Master Thesis

Estimation-Based System-Level Power Management for Symmetric Multi-Core Processor Systems

Norbert Druml, BSc

Institute for Technical Informatics
Graz University of Technology
Head: O. Univ.-Prof. Dipl.-Ing. Dr. techn. Reinhold Weiß

Reviewer: Ass.Prof. Dipl.-Ing. Dr.techn Christian Steger

Advisor: Ass.Prof. Dipl.-Ing. Dr.techn Christian Steger
Dipl.-Ing. Andreas Genser

Graz, May 2011

Kurzfassung

Die Komplexität und Geschwindigkeit elektronischer Schaltkreise wächst exponentiell. Der damit einhergehende steigende Energieverbrauch ist ein Kernproblem moderner Chips, vor allem wenn diese in batteriebetriebenen, mobilen Applikation Verwendung finden. Sogenannte Energy Harvesting Anwendungen spielen eine spezielle Rolle, da elektrische Energie aus der Umgebung gewonnen wird und nur sehr limitiert zur Verfügung steht. Smart Cards sind typische Energy Harvesting Applikationen, die elektromagnetische Felder zur Energieerzeugung verwenden. Damit eine korrekte Funktionsfähigkeit des Systems gewährleistet werden kann, sollten längerandauernde Leistungsverbrauchspitzen generell vermieden werden. Andernfalls wäre ein Einbrechen der Versorgungsspannung möglich, was im schlimmsten Fall einen Neustart des Smart Card Prozessorsystems zur Folge hätte.

Das Ziel des vorliegenden Projektes ist es, ein anhand elektromagnetischer Felder betriebenes Smart Card Multiprozessorsystem mitsamt Leistungsanalyse- sowie Powermanagementeinheiten in einem FPGA zu emulieren. Eine Leistungsabschätzeinheit liefert zyklenakkurate Leistungswerte von der Zielhardware. Die Versorgungsspannung wird von einer speziellen Spannungsabschätzeinheit ermittelt. Ferner wird eine Powermanagement Einheit implementiert, die mittels dynamischer Spannungs- und Frequenzänderungen (DVFS) die Leistungsaufnahme der einzelnen Prozessoren im Notfall regeln kann. Verschiedenartige DVFS Algorithmen mit unterschiedlichsten Optimierungsstrategien werden implementiert um die Leistungsaufnahmeeigenschaften des Smart Card Systems zu verbessern und Versorgungsspannungseinbrüche zu verhindern. Die gewonnenen Ergebnisse werden abschließend präsentiert und analysiert.

Abstract

Power-aware computing addresses the problem that electronic circuits and algorithms are growing exponentially in their complexity. Thus, the power dissipation of electronic circuits also increases rapidly, which is especially problematic for mobile or battery operated applications. Given that the development of battery capacities cannot keep up with this rapid evolution and is many times slower, power-aware applications are built to use the available power smarter and more efficiently. Energy harvesting is an important application area regarding power-awareness. A smart card is a typical energy harvesting application, which relies on electrical energy gathered from an electromagnetic field. If such an RF-powered smart card device consumes too much power, its supply voltage consequently drops and the device's processor may reset.

In this project, a future RF-powered symmetric multi-core processor (SMP) system is emulated within a field programmable gate array (FPGA) board. Power analysis and power management techniques are used to improve the system's efficiency and to avoid supply voltage drops. A power estimation unit monitors the target hardware's power consumption and delivers it cycle accurately. A voltage emulation unit estimates the supply voltage based on the estimated target hardware's power consumption. A power management unit will be implemented for scaling the system's voltage and frequency parameters (DVFS) and therefore the system's power consumption is reduced if required. Several algorithms aiming at different optimization strategies are implemented and the results will then be compared and evaluated.

Acknowledgement

First and foremost, I would like to thank Ass.Prof. Dipl.-Ing. Dr.techn Christian Steger for the opportunity to make a contribution to this interesting field of research and the supervision of this master thesis. Also, I am deeply grateful for Dipl.-Ing. Andreas Genser's inspiring support and constructive discussions during the master project. Special thanks also go out to François Reney for his assiduous and thorough correction work.
My greatest gratitude belongs to my family, friends and partner for supporting, encouraging and guiding me throughout my life.

Graz, May 2011 Norbert Druml

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)

Contents

List of Figures

List of Tables

Nomenclature

ALU	Arithmetic and Logical Unit
ASIC	Application Specific Integrated Circuit
ASR	Application Specific Register
CE	Consumer Electronics
CMOS	Complementory Metal Oxide Semiconductor
CORDIC	Coordinate Rotation Digital Computer
DVFS	Dynamic Voltage and Frequency Scaling
FPGA	Field Programmable Gate Array
IC	Integrated Circuit
ICT	Information and Communication Technologies
IEA	International Energy Agency
IP	Intellectual Property
MIPS	Microprocessor without Interlocked Pipeline Stages
PEU	Power Estimation Unit
PM	Power Management
PMU	Power Management Unit
PPDU	Power Performance and Debug Unit
SMP	Symmetric Multiprocessing
SoC	System on Chip
SVEU	Supply Voltage Estimation Unit
VDCU	Voltage Drop Compensation Unit
VHDL	Very High Speed Integrated Circuit Hardware Description Language

Chapter 1

Introduction

The globally increasing demand for energy is a major challenge nowadays. According to the International Energy Agency [Age09], residential electronic devices account for about 15% of the global energy consumption and this consumption is about to increase in the future. Figure 1.1 illustrates the power consumption trend of information and communication technologies (ICT) as well as consumer electronics (CE).

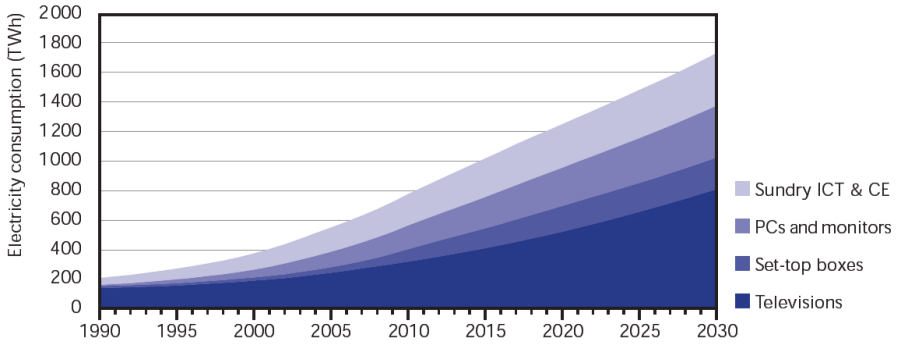

Figure 1.1: Electricity Consumption of ICT and CE [Age09]

The International Technology Roadmap for Semiconductors (ITRS) presented a similar view regarding System on Chips (SoC) in portable applications. According to this study, the complexity of SoCs will increase exponentially in the coming years in order to cope with upcoming application and computation challenges. As a result, the power consumption of these chips will increase correspondingly very quickly. Figure 1.2 depicts these trends.

Jonathan G. Koomey published in 2007 an article about the total power consumption of all data centers around the world [Koo07]. According to this study, the amount of power, that data centers are consuming, has doubled between the years 2000 and 2005. Fourteen 1000 MW power plants are needed by IT servers and their infrastructure (e.g., cooling system) worldwide. Intel published an article [Int09] as well as a software tool [Int11] that exemplify the enormous environmental and ecological potential when old server systems are replaced by newer ones.

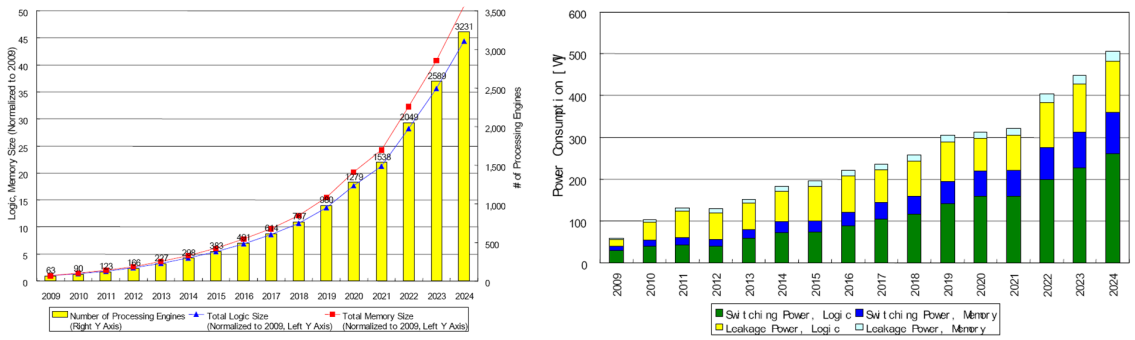

Figure 1.2: System On Chip Complexity and Power Consumption Trends [ITR11]

For a scenario where an about five year old server system consisting of 100 Dual Core Xeon 5160 processor is replaced by the newest generation of Xeon processors (E7 series), the following savings and improvements are gained:

- A 16% performance increase while simultaneously reducing the processor count by 93 units.
- 80% of electrical energy and correspondingly 84% of CO2 emissions are saved within one year.
- The capital investment of about 303.000$ is amortized after 28 month.

Figure 1.3 summarizes the improvements gained by a server system upgrade.

A chip's power consumption also influences other aspects: The higher the dissipated power, the higher the temperature of the chip. According to Scott et al. [SK94] a chip running at high temperature is prone to errors like silicon interconnect fatigue, electrical parameter shift, package related failure, junction fatigue, etc.

Power Consumption in Mobile and Energy Harvesting Applications

Power consumption plays a very important role when mobile devices (e.g., mobile phones) are used. The higher the power consumption, the shorter the lifetime of battery operated applications. Some smart card applications rely on electrical energy gathered from the environment, such as solar power or a fluctuating magnetic field, by using passive power sources (e.g., photovoltaic cells, induction circuits). This gathered electrical energy is very limited and is buffered by capacitors. If the smart card's hardware dissipates too much electrical power at given moments, then the supply capacitors could be depleted. As a result, the hardware's supply voltage would drop too low and the functionality of the application could be compromised.

If a smart card application could be provided with information about its instantaneous power consumption and supply voltage level, it could adapt its operations according to its remaining power. For example, in case of low power emergency, the application could, in order to save power, deactivate hardware components, adapt the operation processor using a dynamic voltage and frequency scaling algorithm or switch off the clock completely.

Figure 1.3: Server Refresh Potential [Int11]

1.1 Objectives and Motivation

The main objective of this master project, which is part of the POWERHOUSE[1] project, is to develop and construct an FPGA based emulation platform for a smart card target hardware in order to evaluate and explore a smart card's power consumption and supply voltage behavior during its operation. Basically, this is accomplished by combining a smart card target design as well as power and supply voltage analysis techniques within the FPGA. The evaluation of the gathered information may then reveal power bugs in the target design. Power bugs can be, for example, severe supply voltage drops or too high power consumptions. The big advantage of this approach is the possibility to detect and correct such power bugs in early design stage, long before a chip's tape-out.

In a second step, the emulation platform is enhanced with power management techniques. The instantaneous power and supply voltage values are continuously monitored and, if a power emergency arises (e.g., very high power consumption peaks or supply voltage drops), then the frequency and voltage of the target hardware's processor cores are decreased to reduce the smart card's total power consumption. For this purpose, various dynamic voltage and frequency scaling (DVFS) algorithms are implemented which use distinct input parameters and strategies.

Finally, benchmarks are executed on the emulation platform to evaluate the effectiveness of power and supply voltage analysis implementations as well as the various DVFS

[1] „POWER-aware, Hardware-supported Operating system and Ubiquitous application Software development Environment“. Funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 815193.

policies. The gathered results while using the DVFS algorithms are then post-processed, compared and analyzed. Furthermore, it is investigated if a final smart card hardware or ASIC could be enhanced with the presented power management methods.

