen.2	29	123	58	122	47	117

C. Area and Power Overhead

Synthesized on an *Altera StratixII FPGA*, the entire system allocates 84785 ALUTs, while power estimation and power profile flattening hardware occupy 2416 ALUT resources. Hence, hardware extensions contribute to 2.8% to the total system area.

TABLE II
NORMALIZED AVERAGE POWER OF THE POWER PROFILE FLATTENING HARDWARE (PPF-HW) COMPARED TO THE CURRENT FLATTENING CIRCUIT PROPOSED IN [6]

	Sys.	Sys. + PPF-HW	Overhead (%)
Our architecture	100	102	**2**
Vahedi et al. [6]	100	122	**22**

[1] Power estimates have been derived from *Altera PowerPlay*. Data have been normalized for confidentiality reasons.

As depicted in Tab. II, the proposed estimation-based power profile flattening architecture accounts only to approx. 2 % to the overall power consumption of the system. In constrast, the current flattening technique proposed in [6] performs the flattening at an average power overhead of approx. 22 %.

V. CONCLUSION

RF-powered smart-cards rely on energy sources that provide power at a high variability over time. This yields a high susceptibility of these systems to supply voltage drops caused by power shortages.

The proposed estimation-based power profile flattening approach aims at minimizing power peaks by employing DVFS system adaptions based on a greedy power profile flattening policy. For a typical payment application executed on a 16-bit RF-powered smart-card system, power peak reductions are achieved in a way that harmful voltage drops can be avoided. The hardware overhead for power profile flattening hardware extensions contributes only to 2.8% to the overall system area and adds approx. 2% of power overhead. The presented work provides an effective run-time countermeasure against harmful power peaks that can significantly enhance system reliability.

REFERENCES

[1] R. Muresan and C. Gebotys. Current flattening in software and hardware for security applications. In *CODES+ISSS*, 2004.
[2] M. Wendt, M. Grumer, C. Steger, R. Weiss, U. Neffe, and A. Muehlberger. System level power profile analysis and optimization for smart cards and mobile devices. In *SAC*, 2008.
[3] M. Grumer, M. Wendt, C. Steger, R. Weiss, U. Neffe, and A. Muhlberger. Automated software power optimization for smart card systems with focus on peak reduction. In *AICCSA*, 2007.
[4] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid. An automated framework for power-critical code region detection and power peak optimization of embedded software. In *PATMOS*, 2010.
[5] X. Li, H. Vahedi, R. Muresan, and S. Gregori. An integrated current flattening module for embedded cryptosystems. In *ISCAS*, 2005.
[6] H. Vahedi, R. Muresan, and S. Gregori. On-chip current flattening circuit with dynamic voltage scaling. In *ISCAS*, 2006.
[7] A. Genser, Ch. Bachmann, J. Haid, Ch. Steger, and R. Weiss. An Emulation-Based Real-Time Power Profiling Unit for Embedded Software. In *SAMOS*, 2009.
[8] A. Bogliolo, L. Benini, and G. De Micheli. Regression-based RTL power modeling. In *ACM Trans. on Design Autom. of Elect. Sys.*, volume 5, 2000.
[9] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid. Automated power characterization for run-time power emulation of soc designs. In *DSD*, 2010.
[10] Linear Technology, LTSpice IV. (http://www.linear.com/designtools/software/), April 2010.
[11] Josef Haid, Walter Kargl, Thomas Leutgeb, and Dietmar Scheiblhofer. Power management for rf-powered vs. battery-powered devices, 2005.

Supply Voltage Emulation Platform for DVFS Voltage Drop Compensation Explorations

Andreas Genser[1], Christian Bachmann[1], Christian Steger[1] and Reinhold Weiss[1], Josef Haid[2]
[1]Institute for Technical Informatics, Graz University of Technology, Austria
[2]Infineon Technologies Austria AG, Design Center Graz, Austria
{andreas.genser, christian.bachmann, steger, rweiss}@tugraz.at
josef.haid@infineon.com

***Abstract*—The supply voltage level has emerged as an important metric alongside power consumption information to investigate system stability and reliability issues. In this paper, we propose a supply voltage emulation platform based on a power emulation approach to derive real-time power and supply voltage information from a system. Moreover, we present a dynamic voltage and frequency scaling (DVFS) based voltage drop compensation scheme to maintain stable system operation. The early design phase applicability of our emulation-based approach enables the investigation of the effectiveness of voltage drop compensation schemes before the final chip is available.**

I. Introduction

The high design complexity of embedded systems powered by energy-scavenging devices increases the risk of supply voltage drops due to the limited amount of power that is provided by the energy source. Heavy load changes within the system can cause the supply voltage to fall below a critical limit, which can severely affect the reliability of these systems. Voltage drop compensation (VDC) is a promising relief ensuring stable system operation.

The majority of VDC methods proposed in the past are purely simulation-based enabling early design phase applicability before the silicon implementation of the system exists [1], [2]. Nevertheless, especially power simulations of complex systems can become computationally intensive, which makes the simulation-based investigation of VDC schemes infeasible. On the other hand, silicon evaluations are fast and accurate but are not applicable before the final system is available, which leads to long redesign cycles if design errors are discovered.

In order to allow for fast and early design phase VDC investigations, in this paper we propose a supply voltage emulation platform to evaluate dynamic voltage and frequency scaling (DVFS) based VDC schemes for RF-powered smart card systems. The system is functionally emulated on an FPGA prototyping platform, while power consumption information is gained from a power emulation approach proposed in previous work [3]. Based on the provided power consumption information a supply voltage emulation approach is proposed to obtain voltage information. By employing DVFS adaptions, the power consumption of the system is influenced in a way that harmful voltage drops can be minimized.

II. Power Emulation Background

In contrast to simulation-based power estimation, *hardware-accelerated power estimation* delivers cycle accurate power consumption information by evaluating power models in real-time in a purely digital manner. We have implemented a system-level hardware-accelerated power estimation architecture in our previous work that keeps the hardware effort low while still providing reasonable estimation accuracies [3].

Linear regression methods as proposed by Bogliolo et al. in [4] form the basis for power models on a high level of abstraction, which makes them suitable for low overhead hardware integration. A linear regression model is based on the linear combination of model parameters x_i and model coefficients c_i, which gives the power estimate $P(\mathbf{x})$ as shown in Equ. 1.

$$P(\mathbf{x}) = c_0 + \sum_{i=1}^{n} c_i x_i + \epsilon \tag{1}$$

The hardware implementation of the established power model given in Equ. 1 maps each system state x_i to a power model coefficient c_i in a power estimation (PE) architecture. Power information of single system states are summed up, which finally assembles the cycle-accurate power samples $P(\mathbf{x(t)})$. The power emulation approach has been presented in previous work, where we have reported speedups of several orders of magnitude compared to gate-level power simulations [3].

III. Supply Voltage Emulation and Compensation Methodology

The principle of voltage drop compensation for an RF-powered smart card system is outlined in Fig. 1. It incorporates power estimation hardware to derive the present power consumption of the system as well as supply voltage estimation and VDC hardware to estimate the behavior of the supply voltage and to compensate supply voltage drops, respectively.

A number of i system cores feed system state information to dedicated PE architectures that derive power information $P(\mathbf{x_i}(t))$. A scaling unit performs the proper power consumption scaling incorporating frequency and supply voltage information yielding the overall power consumption $\mathbf{P}(\mathbf{x}(t), f(t), V_{DD}(t))$.

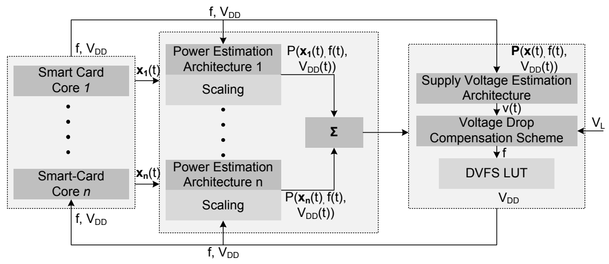


Fig. 1. System overview of an estimation-based voltage drop compensated smart card system

$\mathbf{P}(\mathbf{x}(t), f(t), V_{DD}(t))$ is then provided to the supply voltage estimation (SVE) architecture that estimates the supply voltage $v(t)$. The DVFS VDC scheme compares supply voltage estimates $\hat{v}(t)$ and a given absolute minimum supply voltage limit V_L. Based on the comparison result, a new system frequency and supply voltage set point is computed.