1.2 Structuring

This document is structured into the following chapters. Chapter 2 presents the theoretical background of this project. Topics like power and supply voltage analysis, smart card systems and specific hardware based mathematical operations are explained in detail. In Chapter 3, the hardware components used for this project, which have been developed by contributers, are presented. Chapter 4 shows the design of the emulation system and how the various integrated hardware components are interconnected. The exact implementation of the system is depicted in Chapter 5. Chapter 6 shows and discusses the results gained from various tests performed on the final power and supply voltage emulation platform. This document is concluded with a discussion on the last remaining open points in Chapter 7.

Chapter 2

Related Work

2.1 Power Analysis

According to Bellaouar [BE99], power analysis is a technique used to estimate the average power consumption of electric circuits. Since Bellaouar's publication in the year 1999, new approaches have been invented which also allow for transient power analysis. Basically, there are two methods that perform power analysis:

- Measurement based: The drawn current or the temperature of the chip is measured. The results obtained are, in general, very accurate. However, expensive measuring devices are needed for this method. Under certain circumstances, this method is impractical: the dissipated power would need to be measured for each subcomponent of a chip, which is difficult, at best, to perform.
- Estimation based: This method can be further subdivided into:
  - Simulation based: The electrical circuit is simulated using a model in software. The simulation approach may be very accurate, depending on the chosen abstraction level. If complex circuits are simulated at a low abstraction level (e.g., transistor level), the calculations involved to solve the model can increase considerably the time needed to complete the simulation.
  - Hardware accelerated based: The estimation procedure and all calculations needed for this purpose are done within an additional hardware component. This approach enables up to nearly real-time calculations.

Estimation based power analysis can be performed with many different approaches and at every abstraction level. Zaccaria et al. [ZSSS04] group them in the following way:

- At transistor abstraction level, dedicated circuits are simulated with tools like **SPICE**. Immense calculation time is needed for large circuits.
- **Probabilistic** and **event driven** simulations are done at gate level. Whilst the probabilistic is a fast method used for logic synthesis, event driven power estimation uses a logic simulator and power events to simulate the dissipated power of each logic cell.

- Several different methods have been proposed, which are based on the register transfer level. Among others, one simulation technique features **power macro models**. Macro models specify switching activities and capacitance values of dedicated components. Then, the models are evaluated by circuit simulators and co-simulators.
- At the microarchitectural level, **simulations** are faster than at register transfer level but less accurate.
- Simulations performed on **instruction-set** level assign each instruction of a processor a certain power value. Into account are taken cache misses, pipeline stalls as well as the switching between consecutive instructions.
- At system level components like CPU, hard disks, network interface, etc are analyzed. **State machines** are used in recent approaches. Each state is a certain power and utilization value assigned. The total system's power consumption is then estimated by adding up the power values of all currently active states. System level approaches are the fastest but also the least accurate methods.

2.1.1 Hardware Accelerated Power Analysis

To speed up power simulations, the calculations can be implemented in hardware, thus the power analysis is done nearly in real-time. Such approaches can be done at multiple abstraction levels. Any synthesizable power model/simulation based technique may be used for this method. According to Coburn (see [CRR05] and [CRA05]) the complete target hardware is integrated in a hardware emulation environment, like a field programmable gate array (FPGA). This approach is known as „Power Emulation“ and provides the development team a big advantage: the emulation platform is already available in early design stages. Power bugs (e.g., power peaks exceeding a maximum allowed value) may therefore be found and fixed before the tape-out.

Coburn proposes a power estimation platform based on the register transfer abstraction level. Power macromodels are attached to each register transfer component of interest. A power macromodel's tasks is to compute to component's power consumption by means of observing its input and output signals. The system's total power consumption is then computed by accumulating all macromodel power values.

2.1.2 Hardware Accelerated Power Analysis Implementation

Genser et al. [GBH+09] propose a power estimation system at the system abstraction layer. The internal power model is implemented as a linear regression model (depicted by Equation (2.1)), which bases upon methods suggested by Bogliolo et al. [BBDM00].

$$\widehat{y} = \sum_{i=0}^{n-1} c_i \cdot x_i + \epsilon \tag{2.1}$$

$\mathbf{x}$ is a vector whose elements specify a certain system state (e.g., CPU running, CPU idle, etc). Every state has a model coefficient assigned to itself. A model coefficient defines how much power is dissipated while being in the corresponding system state. The model coefficients compose the vector $\mathbf{c}$. The linear combination of the model parameters $\mathbf{x}$ and

the model coefficients $\mathbf{c}$ plus an uncertainty factor ϵ (difference between the estimated and the real power value) form the estimated power value $\widehat{y}$.

After the power model has been defined, the power characterization process is performed. During this process, model parameter $\mathbf{x}$ are chosen and perturbed, the corresponding power values $\mathbf{y}$ are measured and the model coefficients $\mathbf{c}$ are calculated. The chosen model parameters influence the accuracy of the power emulation model directly. To calculate the model coefficients $\mathbf{c}$, a matrix form of Equation (2.1) is introduced by Equation (2.2).

$$\mathbf{y} = \mathbf{X} \cdot \mathbf{c} \tag{2.2}$$

The matrix $\mathbf{X}$ and the power values within the vector $\mathbf{y}$ form the training set $\mathbf{T}$, which is shown by Equation (2.3).

$$\mathbf{T} = (\mathbf{y}, \mathbf{X}) \tag{2.3}$$

Usually the training set is bigger than the number of model coefficients $\mathbf{c}$, thus there is no exact solution for $\mathbf{c}$. However, this can be solved by applying a least square fit method. Afterwards the model coefficients $\mathbf{c}$ can be calculated by solving the system of equations (2.2).

Now, the power emulation platform can be implemented in an FPGA. Therefore, the hardware under test must be available as synthesizable code. Figure 2.1 illustrates the power emulation system's architecture according to the approach of Genser [GBH+09].

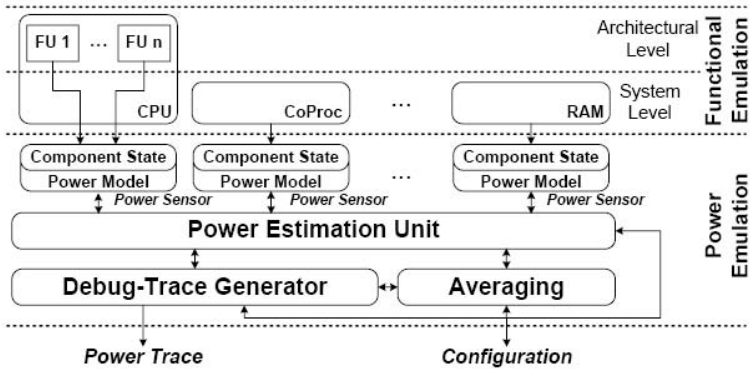

Figure 2.1: Power Emulation Harddware [GBH+09]

The power emulation system uses power sensors to retrieve the model states of each hardware component. If model states at system abstraction level are too inaccurate, then they can be also gathered from lower abstraction levels (e.g., architectural level, see Figure 2.1). The power states are then mapped to their corresponding power values. Afterwards, the power values are summed up and equal the total target system's power dissipation. Finally, a time-dependency is introduced with the help of Equation (2.4).

$$y(t) = \sum_{i=0}^{n-1} c_i \cdot x_i(t) \tag{2.4}$$

The power emulation system proposed by Genser et al. [GBH+09] delivers cycle accurate power values in real time with only about 1.5% additional hardware costs. Figure 2.2 illustrates the power profile of a payment application. A comparison is done between

a gate level simulation and the power information gained from the emulation system. A relative average error of only 8.4% can be detected. Further benefits of this power emula-

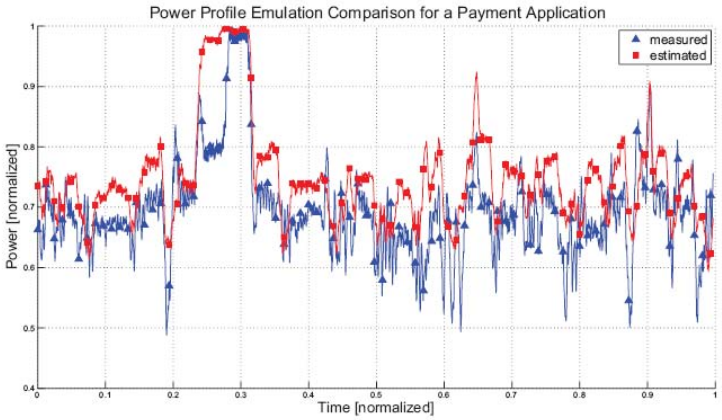

Figure 2.2: Power Profile Result Comparison [GBH+09]

tion approach are outlined by Table 2.1. Remarkable are the time differences between the emulation platform calculation time (e.g., 338.0μs) and the reference simulation time (98.3 hours). This behavior underlines the fact that power simulations of complex hardware is unfeasible.

	Time		Average Error	
Algorithm	Simulation [h]	Emulation [μs]	Power [%]	Energy [%]
ALU	4.1	70.8	7.3	6.4
CPU	0.78	31.3	-2.1	-3.2
Cache	14.0	12.4	-1.5	-2.6
RAM	2.9	56.0	-4.9	-5.5
SCP-AES128	17.2	13.5	1.2	-0.2
SCP-AES256	24.0	15.7	1.1	-0.2
SCP-DES	5.5	82.2	1.1	0.3
SCP-DDES	6.3	76.9	0.5	1.0
Payment	98.3	338.0	8.4	2.0
Dhrystone	18.1	139.0	0.4	-2.0

Table 2.1: Power Emulation Platform Results [GBH+09] with Modifications

2.2 Supply Voltage Analysis

During the past 30 years, the number of transistors on a die has increased exponentially and the power consumption of a single System On Chip has reached over 100 Watts (cf. [ITR11] and [Bor99]). Some high end processors (e.g., Intel Core i7-900 Desktop Processor Extreme Edition, see [Int10]) can be operated below a supply voltage of 1V but with a maximum current draw of up to 140A.

This ongoing increase of power consumption and decrease of supply voltage results in the following technical problems:

- During the transistor switching times high current draw variations occur. These current changes provoke a voltage across an inductance L according to Equation (2.5). The inductance value L is given by wires and pins between the power supply and the processors. This effect is known as the „di/dt problem“ [GAT02].

$$V = L \cdot di/dt \tag{2.5}$$

- High current flows between power and ground busses cause voltage variations in these busses. According to [BBH01], such voltage variations have an impact on the gate delay. Thus, the delay of the critical path changes. This behavior plays a considerable role, especially for processors with high clock frequencies.

- High frequency current variations provoke electromagnetic interferences [NIA03] [NIA04].

- The usage of a low supply voltage reduces the noise margin and therefore increases a processor's vulnerability against voltage droop effects. As a result, the following effects may arise: false triggering logic, double clocking or missing clocked pulses [SP93].

- Energy harvesting devices generate their electrical energy from the environment (e.g., electromagnetic fields). The amount of energy generated is very limited. In case of smart cards, this energy is saved in capacitors. If the processor's power consumption is too high or changes too fast, the supply voltage may drop and the processor would reset or would perform its functions incorrectly. This subject is analyzed in detail within Chapter 2.4

These problems illustrate that voltage analysis and voltage control are crucial for modern integrated circuits and power supplies. Several approaches have been presented to cope with these problems. Basically, they can be divided into design-time and run-time based solutions.

2.2.1 Design-time Based Approaches

During the design-time the di/dt problem can be reduced, for example, by shaping the electrical current with the help of a semi-asynchronous architecture [BTD+02] or by adding decoupling capacitor to reduce the inductance L [SSN02]. However, both approaches prove to be disadvantageous: In the first case the design constraints complicate the design phase, in the second case the decoupling capacitors require more die area.

Grochowski et al. [GAT02] presented a simulation based approach to control the supply voltage. Based on a current simulator, the power consumption of each clock gated processor part is calculated and summed up. Then, a detailed model of the power distribution network is generated and its impulse response computed. The supply voltage is finally calculated by performing a convolution of the summed up current and the power distribution networks impulse response. A voltage control mechanism is finally achieved

by a feedback loop (see Figure 2.4). In case the threshold comparators recognize a voltage violation, specific clock gated processor components are switching on/off. In emergency situations even the main clock is deactivated for a short amount of time.

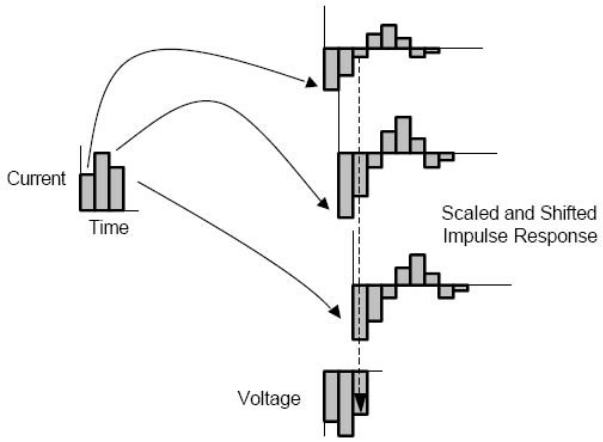

Figure 2.3: Voltage Computation by Means of Convolution Calculation [GAT02]

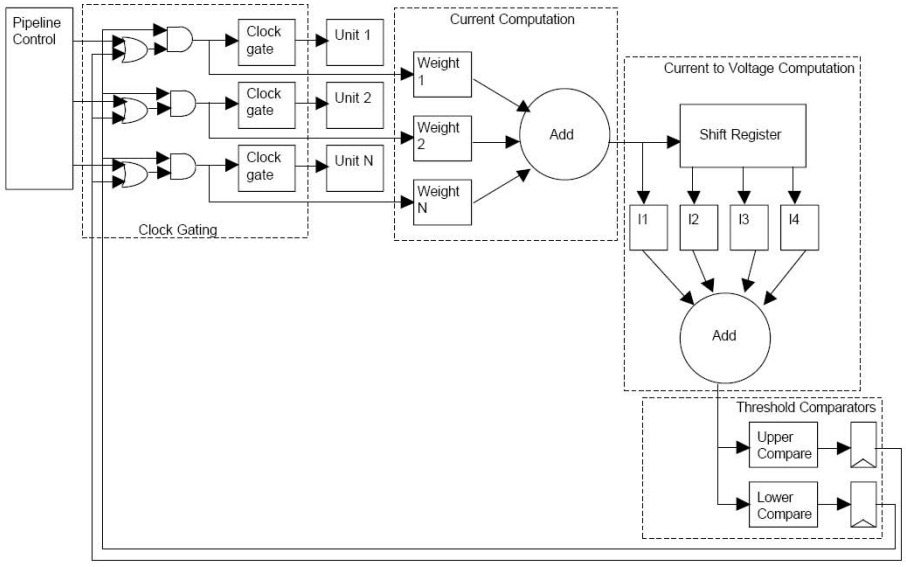

Figure 2.4: Voltage Control Mechanism by [GAT02]

In general, design-time based solutions need a very pessimistic design and detailed processor and power network models [SSN02]. Furthermore, very time intense simulations must be performed.

2.2.2 Run-time Based Approaches

There are several methods to monitor and control the supply voltage during run-time.

On-die circuits are used by [HNB08] to detect voltage drops and inject up to 100mA into the specific node. The disadvantages of this approach are the need for additional

die area for the circuit, up to 250mW of additional power consumption and a secondary supply voltage which is higher than the chip voltage to provide the injection-current.

Analog-to-digital converts [ASH05] and **voltage comparators** [NIA04] can be used to measure the supply voltage and detect drops. The sensor delay can be noted as a drawback of these methods and can limit their effectiveness.

All sensor and circuit based solutions have in common that they are available only at a late stage within an IC development process.

Shift registers are used by [GAT02] to delay clock gated processor components. As a result, there is a limited possibility to switch on/off several processor components simultaneously. This architectural modification reduces the voltage noise which is generated by high current changes.

Grochowski et al. [GAT02] implement their simulation approach (explained in Chapter 2.2.1) in real hardware. A **current estimation unit** is used instead of the current simulator. This unit monitors the clock gating signals and estimates the power consumption of the currently active processor components. Major disadvantages of this approach are the convolution engine's high computation complexity and time delay.

A **predictive approach** is proposed by [RGH+09]. Signatures of the running program are analyzed. A signature consists of program path sequences and micro architectural events (e.g., cache misses, pipeline stalls, etc). In case a signature matches an emergency pattern the processor is throttled. The accuracy of detected emergencies is above 90% and varies depending on the test programs, the size of pattern tables, etc. The immense implementation effort can be noted as a major drawback of this method.

Zhao et al. [ZDBT10] demonstrate the influence of temperature on voltage drops. The higher the temperature the lower the drop. With the help of this knowledge, they implemented a thermal-aware, on signatures based predictive voltage control. The **thermal awareness** improved the system's performance by more than 5% compared to other signature based implementations.

Genser et al. propose in [GBH+11] a **voltage emulation system** which implements an estimation approach. A *power estimation unit* (explained in Chapter 2.1.2) estimates the instantaneous power consumption of the target hardware by means of its states (e.g., CPU running, reading memory, etc). The power values are then processed by a voltage estimation unit, which is based upon the model of a smart card's supply circuit (see Chapter 2.4.1). Both units and the target hardware are then integrated into an FPGA. Genser's emulation system delivers cycle accurate power and voltage values in real time and consumes only 1.5% of additional FPGA space. The error done by the estimation procedure is below 8.4%. Above all, this approach can be performed in early product design stages. Consequently, power bugs within the target hardware can be found and corrected very soon.

2.3 Dynamic Power Management

The CMOS dynamic power consumption is basically given by Equation (2.6) [KM08]. The formula says that the dissipated power is proportional to the load capacity C, the squared supply voltage V, an activity factor A and the frequency f. The load capacity C mostly depends on wire lengths within the chip. Cleverly designed architectures may reduce

the capacity value such as implementing several small processors. The activity factor A describes how often signal changes take place. A signal switching at the maximum frequency (e.g., the main clock signal) would be assigned a value of one, while other signals with lower switching frequencies would be assigned a value between zero and one.

$$P \sim C \cdot A \cdot V^2 \cdot f \tag{2.6}$$

Tiwari et al. coined the name „Dynamic Power Management“ in the year 1997 by summarizing and describing the most important former techniques to save power in integrated circuits dynamically [TDMG97]. Among others, a few very popular dynamic power management methods are presented in the following paragraphs:

An integrated circuit's clock tree represents a large amount of load. **Clock gating** specifies a method to propagate the clock signal only to chip components that really need it. This technique is a very effective way to save power (activity factor A of Equation (2.6) is affected) and can be implemented very easily at low cost (see Figure 2.5). Analysis of Pokhrel showed that 20% die area and between 34% to 43% power can be saved with the clock gating methodology [Pok07].

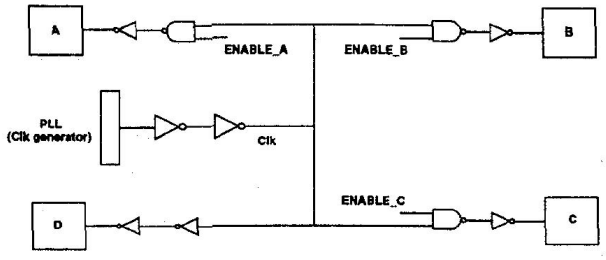

Figure 2.5: Clock Gating [TDMG97]

Guarded evaluation is a way to disable the propagation of transitions to a dedicated component. This is viable when multiple combinational blocks share, for example, the same input but only one is performing valid activities. The input for all other blocks is disabled in order to avoid unnecessary transitions that consume power. Figure 2.6 depicts a typical application for this technique within a processor's ALU.

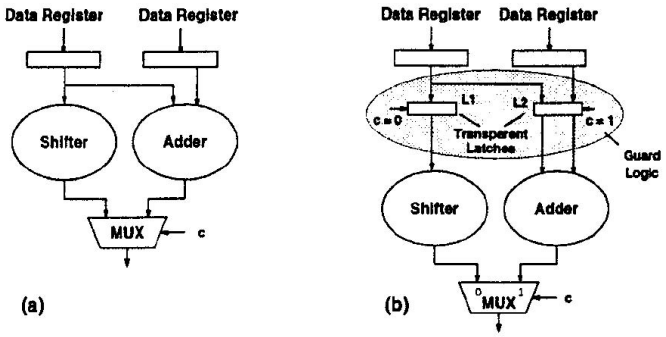

Figure 2.6: Guarded Evaluationg [TDMG97]

Bus deactivation is a method to drive a bus only when the data on the bus must be used. Thus, the power waste is reduced when bus data is not needed at all.

All the dynamic power management techniques presented here are very well known and commonly used. However, for this master thesis the „dynamic voltage and frequency scaling“ methodology is used, which has several benefits.

2.3.1 Dynamic Voltage and Frequency Scaling

DVFS is a technique which is used to modify a processor's clock frequency and supply voltage. In contrast to other dynamic power management methods, DVFS provides an elegant and very precise way to control an IC's power dissipation. According to Equation (2.6) a cubic impact can be achieved by modifying these two parameters. Any modification to the DVFS frequency parameter also affects the processor's performance linearly. Furthermore, not any arbitrary combination of voltage and frequency parameters can be selected. In order to operate a processor at a specific frequency, a dedicated minimum voltage must be supplied. Otherwise the processor would not function properly because transistors would not have enough time to switch states. According to Kaxiras and Martonosi [KM08], DVFS is utilized at three major abstraction levels:

- At system level, a whole processor is affected by the voltage and frequency modifications. Often DVFS is applied during processor idle times.
- Program level based DVFS is driven by a program's behavior. For example, memory operations with long latencies can be exploited.
- Another level below, hardware based slack is addressed directly from within the hardware.

2.3.2 Multi-Core DVFS

There are numberless approaches and algorithms available which treat DVFS implementation and utilization, particularly with regard to the field of multi-core processor systems. In the following sections, several research papers on this topic and relevant for this master thesis are described.

Globally Asynchronous Locally Synchronous Architecture

Semeraro et al. [SMB+02] propose a multi-core system featuring a globally asynchronous locally synchronous architecture (GALS, see [MHK+99]). The processor is subdivided into four clock domains (front end, integer units, floating point units and load/store units). In each domain, the frequency and voltage parameters are controlled independently. Additionally, processor components which are not used at all can be deactivated by means of clock gating. Tests with circuit simulators have shown that an energy-delay product (see [GH96] regarding the energy-delay product metric) improvement of up to 20% can be achieved compared to the system without any DVFS control. The main drawback of this approach is its complexity and hardware overhead, because communication between different voltage/clock domains must be performend via queues.

Talpes and Marculescu present a simulation based GALS design exploration framework [TM05]. With the help of this framework system, designers are able to rapidly examine how certain voltage/frequency island granularities affect the system's power consumption

and performance. Taples and Marculescu also demonstrate a processor featuring a GALS architecture, which is able to save 25%−30% of the power whilst the performance is only reduced by 5%−7%.