The SVE architecture implements an equivalent circuit of an RF energy source powering the RF-powered smart card system [5]. Equ. 2 gives the modeled supply voltage $\hat{v}(t)$ derived from the equivalent circuit and implemented in the SVE architecture [6].

$$\hat{v}(t) = \begin{cases} V_s - i(t)R_i \\ +(V_0 - V_s + i(t)R_i)e^{-\frac{T}{R_i\ C}} & \text{if } i(t) \geq i_s(t) \\ V_z & \text{else} \end{cases} \quad (2)$$

IV. Case Study: Voltage Drop Compensation Scheme Evaluation on a Dual-Core Smart Card System

Fig. 2 illustrates the emulation profiling result of a relevant section of two benchmarking algorithms, each executed on a dedicated core of the smart card system. It operates at the maximum system frequency and supply voltage settings. The superposition of high power consuming algorithm sections causes the supply voltage $v(t)$ to drop below the critical limit V_L, which would cause the system to become unstable.

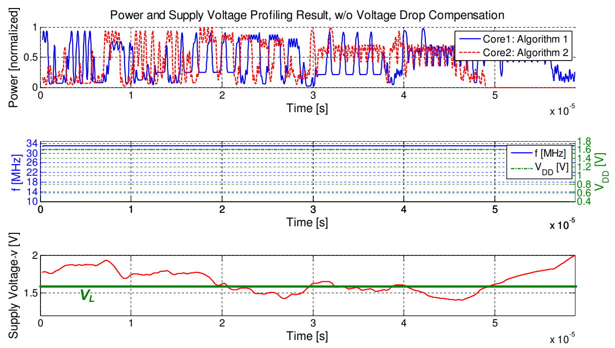


Fig. 2. Power and supply voltage emulation profiling result w/o voltage drop compensation, supply voltage is below the critical voltage V_L

Fig. 3 shows the similarly executed benchmarking algorithms with the activated DVFS VDC scheme that should prevent the system to run in an unreliable state. It can clearly be observed that system frequency and supply voltage adaptions reduce the power consumption of the system within critical algorithm sections and in turn minimize severe supply voltage drops.

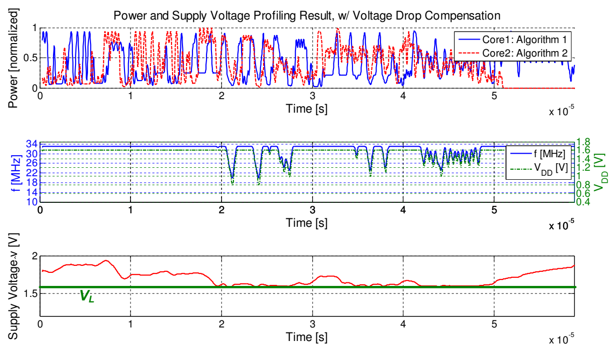


Fig. 3. Power and supply voltage profiling result, DVFS VDC scheme active, supply voltage is above the critical voltage V_L

The power emulation approach achieves relative errors below 4 % on average and variances below 3 % compared to gate-level power simulations.

The relative average error of voltage estimates delivered by the SVE architecture for a number of benchmarking algorithms can be reported as low as 2 %. The corresponding variance is below 0.2 %.

The hardware overhead introduced by the PE and SVE architecture accounts to 3.2 % and 0.8 % to the total resource demand of the system, respectively. The VDC scheme has a minor impact of 0.1 %.

V. Conclusion

Voltage drop compensation methods have gained great importance since increasingly complex systems have started to penetrate the embedded system's domain. In order to investigate their effectiveness in an early design phase, we have proposed a supply voltage emulation platform allowing for DVFS system adaptions to avoid harmful voltage drops. The presented approach enables real-time profiling of the power consumption and the supply voltage behavior based on power and supply voltage estimation methods at accuracies above 96 % and 98 %, respectively. We have shown the applicability of our emulation platform by evaluating a voltage drop compensation scheme that greatly reduces harmful voltage drops.

References

[1] C. Bachmann, A. Genser, C. Steger, R. Wei, and J. Haid, "An automated framework for power-critical code region detection and power peak optimization of embedded software," in *PATMOS*, 2010.
[2] J. Zhao, B. Datta, W. Burleson, and R. Tessier, "Thermal-aware voltage droop compensation for multi-core architectures," in *GLSVLSI*, 2010.
[3] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss, "An Emulation-Based Real-Time Power Profiling Unit for Embedded Software," in *SAMOS*, 2009.
[4] A. Bogliolo, L. Benini, and G. D. Micheli, "Regression-based RTL power modeling," in *ACM Trans. on Design Autom. of Elec. Sys.*, 2000.
[5] J. Haid, W. Kargl, T. Leutgeb, and D. Scheiblhofer, "Power management for rf-powered vs. battery-powered devices," 2005.
[6] M. Wendt, M. Grumer, C. Steger, R. Weiss, U. Neffe, and A. Muehlberger, "System level power profile analysis and optimization for smart cards and mobile devices," in *SAC*, 2008.

Bibliography

[1] International Energy Agency. Gadgets and Gigawatts - Policies for Energy Efficient Electronics. 2009.

[2] International Technology Roadmap for Semiconductors 2010 Update, System Drivers, 2010.

[3] International Technology Roadmap for Semiconductors 2010 Update, Design, 2010.

[4] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss. An Emulation-Based Real-Time Power Profiling Unit for Embedded Software. In *SAMOS 2009, IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, pages 67–73, 2009.

[5] C. Bachmann and A. Genser. POWERHOUSE - Power-Aware, Hardware-Supported Operating System and Ubiquitous Application Software Development Environment. Technical report, 2010.

[6] A. Genser, C. Bachmann, C. Steger, R. Weiss, and J. Haid. Power Emulation Based DVFS Efficiency Investigations for Embedded Systems. In *SOC 2010, IEEE International Symposium on System on Chip*, pages 173–178, 2010.

[7] A. Genser, C. Bachmann, C. Steger, R. Weiss, and J. Haid. A Hardware-Accelerated Estimation-Based Power Profiling Unit - Enabling Early Power-Aware Embedded Software Design and On-Chip Power Management. *Transactions on HiPEAC*, volume 5, 2011.

[8] A. Genser, C. Bachmann, C. Steger, R. Weiss, and J. Haid. Supply Voltage Emulation Platform for DVFS Voltage Drop Compensation Explorations. In *ISPASS 2011, IEEE International Symposium on Performance Analysis of Systems and Software*, 2011.

[9] J. Haid, W. Kargl, T. Leutgeb, and D. Scheiblhofer. Power Management for RF-Powered vs. Battery-Powered Devices. In *Workshop on Wearable and Pervasive Computing*, 2005.

[10] Aeroflex Gaisler. (`http://www.gaisler.com/`), February 2011.

[11] A. Genser, C. Bachmann, C. Steger, R. Weiss, and J. Haid. Estimation-Based Run-Time Power Profile Flattening for RF-Powered Smart Card Systems. In *APCCAS 2010, IEEE Asia Pacific Conference on Circuits and Systems*, 2010.

[12] METASEC - Mobile Energy-Efficient Trustworthy Authentication Systems with Elliptic Curve Based Security. FIT-IT Project Proposal Cooperative Research Projects, Graz University of Technology, 2011.

[13] Linear Technology, LTSpice IV. (`http://www.linear.com/designtools/software/`), February 2011.

[14] D. J. Brown and C. Reams. Toward Energy-Efficient Computing. *ACM Queue*, vol. 8, February 2010.

[15] J. Flinn and M. Satyanarayanan. PowerScope: A Tool forProfiling the Energy Usage of Mobile Applications. In *WMCSA 1999, IEEE Workshop on Mobile Computer Systems and Applications*, pages 2–10, 1999.

[16] Texas Instruments. *Analyzing Target System Energy Consumption in Code Composer Studio^{TM} IDE*, 2002.

[17] V. Tiwari, S. Malik, and A. Wolfe. Power Analysis Of Embedded Software: A First Step Towards Software Power Minimization. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, volume 2, pages 384–390, 1994.

[18] M. Sami, D. Sciuto, C. Silvano, and V. Zaccaria. An Instruction-Level Energy Model for Embedded VLIW Architectures. In *TCAD 2002, IEEE Transactions on CAD of Integrated Circuits and Systems*, volume 21, pages 998–1010, 2002.

[19] M. Lajolo, A. Raghunathan, S. Dey, and L. Lavagno. Cosimulation-Based Power Estimation for System-on-Chip Design. In *IEEE Transactions on Very Large Scale Integration Systems*, volume 10, pages 253–266, 2002.