Per-Core versus Chip-Wide DVFS

Kim et al. explore the DVFS energy saving potential with a four core processor system [KGWB08]. Comparisons are done regarding benefits of per-core versus chip-wide DVFS. Furthermore, one slow off-chip and four very fast on-chip DVFS regulators are available for different test scenarios. The on-chip regulators allow voltage changes within nanoseconds, whilst the off-chip regulator is only able to modify the voltage within microseconds. Figure 2.7 illustrates the system architecture of three different test settings:

- Using only the off-chip regulator ≙ chip-wide DVFS
- Using the off-chip and one on-chip regulator ≙ chip-wide DVFS
- Using the off-chip and all four on-chip regulators ≙ per-core DVFS

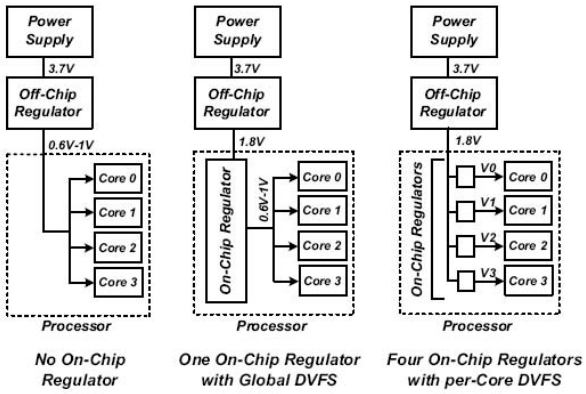

Figure 2.7: DVFS Test-System Architecture [KGWB08]

Each regulator causes electrical losses which are taken properly into account during all tests. Kim et al. used an „offline algorithm“ to control the voltage regulators. This algorithm minimizes the processor's energy consumption by exploiting the slack of memory accesses and pays attention to certain performance constraints simultaneously.

An implementation of DVFS should aim to increase the operating time of the application while keeping the performance degradation below 5%. Figure 2.8 shows the test results of the algorithm. If only one on-chip voltage regulator is used, the electrical losses downsize the theoretical power savings compared to the off-chip regulator setting. For the case where all four on-chip regulators are used, the power consumption of the complete processor can be controlled very precisely with DVFS. Thus, an improvement of up to 21% can be achieved.

DVFS Policies

Herbert and Marculescu simulate a symmetric 16-core processor system in [HM07]. Various different DVFS policies are examined and the achieved power saving is analyzed. The

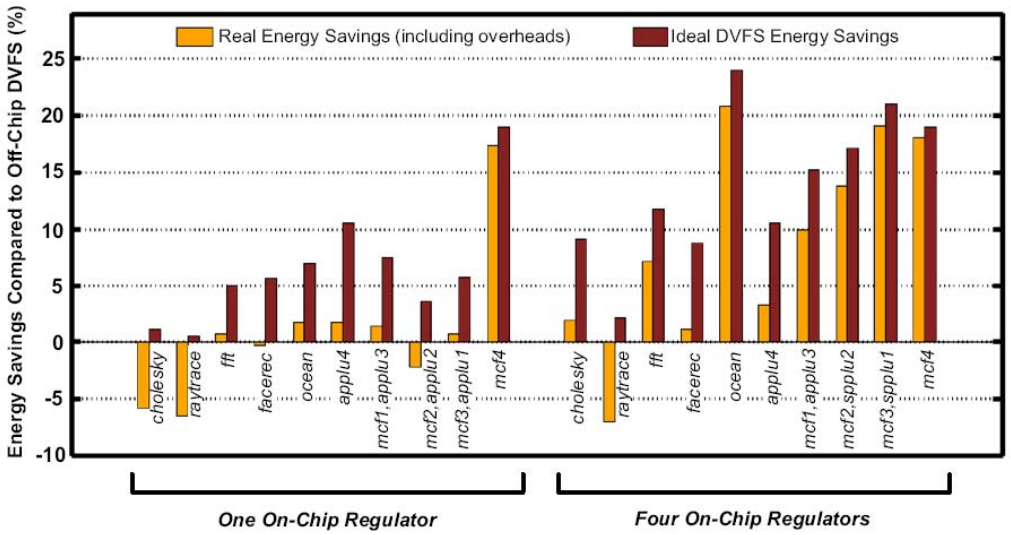

Figure 2.8: Chip-Wide versus Per-Core DVFS [KGWB08]

following algorithms are used:

- A threshold algorithm that increases voltage and frequency if an upper threshold is breached and respectively decreases voltage and frequency if a lower threshold is breached.
- A „Greedy“ algorithm that constantly searches for optimal voltage and frequency values to minimize the proportion of the energy/throughput^2 metric.
- An algorithm based on control theory, which implements a proportional-integral controller to regulate power according to processing load.

The following voltage and frequency combinations are arranged during the tests:

- Voltage and frequency parameters are equal for all cores $\hat{=}$ chip-wide DVFS.
- Groups of four cores are composed. Cores from the same group are operated with the same voltage and frequency parameters.
- Every core is run with independent parameters $\hat{=}$ per-core DVFS.

Herbert and Maculescu demonstrate that the proportion of energy/throughput^2 metric can be reduced by 38.2% when the Greedy algorithm and the highest possible voltage/frequency combination is used.

Isci et al. use a simulation approach to analyze miscellaneous DVFS policies [IBC+06]. These algorithms try to optimize the performance of a multi-core processor system while considering another constraint: the total processor's power consumption should not exceed a specified power budget. The following per-core policies are tested:

- The „Priority“ policy assigns different priorities to each processor core. The algorithm tries to run the core with the highest priority as fast as possible. When power needs to be economized, the lowest prioritized core is throttled first and the highest prioritized core is throttled last.

- „PullHiPushLo“ is a policy that tries to distribute the power consumption fairly between the cores. The core with the highest power consumption is throttled and the core with the lowest power consumption is accelerated.
- „MaxBIPS“ optimizes the system performance by adjusting the processor's instructions per second ratio. This is accomplished by a technique which predicts the cores' future utilization and power consumption.
- A chip-wide DVFS method is implemented as well. This scheme applies to all cores the same voltage and frequency parameters.

Figure 2.9 depicts the resulting performance and power consumption curves. The left image shows the performance degradation of all four policies depending on the preset power budget. The per-core DVFS policy „MaxBIPS“ performs best. In contrast, the chip-wide DVFS algorithm performs worst. The sub-figure on the right shows the power consumption curve of each policy. The dashed curve represents the preset power budget. Remarkable here is the big power consumption slack of the chip-wide policy. The reason for this is that the power consumption impact is many times greater when all cores are moved together to the next higher DVFS power/performance mode. Hence, the adjustable voltage and frequency granularity of per-core policies is far better.

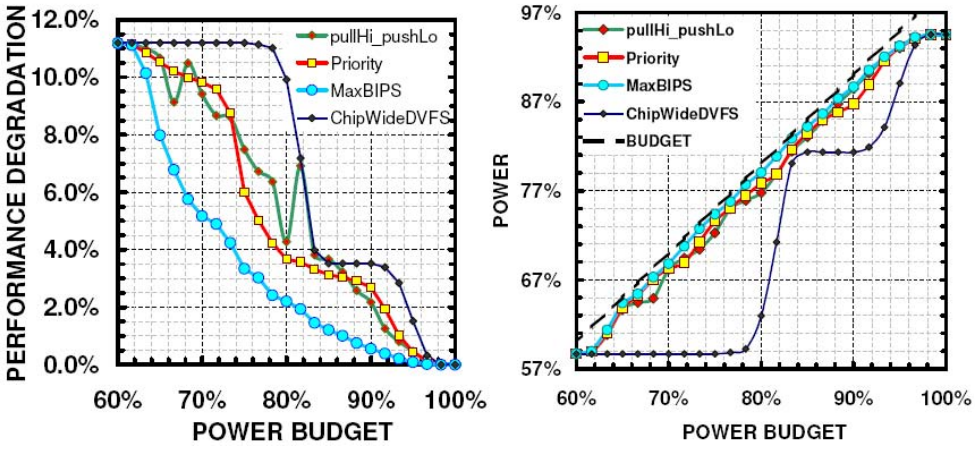

Figure 2.9: DVFS Policy Comparison [IBC+06]

Bergamaschi et al. [BHB+08] use a simulation based approach to conduct investigations regarding chip-wide and per-core DVFS policies. Basically, two algorithms are used: The MaxBIPS (which has been proposed by [IBC+06]) and a continuous power model algorithm. The special feature of the continuous power model policy is its ability to use any arbitrary frequency and voltage couple within predefined upper and lower bounds. Figure 2.10 illustrates the expected outcome. Per-core DVFS outperform chip-wide DVFS approaches.

2.4 Smart Card Specific Power Management

Haid et al. [HKLS] divide a smart card system into the following two components: A reader hardware and the smart card (transponder) itself. The reader hardware generates

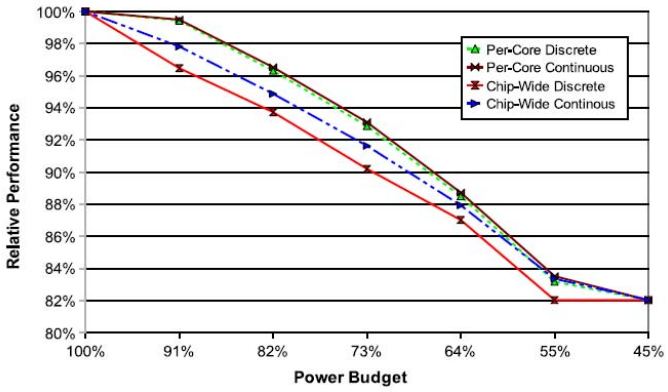

Figure 2.10: Chip-Wide versus Per-Core DVFS Policies [BHB+08]

an electromagnetic field for power supply and communication purposes. The magnetic field inducts an electrical current in the smart card. The smart card uses the electrical energy to power a small processor. Figure 2.11 illustrates this assembly. In order to ensure

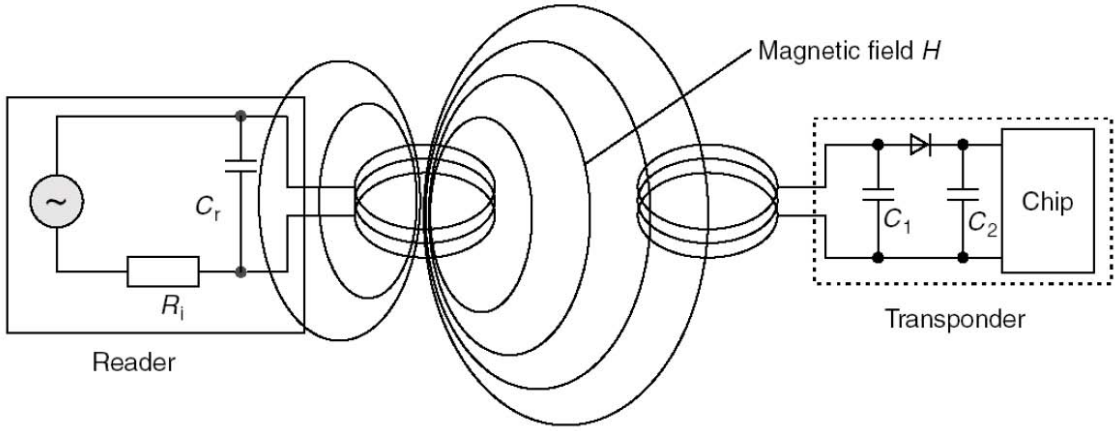

Figure 2.11: Smart Card System [Fin03]

a robust working smart card system, several aspects need to be taken into account.

Firstly, the available electrical energy is very limited. An available power budget can be calculated on the basis of the electromagnetic field strength, the antenna design, the resonance circuits and capacitors, which are used as energy storages (see Figure 2.13). The energy consumption of the smart card should never exceed this power budget to avoid a power breakdown. Attention must be paid to both, high average power consumption and high power peaks.

Secondly, the communication between reader and smart card is often modulated with the amplitude shift keying method and a modulation index of $\leq 100\%$. The load modulation is simply done by switching on/off an additional resistor. Thus, the modulation is directly influenced by any processor load change. As a result, load changes can corrupt the communication.

Figure 2.12 demonstrates the impact of different workloads on the processor's supply

voltage. Both workloads consume the same electrical energy but with different power peaks and lengths. When the processor's current rises above 3mA, both capacitors from Figure 2.11 are being depleted and the processor's supply voltage drops. It is evident: The higher the power peak, the greater the supply voltage drop. In case the supply voltage drops below a certain level, the reset logic puts the processor back into its initial state and provokes a restart.

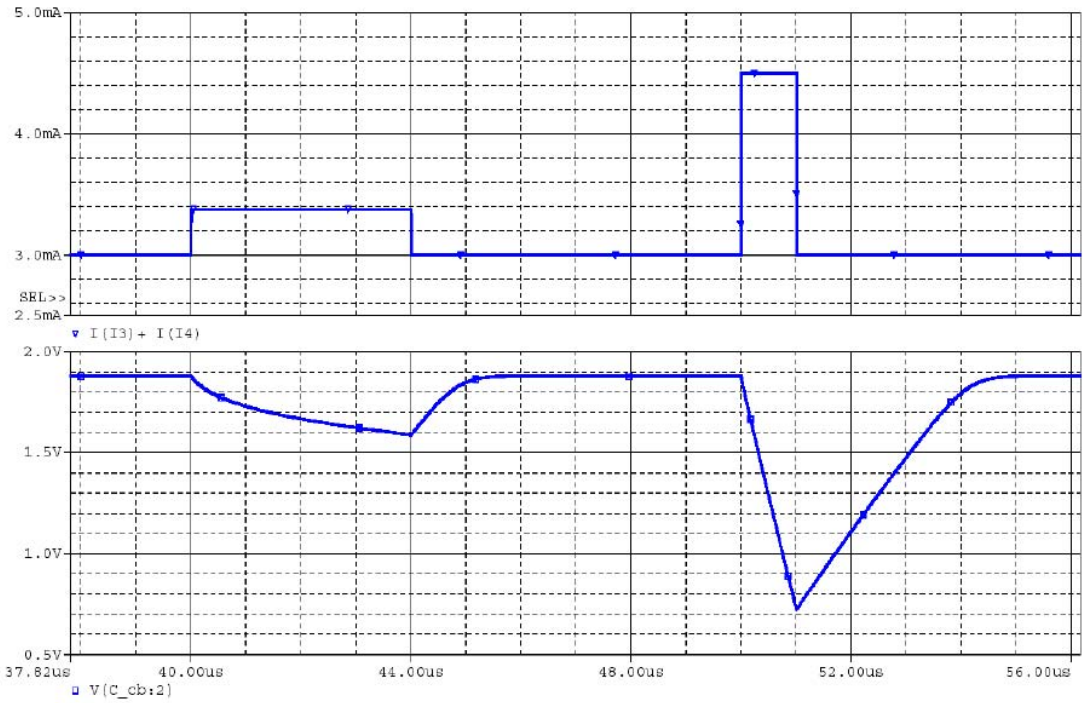

Figure 2.12: Power and Voltage Curves of a Smart Card System [HKLS]

In summary, a smart card power management must take the following crucial issues into account to guarantee a proper working system:

- The **power profile** needs to be **flattened** to minimize negative impacts on the communication.
- The power consumption should be **below** a specific **power budget** to avoid power breakdowns.
- The **supply voltage** must not drop below a certain **level** to avoid processor resets.

2.4.1 Smart Card Power Supply Model

In order to estimate the supply voltage of smart card systems, Wendt et al. propose a well fitting model [WGSW08]. In this model, the smart card is powered by a magnetic field, which is generated by a reader device. The electromagnetic field induces a sinusoidal current in the smart card. This alternating current is then transformed into a direct current. This direct current finally powers the smart card's processor.

The exact functions of the electrical components are:

- $v_s(t)$ generates an alternating voltage with a frequency of 13.56 MHz.
- R1 and R2 model the ohmic losses of the coils.
- Coil L1 generates an electromagnetic field.
- Coil L2 is used in conjunction with the magnetic field to induce a sinusoidal current.
- C1 and L2 form a series resonance circuit. To ensure a maximum power transfer between the reader and the smart card, the resonance frequency must match the frequency of the readers alternating voltage. In this example, the circuit is configured for a resonance frequency of 13.56 MHz.
- D1 to D4 are rectifying the induced current.
- C2 does smooth the rectified current and stores electrical energy.
- Zener Diode D5 is used to regulate the output voltage.
- $v(t)$ specifies the target hardware's supply voltage.

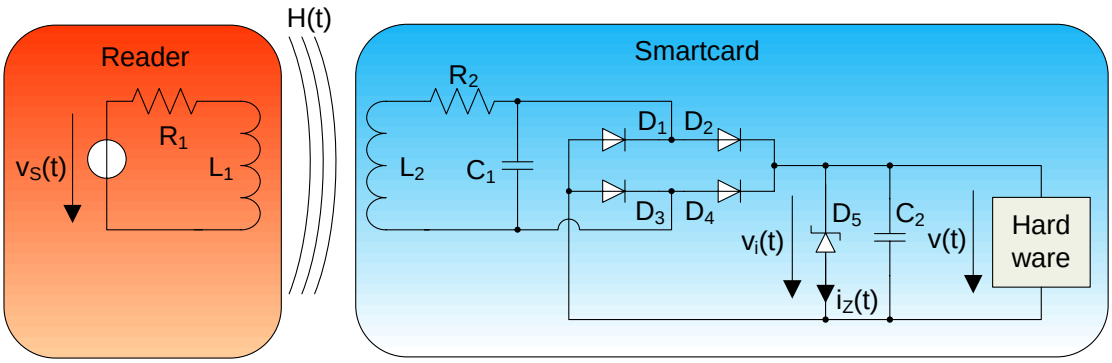

Figure 2.13: Smart Card Model

According to [WGSW08] this smart card model can be further simplified (see Figure 2.14). The generation of the electromagnetic field, the induction and the rectification of the current are replaced by a Thevenin voltage source $v_i(t)$ and a resistance R_i. $v(t)$ again defines the important target hardware's supply voltage. The target hardware consists of the processor and some clock and reset logic. $i(t)$ is the current the target hardware is consuming.

2.4.2 Smart Card Power Supply Model Analytical Analysis

The current $i(t)$ in Figure 2.14 is drawn by the target hardware and varies depending on:

- The voltage (Vdd) at which the processor is operated.
- The frequency (f) of the processor's clock.

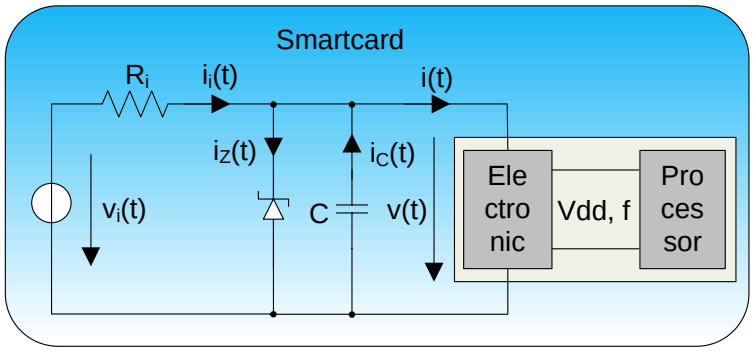

Figure 2.14: Simplified Smart Card Model

- The processor's current workload.

Looking at Figure 2.14, it can be observed, using Kirchhoff's circuit laws, that the voltage drop across the electronic load, $v(t)$, is the same as the voltage difference across the capacitor and the Zener diode. Also, this voltage is affected by the behavior of the Zener diode placed in an inverse configuration. Here is a list of all the cases describing the circuit's behavior:

- $v(t) < V_Z$: in this case, the Zener diode acts almost like an open circuit ($i_Z(t) \approx 0$) and therefore, the capacitor alone sets the value of $v(t)$. However, the capacitor voltage value is strongly affected by all electric current values present in the circuit. The possibilities are:
  - $i(t) < i_i(t)$: this describes that the current drawn in the electronic circuit is less than the supply current given by the RF receptor. In this case, $i_C(t)$ is negative, which means that the capacitor is charged and $v(t)$ increases. The supply voltage is therefore not constant.
  - $i(t) = i_i(t)$: this describes that the current drawn in the electronic circuit is equal to the supply current given by the RF receptor. In this case, $i_C(t)$ is 0, which means that $v(t)$ is constant at this instant. The supply voltage cannot be considered constant over a significant amount of time as $i(t)$ and $i_i(t)$ can fluctuate independently from one another.
  - $i(t) > i_i(t)$: this describes that the current drawn in the electronic circuit is greater than the supply current given by the RF receptor. In this case, $i_C(t)$ is positive, which means that the capacitor is discharged and $v(t)$ decreases. The supply voltage is therefore not constant.
- $v(t) = V_Z$: in this case, the diode is at the threshold between acting as an open-circuit and acting as a perfect wire. In this case, the circuit's behavior can change drastically depending on the values of all electrical current:
  - $i(t) < i_i(t)$: this describes that the current drawn in the electronic circuit is less than the supply current given by the RF receptor. In this case, $i_C(t)$ is negative, which means that the capacitor should be charged and $v(t)$ should increase. However, this is not the case because the diode will switch its operation and

act as a perfect wire. This results in a current drain of both the capacitor and the source until $v(t)$ goes back to be equal to V_Z. In practice, the supply voltage stays constant because the diode bleeds off any small voltage excesses very rapidly.

- $i(t) = i_i(t)$: this describes that the current drawn in the electronic circuit is equal to the supply current given by the RF receptor. In this case, $i_C(t)$ is 0, which means that $v(t)$ is constant and can be considered at the limit of stability.
- $i(t) > i_i(t)$: this describes that the current drawn in the electronic circuit is greater than the supply current given by the RF receptor. In this case, $i_C(t)$ is positive, which means that the capacitor is discharged and v(t) decreases. The supply voltage is therefore not constant.

According to [WGSW08] the instability of this system and the resultant changing supply voltage $v(t)$ can described by Equation (2.7).

$$v(t) = v_i(t) - R_i \cdot i(t) + e^{(-t \cdot R_i \cdot C)} \cdot (v_0 - v_i(t) + R_i \cdot i(t)) \tag{2.7}$$

This analytical model has been checked against a reference model by using a MIPS power simulator. Figure 2.15 shows both voltage curves of a md5 checksum calculation. The difference between the analytical and the reference model is less than 2%.

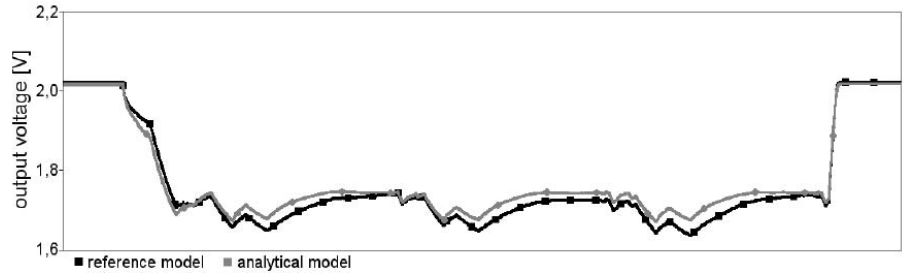

Figure 2.15: Reference and Analytical Model Comparison [WGSW08]

2.5 Exponential Function in Hardware

For this master thesis, a methodology has been developed to compute the exponential term $e^{(-t \cdot R_i \cdot C)}$ of Equation (2.7) quickly and accurately within hardware. Furthermore, the synthesized algorithm should use die area economically. Deschamps et al. [DBS06] outline the most feasible ways to implement a hardware based exponential function. Approaches like the classic TaylorMacLaurin Series or the computation via Additive Normalization are presented. However, the majority of these algorithms require a high amount of hardware resources like multiplier- and divisor-units. Therefore, they are considered to be impractical for this master thesis.