[20] I. Lee, H. Kim, P. Yang, S. Yoo, E. Chung, K. Choi, J. Kong, and S. Eo. PowerViP: SoC Power Estimation Framework at Transaction Level. In *ASPDAC 2006, IEEE Asia and South Pacific Design Automation Conference*, pages 551–558, 2006.

[21] F. Bellosa. The Benefits of Event Driven Energy Accounting in Power-Sensitive Systems. In *SIGOPS European Workshop*, pages 37–42, 2000.

[22] R. Joseph and M. Martonosi. Run-Time Power Estimation in High Performance Microprocessors. In *ISLPED 2001, IEEE International Symposium on Low Power Electronics and Design*, pages 135–140, 2001.

[23] J. Haid, G. Kaefer, Ch. Steger, and R. Weiss. Run-Time Energy Estimation in System-on-a-Chip Designs. In *ASPDAC 2003, IEEE Asia and South Pacific Design Automation Conference*, pages 595–599, 2003.

[24] J. Coburn, S. Ravi, and A. Raghunathan. Power Emulation: A New Paradigm for Power Estimation. In *DAC 2005, Design Automation Conference*, pages 700–705, 2005.

[25] M.A. Ghodrat, K. Lahiri, and A. Raghunathan. Accelerating System-on-Chip Power Analysis Using Hybrid Power Estimation. In *DAC 2007, Design Automation Conference*, pages 883–886, 2007.

[26] A. Bhattacharjee, G. Contreras, and M. Martonosi. Full-System Chip Multiprocessor Power Evaluations Using FPGA-Based Emulation. In *ISLPED 2008, IEEE International Symposium on Low Power Electronics and Design*, pages 335–340, 2008.

[27] L. Benini and G. de Micheli. *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Academic Publishers, 1998.

[28] T. Pering, T. Burd, and R. Brodersen. The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms. In *ISPLED, IEEE International Symposium on Low Power Electronics and Design*, pages 76–81, 1998.

[29] J. Flynn and B. Waldo. Power Management in Complex SoC Designs. Technical report, Synopsys Inc. White Paper, 2005.

[30] S. Ahuja, D. A. Mathaikutty, G. Singh, J. Stetzer, S. K. Shukla, and A. Dingankar. Power Estimation Methodology for a High-Level Synthesis Framework. In *ISQED 2009, IEEE International Symposium on Quality of Electronic Design*, pages 541–546, 2009.

[31] W. L. Bircher and L. K. John. Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events. In *ISPASS 2007, IEEE International Symposium on Performance Analysis of Systems and Software*, pages 158–168, 2007.

[32] P. Pillai and K. G. Shin. Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. In *SOSP 2001, ACM Symposium on Operating Systems Principles*, pages 89–102, 2001.

[33] C. Isci, A. Buyuktosunoglu, C. Cher, P. Bose, and M. Martonosi. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. In *MICRO 39, IEEE/ACM International Symposium on Microarchitecture*, pages 347–358, 2006.

[34] B. H. Calhoun and A. P. Chandrakasan. Ultra-Dynamic Voltage scaling (UDVS) Using Sub-Threshold Ooperation and Local Voltage Dithering. In *IEEE Journal of Solid-State Circuits*, pages 238–245, 2006.

[35] S. Lee and T. Sakurai. Run-Rime Power Control Scheme Using Software Feedback Loop for Low-Power Real-Time Applications. In *ASPDAC 2000, IEEE Asia and South Pacific Design Automation Conference*, pages 381–386, 2000.

[36] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi. Profile-Based Optimization of Power Performance by Using Dynamic Voltage Scaling on a PC Cluster. In *IPDPS 2006, IEEE International Parallel and Distributed Processing Symposium*, pages 25–29, 2006.

[37] J.M. Cebrian, J.L. Aragon, J.M. Garcia, P. Petoumenos, and S. Kaxiras. Efficient Microarchitecture Policies for Accurately Adapting to Power Constraints. In *IPDPS 2009. IEEE International Symposium on Parallel and Distributed Processing*, pages 1 –12, 2009.

[38] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. HotLeakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects. Technical report, 2003.