In the following section, the CORDIC algorithm is presented which is the most suitable method for this task.

2.5.1 CORDIC Approach

Jack E. Volder introduced the Coordinate Rotation Digital Computer (CORDIC) algorithm in the year 1959 [Vol59]. It is a linear convergence method. It iteratively approximates any trigonometric function by rotating vectors. The special feature of this approach is that only a few lookup tables, shift and add operations are used. Thus, CORDIC is a very practical method to be implemented in hardware. According to [Vol59], [HTHR94] and [EL04] the CORDIC algorithm is basically given by the iterative Equations (2.8), (2.9) and (2.10).

$$x_{j+1} = x_j - m\sigma_j 2^{-j} y_j \tag{2.8}$$

$$y_{j+1} = y_j + \sigma_j 2^{-j} x_j \tag{2.9}$$

$$z_{j+1} = \begin{cases} z[j] - \sigma_j \tan^{-1}(2^{-j}) & \text{if } m = 1 \\ z[j] - \sigma_j \tanh^{-1}(2^{-j}) & \text{if } m = -1 \\ z[j] - \sigma_j(2^{-j}) & \text{if } m = 0 \end{cases} \tag{2.10}$$

$$j = 0, 1, ...N - 1$$

$$\sigma_j = \begin{cases} 1 & \text{if } z[j] \geq 0 \\ -1 & \text{if } z[j] < 0 \end{cases} \tag{2.11}$$

$$K_m[j] = (1 + m2^{-2j})^{1/2} \tag{2.12}$$

j conforms the number of iteration. The rotation direction is given by σ_j, which depends on the instantaneous value of z_j. The CORDIC algorithm supports several operation modes. Each mode computes different trigonometric functions. The various possible modes are presented by Table 2.2. After N+1 computation iterations an accuracy of N-bit

Variable	Value	Mode
m	-1	Hyperbolic Coordinates
m	0	Linear Coordinates
m	1	Circular Coordinates
y	$\rightarrow 0$	Rotating
z	$\rightarrow 0$	Vectoring

Table 2.2: CORDIC Operation Modes

is achieved. Finally, the resulting x and y values must be compensated by a factor K_m, which is given by Equation (2.12). The value of this factor K_m depends on the operation mode m. Figure 2.16 exemplifies a CORDIC iterative vector rotation. The starting vector given by the values x_{in} and y_{in} is several times rotated until the final rotation θ and the values x_f and y_f are yielded.

CORDIC Hyperbolic Mode

Basically, the exponential function can be expressed by Equation (2.13)

$$e^{\theta} = \cosh(\theta) + \sinh(\theta) \tag{2.13}$$

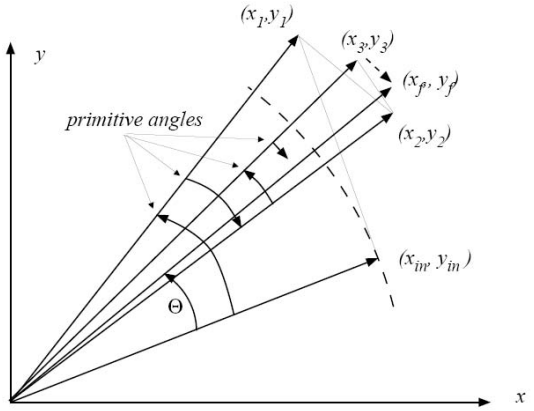

Figure 2.16: CORDIC Iterative Vector Rotation [EL04]

To achieve this calculation, the CORDIC algorithm is used in its hyperbolic - rotation mode (see [Vol59], [HTHR94] and [EL04]). Thus, the values of Table 2.3 must be applied. Running the iterative CORDIC algorithm results in the final values, which are displayed

Variable	Value	Comment
j	1	Initial value
m	-1	Hyperbolic corrdinates are used
x_{in}	1	Initial value
y_{in}	0	Initial value
z_{in}	θ	
z_f	0	$z \to 0, z$ is driven to 0
K_{-1}	≈ 0.82816	
θ_{max}	1.11817	Maximum converging input value
Repeated Iterations	3k+1	3, 14, 40,...

Table 2.3: Values for the CORDIC Hyperbolic Mode

in Equation (2.14). The current version of the algorithm does not converge yet. To resolve this issue, every $3k + 1$ iteration is carried out twice.

$$\begin{aligned} x_f &= K_{-1}(x_{in}\cosh(\theta) + y_{in}\sinh(\theta) \\ y_f &= K_{-1}(x_{in}\sinh(\theta) + y_{in}\cosh(\theta) \\ z_f &= 0 \end{aligned} \tag{2.14}$$

At last, x_f is accumulated with y_f and x_{in}, y_{in} are filled with the corresponding values from Table (2.3). Thus, Equation (2.15) displays the resulting exponential function.

$$x_f + y_f = K_{-1}(\cosh(\theta) + \sinh(\theta)) = K_{-1}e^{\theta} \tag{2.15}$$

CORDIC hardware implementations have been presented by Hu [Hu92], Andraka [And98] and Boudabous et al. [BGKM04]. Their approaches are described in detail by the authors and form the basis for this master thesis' CORDIC version of the *supply voltage estimation unit*.

Chapter 3

Design Prerequisites

The design as well as the implementation of the emulation system rely on several components, which have already been developed and used in the past:

- The GRXC3S-2000 development platform is used for rapid VHDL and FPGA development purposes. The FPGA-synthesizable processor used in this project is a LEON3 processor. This processor uses 32-bit instructions and data structures and it fully complies with the IEEE-1754 SPARC V8 specification. Also, it can be used within a multi-core system.
- To receive cycle accurate power consumption information about the smart card a *power estimation unit* is used. This unit has been developed by Genser et al. [GBH+09].
- The dynamic voltage and frequency scaling technique is implemented according to a lookup table approach, which has already been developed by A. Genser.
- Genser's *supply voltage estimation unit* is used to receive information about the target hardware's supply voltage level [GBH+11].
- A *power performance and debug unit* is integrated into the emulation platform to transmit all relevant information to a host PC for further analysis tasks. This unit has been developed by M. Lackner [Lac10].

Thanks to the participation of all contributors, the development process was focused solely on the most important issues. In this chapter, the utilized parts are presented in detail.

3.1 LEON3 Platform

The GRXC3S-2000 development board [Gai10] is being used during this master thesis. This board, illustrated in Figure 3.1, has been developed by the company PENDER ELECTRONIC DESIGN GmbH. It features a Spartan 3 Xilinx FPGA, 64 MByte SDRAM and peripheral units like an Ethernet-interface, two RS232 interfaces and several more.

The LEON3 is a synthesizable open source processor (whose VHDL source code is published under GNU GPL license) which has been developed by Aeroex Gaisler on behalf

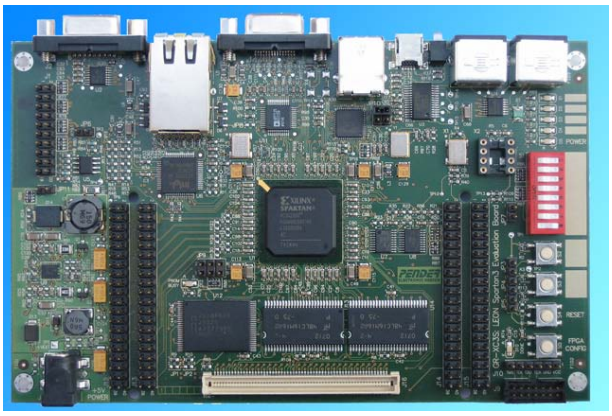

Figure 3.1: GRXC3S-2000 development board [Gai10]

of the European Space Agency. It is shipped with a comprehensive IP core library, namely the GRLIB [Gai09]. The main features of the LEON3 processor are:

- 32-bit harvard architecture processor, fully compliant with the IEEE-1754 SPARC V8 standard. Up to 16 processor cores can be used simultaneously within a multi-core environment.
- A seven stage integer pipeline and support for 15 asynchronous interrupts.
- A floating point unit and a user-defined coprocessor.
- Basic power saving methods, such as power down mode, are supported. Furthermore, clock gating techniques can be added easily.
- For inter-component communication purposes, the Advanced Microcontroller Bus Architecture (AMBA) AHB and APB bus systems are used [ARM99]. Figure 3.2 illustrates the numerous supported peripheral components.
- A fault tolerant processor version is available. This processor type is mostly used for critical applications where errors caused by „single event upset“ (a type of temporary logic error caused by non-damaging ionizing radiation) may occur.

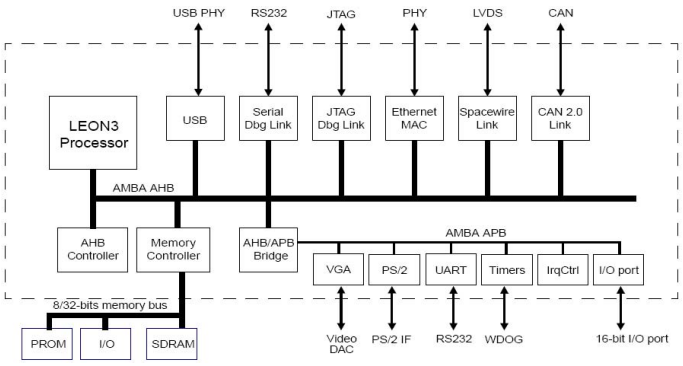

Figure 3.2: LEON3 Components and Peripherals [Gai10]

A power emulation system has been developed during a previously carried out project [Dru10]. It featured the LEON3 processor as target hardware and the *power estimation unit* which is presented in Section 3.2. This emulation system and the experience gained from the IT-Project are used as a basis for this master thesis.

3.2 Power Estimation Unit

The *power estimation unit* (PEU) is used to estimate in real time the power consumption of the target hardware. This unit has already been developed by A. Genser and C. Bachmann at the Institute for Technical Informatics in Graz [GBH+09]. The mathematical principle involved in the PEU operation is explained in Section 2.1.2 in details. Basically, small power sensors monitor in real time the processor signals. Based on this information, the corresponding processor states (e.g., memory read, memory write, ALU multiplication, etc) are derived. Then, the states are mapped against power values. The sum of all power values equals the momentary total power consumption of the processor core. The power model utilized within the PEU has been developed by Bachmann et al. [BGS+10]. It is based on gate level simulations. Figure 3.3 illustrates the principle behind the PEU.

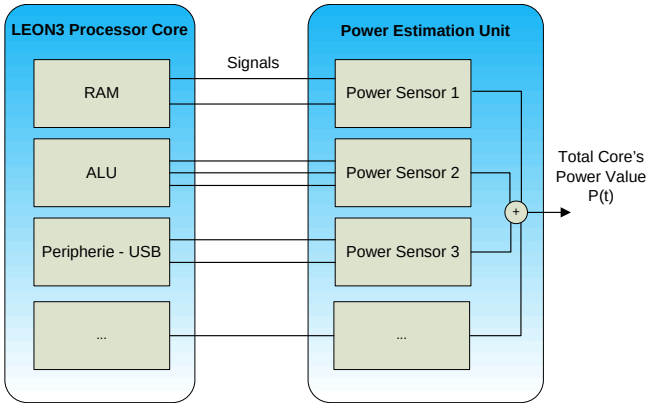

Figure 3.3: Power Estimation Unit Architecture

The target hardware features a symmetric multi-core processor system. Each processor core has one PEU assigned (cf. Figure 4.2). The sum of all power values derived by the PEUs represents then the total SMP system's power consumption. These values are then used by *supply voltage estimation unit* (SVEU), *voltage drop compensation unit* (VDCU) and *power management unit* (PMU) for further computations and DVFS control decisions.

3.3 DVFS Scaling

Due to the fact that this project features a power/supply voltage emulation and evaluation platform, dynamic voltage and frequency scaling of processor cores is only simulated and not implemented in hardware directly. This is achieved by a lookup table approach. Each possible processor clock frequency is assigned to a certain required voltage. The power values $P(t)$ which are received by the PEUs, are then scaled with the DVFS frequency

and voltage, according to Equation (3.1).

$$P(t, f, v) = P(t) \cdot f \cdot v^2 \tag{3.1}$$

Each processor core has one DVFS scaling unit assigned to it. Figure 3.4 depicts the architecture of the DVFS scaling approach.

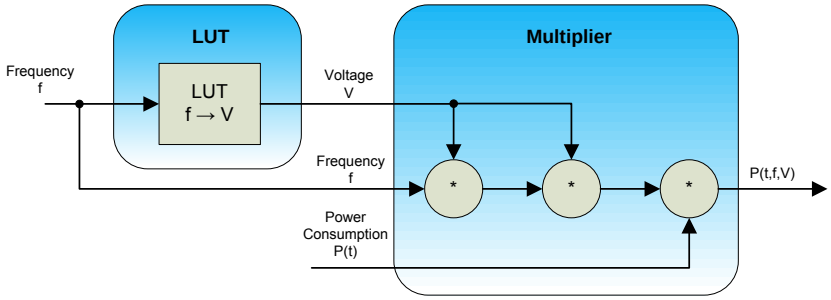

Figure 3.4: DVFS Scaling Approach

However, this simulation approach introduces a major drawback. Basically, in case two identical processors are running at different clock frequencies but are executing the same program, the faster processor finishes earlier. Note that processor cores which are running within this master thesis' proposed emulation platform execute their programs always at the same speed and finish simultaneously. The DVFS approach is only simulated with the help of a lookup table method, different processor frequencies can not be processed 100% realistically. Hence, the results gained from DVFS per-core algorithms must regarded with caution, because they operate the processor cores with different frequencies. Chip-wide DVFS algorithms are not affected by this matter, because all cores are always operated at the same clock frequency.

3.4 Supply Voltage Estimation Unit

Genser et al. proposed a *supply voltage estimation unit* (SVEU) in [GBH+11]. This unit has been developed and integrated within a power and supply voltage emulation platform for a smart card target hardware. The basic principles which have been applied for this SVEU are explained in Section 2.2 and Section 2.4.1 in detail.

PEUs are used to calculate the momentary power consumption of each processor core. The summed up power value is then passed to the SVEU, which computes the supply voltage by means of Equation 3.2.

$$v(t+1) = v_i - R_i \cdot i(t) + e^{(-t \cdot R_i \cdot C)} \cdot (v(t) - v_i + R_i \cdot i(t)) \tag{3.2}$$

$i(t)$ is given by the PEUs. To avoid the calculation of the exponential term $e^{(-t*R_i*C)}$, the parameter t is considered to be constant and is a value of 30ns assigned. Therefore the whole exponential term is constant too. v_i defines the voltage which is supplied by the magnetic field. Its steady value is set to 2.5V. The start condition for t=0 is $v(0) = v_i = 2.5V$, thus the capacitor is fully charged. Figure 3.5 illustrates the design of the supply voltage estimation approach by Genser et al. for a symmetric multi-core processor system.

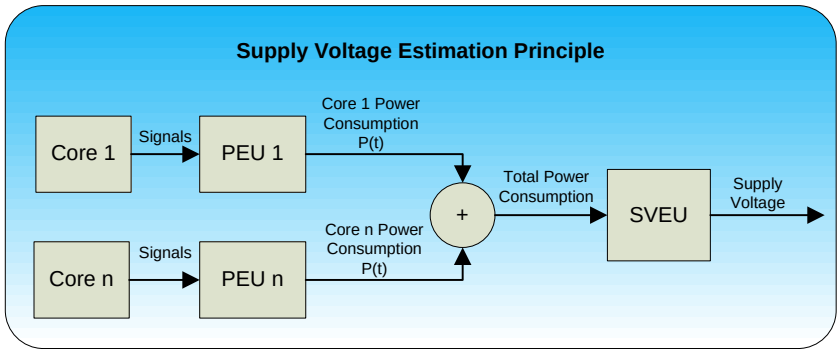

Figure 3.5: Supply Voltage Estimation Principle

3.5 Power Performance and Debug Unit

The *power performance and debug unit* (PPDU) has been designed and implemented by M. Lackner during his master thesis at the Institute for Technical Informatics in Graz [Lac10]. Its task is to transmit information from the emulation system via Ethernet to a host PC for further evaluation and analysis tasks. A JAVA based software has also been developed to receive, display and save the gathered information. The PPDU is being used during this master thesis to transfer power consumption, supply voltage level information and the processor cores' DVFS settings.

Figure 3.6 exemplifies the integration of the PPDU within a power emulation system. An AMBA APB interface is supported by the PPDU for configuration tasks. Data is sent to the unit via the designated State_Core_1 to State_Core_n signals. The data is then internally preprocessed, filled into an Ethernet frame and forwarded directly to Intel's LXT971A 100MBit Ethernet core. Finally, the Ethernet core's responsibility is to ship the Ethernet packets.

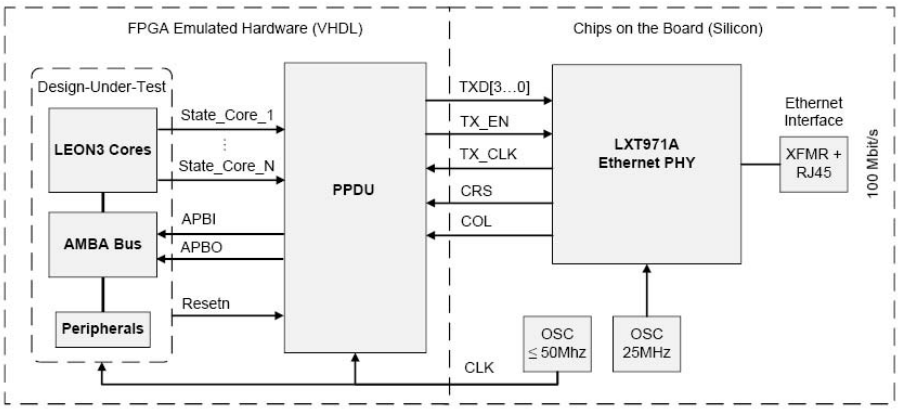

Figure 3.6: PPDU Hardware Integration [Lac10]

Figure 3.7 depicts the basic functionality of the PC based JAVA software. The JPCAP library is used to capture the incoming Ethernet packets [JPC11]. A XML configuration file is used to define the content format of the Ethernet packets and tells the subsequent pre-processing module how the data should be parsed and interpreted. The profile-output

software module then takes over the data and offers the user the possibility to whether display the data within a chart or to save it into a comma-separated file. Finally, the csv file can then be further processed by any analysis programs like MATLAB, Excel, etc.

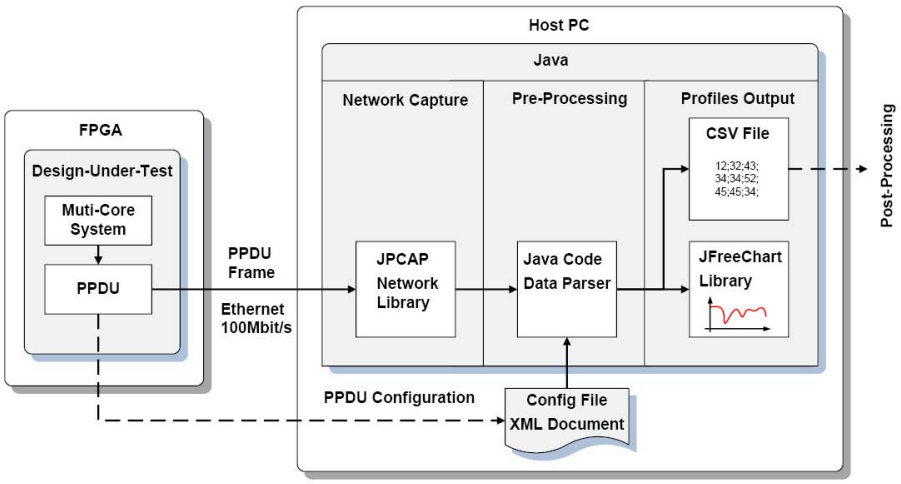

Figure 3.7: PPDU System Integration [Lac10]

According to M. Lackner, the integration of the PPDU within a two core multiprocessor system is quite costly. 19% of the FPGA area is occupied by the PPDU [Lac10]. To minimize the FPGA utilization, the PPDU is being downsized and specialized for this project. Thus, only the most relevant information is being processed and transmitted to the host PC.

The current version of the PPDU is interlinked with an 100 MBit/s Ethernet interface. The power and supply voltage emulation system runs at rounded 30 MHz and generates several bytes of analysis data every clock cycle. It is impossible to transmit all the generated data under the given circumstances. Therefore, the PPDU averages the analysis data over a certain amount of clock cycles, which introduces an unwanted but unavoidable inaccurateness. For details refer to [Lac10].

Chapter 4

Design of the Emulation Platform

This chapter presents the architectural design of the emulation platform and the basic framework of all software components. It also presents the main hardware and software components:

- A LEON3 master core is used to configure the emulation platform, which comprises the PEU, VDCU, PMU and PPDU.
- LEON3 slave cores represent the processor cores of the smart card hardware. These cores are analyzed regarding power consumption and supply voltage.
- *Power estimation units* evaluate as precisely as possible the power consumption of the slave cores.
- A *supply voltage estimation unit* calculates the supply voltage of the smart card based on the estimated power consumption.
- A *power management unit* is designed to apply DVFS policies. These policies are based on the estimated power consumption and set DVFS frequency and DVFS voltage parameters that are applied to the processor cores.
- A *supply voltage drop compensation unit* is designed to apply DVFS policies. These policies act based on the estimated supply voltage and set DVFS frequency and DVFS voltage parameters that are applied to the processor cores.
- Firmware is developed to configure the emulation system and run benchmark programs on slave cores.
- A *power performance and debug unit* is used to transmit power and supply voltage as well as DVFS parameters to a host PC. The data transmission is performed using an Ethernet network connection.
- Software tools, such as a data capture software and evaluation scripts, are used to gather the data from the PPDU and to present the results of performed analyses.

4.1 Emulation System Architecture

The objectives of this master thesis can be subdivided into two distinct parts. In the first part, a power and supply voltage emulation platform is designed, developed and integrated. It enables the operator to evaluate and explore the power consumption and supply voltage behavior of a user given target hardware. The only precondition is that the target hardware must be available as synthesizable VHDL code. The target design is then integrated into an FPGA together with *power and supply voltage estimation units*. The power and supply voltage information gathered by these units is transferred to a host PC, where further data interpretation is performed with software tools like MATLAB, Excel, etc. Figure 4.1 from [GBH+09] depicts the basic concept of this emulation system. The

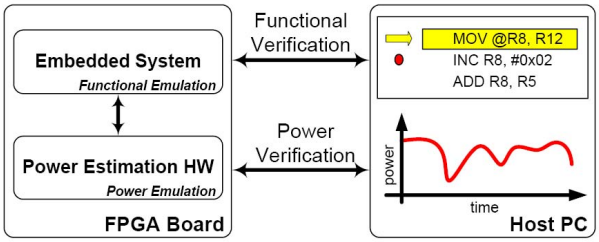

Figure 4.1: Basic Emulation System Approach [GBH+09]

advantages of this emulation based analysis technique are:

- The target design/hardware can be explored and analyzed regarding power consumption and supply voltage behavior in early design stages.
- Power bugs can be detected and resolved before the tape-out. Power bugs can be, for example, supply voltage drops or power consumption peaks that exceed a maximum allowed threshold. Such undesired circumstances may compromise a hardware's functionality. In particular, supply voltage drops can be extremely lethal for a smart card system (see Section 2.4).
- The power and supply voltage analysis is performed in real time by dedicated hardware. Therefore, this analysis is many times faster than a simulation-based one (cf. Table 2.1).