[39] J. Sharkey, A. Buyuktosunoglu, and P. Bose. Evaluating Design Tradeoffs in On-Chip Power Management for CMPs. In *ISLPED, IEEE International Symposium on Low Power Electronics and Design*, pages 44–49, 2007.

[40] K. Meng, R. Joseph, R. P. Dick, and L. Shang. Multi-Optimization Power Management for Chip Multiprocessors. In *PACT 2008, IEEE/ACM International Conference on Parallel Architectures and Compilation Techniques*, pages 177–186, 2008.

[41] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *ISCA 2000, ACM International Symposium on Computer Architecture*, pages 83–94, 2000.

[42] R. Muresan and C. Gebotys. Current Flattening in Software and Hardware for Security Applications. In *CODES + ISSS 2004, ACM International Conference on Hardware/Software Codesign and System Synthesis*, pages 218–223, 2004.

[43] M. Wendt, M. Grumer, C. Steger, R. Weiss, U. Neffe, and A. Muehlberger. System Level Power Profile Analysis and Optimization for Smart Cards and Mobile Devices. In *SAC 2008, ACM Symposium on Applied Computing*, pages 1884–1888, 2008.

[44] M. Grumer, M. Wendt, C. Steger, R. Weiss, U. Neffe, and A. Muhlberger. Automated Software Power Optimization for Smart Card Systems with Focus on Peak Reduction. In *AICCSA 2007, ACS/IEEE International Conference on Computer Systems and Applications*, pages 506–512, 2007.

[45] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid. An Automated Framework for Power-Critical Code Region Detection and Power Peak Optimization of Embedded Software. In *PATMOS 2010, International Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 11–20, 2010.

[46] L. Xuequn, H. Vahedi, R. Muresan, and S. Gregori. An Integrated Current Flattening Module for Embedded Cryptosystems. In *ISCAS 2005, IEEE International Symposium on Circuits and Systems*, pages 436–439, 2005.

[47] H. Vahedi, R. Muresan, and S. Gregori. On-Chip Current Flattening Circuit with Dynamic Voltage Scaling. In *ISCAS 2006, IEEE International Symposium on Circuits and Systems*, pages 4277–4280, 2006.

[48] E. Grochowski, D. Ayers, and V. Tiwari. Microarchitectural Simulation and Control of di/dt-induced Power Supply Voltage Variation. In *HPCA 2002, IEEE International Symposium on High-Performance Computer Architecture*, pages 7–16, 2002.

[49] R. Joseph, D. Brooks, and M. Martonosi. Control Techniques to Eliminate Voltage Emergencies in High Performance Processors. In *HPCA 2003, IEEE International Symposium on High-Performance Computer Architecture*, pages 79–90, 2003.

[50] V.J. Reddi, M.S. Gupta, G. Holloway, Gu-Yeon Wei, M.D. Smith, and D. Brooks. Voltage Emergency Prediction: Using Signatures to Reduce Operating Margins. In *HPCA 2009, IEEE International Symposium on High Performance Computer Architecture*, pages 18–29, 2009.

[51] J. Zhao, B. Datta, W. Burleson, and R. Tessier. Thermal-Aware Voltage Droop Compensation for Multi-Core Architectures. In *GLSVLSI 2010, ACM Great Lakes Symposium on VLSI*, pages 335–340, 2010.

[52] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid. Automated Power Characterization for Run-Time Power Emulation of SoC Designs. In *DSD 2010, Euromicro Conference on Digital System Design*, pages 587–594, 2010.

[53] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner. Theoretical and Practical Limits of Dynamic Voltage Scaling. In *DAC 2004, Design Automation Conference*, pages 868–873, 2004.

[54] R. Andraka. A Survey of CORDIC Algorithms for FPGA Based Computers. In *FPGA 1998, ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 191–200, 1998.

[55] M. Mahesh and D. Nikil. eCACTI: An Enhanced Power Estimation Model for On-Chip Caches. Technical report, 2004.

[56] Magma Design Automation Inc., Blastfusion. (`http://www.magma-da.com/`), February 2011.

[57] Altera Coroporation, Stratix II. (`http://www.altera.com/`), February 2011.

[58] R. McGowen, C.A. Poirier, C. Bostak, J. Ignowski, M. Millican, W.H. Parks, and S. Naffziger. Power and Temperature Control on a 90-nm Itanium Family Processor. *IEEE Journal of Solid-State Circuits*, pages 229–237, 2006.