This master thesis features as target hardware a smart card with a symmetric multicore processor system. A very limited power supply and a high sensitiveness regarding power consumption changes are the prevalent challenges when dealing with smart card systems. Thus, a smart card emulation system, that is aware of its power consumption and supply voltage, is built for testing purposes.

In the second part, the emulation platform is enhanced with dynamic voltage and frequency scaling techniques. The objective is to increase the robustness of the system against power and supply voltage emergencies. Various DVFS algorithms are implemented and their effectiveness is analyzed and compared. Representative embedded benchmarking programs are used for these tests.

The final design of the emulation platform, which is illustrated in Figure 4.2, consists of the following components:

- LEON3 specific hardware components
  - Several symmetric LEON3 processor cores, divided in one master core and in „n“ slave cores.
- Power management specific hardware components
  - One *power estimation unit* (PEU) for each slave core, which estimates the slave processor core's instantaneous power consumption.
  - A *power management unit* (PMU) implements several DVFS algorithms which use power values for DVFS control decisions.
  - A *voltage drop compensation unit* (VDCU) estimates the supply voltage (done by an internal *supply voltage estimation unit* (SVEU)) and implements several DVFS policies using voltage and power values for DVFS control decisions.
  - The emulation platform can either be operated with the PMU or VDCU. A multiplexer unit is used to pass only the analysis information of the currently active PMU or VDCU to the PPDU.
  - The *power performance and debug unit's* (PPDU) tasks are to transmit DVFS parameters, supply voltage and power values to the host PC for further evaluation and analysis purposes.
- Software Components
  - A control and benchmark firmware runs on master and slave cores.
  - A JAVA based software saves and evaluates the incoming PPDU data from the Ethernet interface.
  - MATLAB scripts perform further data post-processing, evaluation and illustration tasks.

In the following paragraphs, each component of the emulation system is explained in detail.

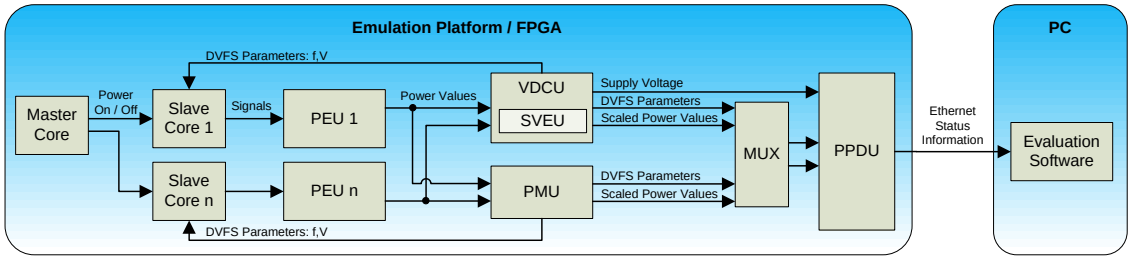

Figure 4.2: Emulation System Architecture

4.2 LEON3 Hardware Components

4.2.1 Master Core

The master core is a LEON3 processor that configures and controls the emulation system. It is not connected to any *power estimation unit*. When the emulation platform is switched on, the master core is the only active one. Then, the master core configures and controls the PEU, VDCU, PMU and PPDU through the AMBA APB bus system and finally activates/deactivates the target hardware (slave cores).

4.2.2 Slave Cores

These cores are LEON3 processors that represent the actual target hardware under test. When the emulation platform is powered up, all slave cores remain deactivated until the master core has finished initializing all analysis units (PEU, PMU, VDCU, PPDU). Benchmark programs are then run on slave cores. The PEU and SVEU analyze these slave cores continuously and deliver the corresponding power and supply voltage data. PMU and VDCU execute various DVFS policies with the help of the results from the PE and SVE units. Slave cores' DVFS voltage and frequency parameters are modified during the execution of the DVFS policies.

4.3 Power Management Components

4.3.1 Power Estimation Units

These units estimate the power consumption of the target hardware/slave cores. They have been taken directly from A. Genser [GBH+09] and are used without any modifications within this master project. The resulting power values form the basis for subsequent supply voltage estimations as well as DVFS control decisions. The applied mathematical and design principles within these PEUs are explained within Section 2.1.2 and Section 3.2 in detail. Basically, power sensors analyze internal processor signals and determine model states for each hardware component (cf. Figure 4.3). Each state has a specific power consumption value assigned to it. The total power consumption is the summation of all active component power consumption values.

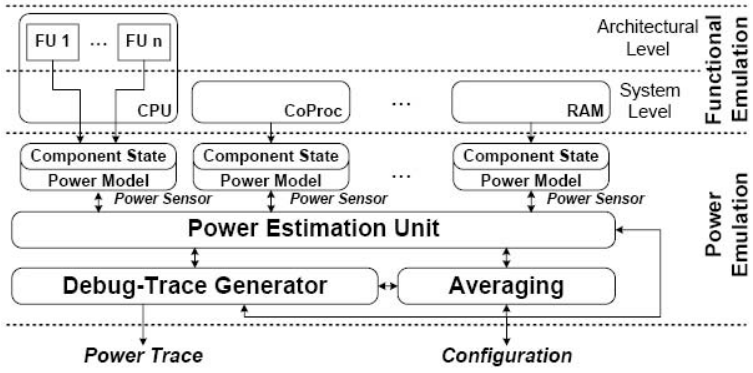

Figure 4.3: Power emulation harddware [GBH+09]

Figure 4.2 depicts the integration of the PEUs within the emulation system. An AMBA APB interface is used to configure and control the PEUs. Genser's PEU supports the following configuration options:

PE_CTRL - Address 0x80000A00

This register defines the operation modes of the PEU.

Name	Bit	Direction	Description
PE_EN	0	RW	Enables the complete PEU.
AVG_EN	1	RW	Enables value averaging. The kind of averaging mode used is further specified by the AVG_MODE bit.
AVG_MODE	2	RW	0: Standard cycle-accurate averaging 1: Coarse-grained averaging. The n-the cycle, which is taken into account, is specified in the PE_AVGSTEP register.

Table 4.1: Power Estimation Unit Register - PE_CTRL

PE_AVGSTEP - Address 0x80000A04

Name	Bit	Direction	Description
PE_AVGSTEP	0:31	RW	Defines the n-th cycle, which is used for coarse-grained averaging.

Table 4.2: Power Estimation Unit Register - PE_AVGSTEP

POWVAL - Address 0x80000A08

Name	Bit	Direction	Description
POWVAL	0:15	R	Contains the currently estimated power value. The register width can be modified through the RES_VAL_WIDTH definition. In case the PEU is disabled by the PE_EN bit, this register does not contain any valid data.

Table 4.3: Power Estimation Unit Register - POWVAL

4.3.2 Supply Voltage Estimation Unit

Estimating the supply voltage of the target hardware is the designated task of *supply voltage estimation unit* (SVEU). It is instantiated and used by the *voltage drop compensation unit* (VDCU). The incorporated mathematical computations utilize a power supply network model based upon a smart card system (see Section 2.4.1). The basic concept for this unit comes from Genser's approach, which is described in Section 2.2.2 and Section 3.4.

The basic functionality of the SVEU can be described as follows: The instantaneous power consumption of all processor cores are estimated by the PEUs and are delivered to the SVEU. The power values are transformed into corresponding electric current values. Several mathematical operations are then executed based on the smart card's power supply model (see Section 2.4.1) and the instantaneous applied core clock frequencies. The resulting $v(t)$ supply voltage values (cf. Figure 2.14) form the basis for VDC DVFS control algorithms. Figure 4.4 depicts the basic architectural model of the SVEU.

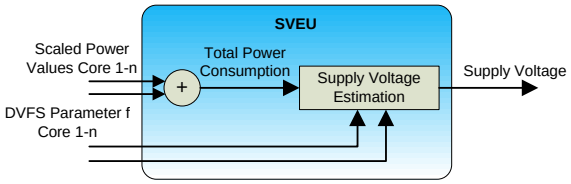

Figure 4.4: SVEU Architecture

During this master project, two distinct versions of the SVEU have been designed. The first design implements the CORDIC algorithm, which is the first attempt to improve Genser's approach by calculating the exponential function more precisely. The second design implements the electrical charge based approach. This approach bypasses the very complex exponential calculation completely by considering electrical charges which charge or discharge the smart card's capacitor. Both design implementations are presented and discussed in Section 5.3.2.

4.3.3 Power Management Unit

The *power management unit* (PMU) combines the DVFS policies which acts based on the instantaneous power consumption.

The PEU delivers power information to the PMU. These power values are then scaled on the basis of the currently set DVFS parameters. The DVFS policy analyzes the scaled power values and controls the DVFS parameters correspondingly. For further analysis tasks all relevant status information is forwarded to the PPDU, which sends the data to the host PC ultimately. Figure 4.5 depicts the basic design of the *power management unit*.

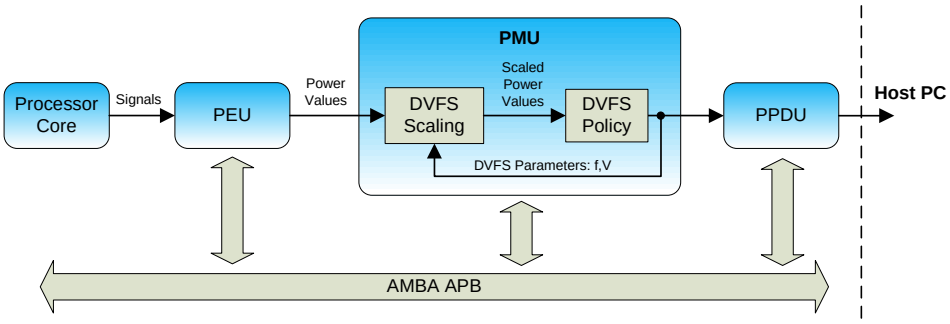

Figure 4.5: PMU Architecture

Configuration tasks can be performed through an AMBA APB interface. The following tables describe the supported features of the interface.

PM_CTRL - Address 0x80000B00

This register defines the operation modes of the *power management unit*. Only one DVFS policy can be enabled at the same time.

Name	Bit	Direction	Description
PM_EN	0	RW	Enables the complete PMU
PM_POLICY[1:5]	1	RW	Enables PM Greedy DVFS Policy
	2	-	Reserved
	3	RW	Enables PM Power DVFS Policy
	4	RW	Enables PM Performance DVFS Policy
	5	RW	Enables PM Gradient DVFS Policy

Table 4.4: Power Management Unit Register - PM_CTRL

PM_POWER_SETPOINT - Address 0x80000B04

Name	Bit	Direction	Description
PM_POWER_SETPOINT	0:31	RW	Defines the DVFS policies' control setpoint/target value

Table 4.5: Power Management Unit Register - PM_POWER_SETPOINT

PM_POWER_VALUE - Address 0x80000B08

Name	Bit	Direction	Description
PM_POWER_VALUE	0:31	R	Contains the instantaneous DVFS scaled power value

Table 4.6: Power Management Unit Register - PM_POWER_VALUE

PM_RESET - Address 0x80000B0C

Name	Bit	Direction	Description
PM_RESET	1	W	Resets the PMU

Table 4.7: Power Management Unit Register - PM_RESET

4.3.4 Voltage Drop Compensation Unit

This unit is designed to estimate the target hardware's supply voltage and compensate detected voltage drops with the help of DVFS policies. Figure 4.6 presents the basic architecture of this unit. The functionality can be described as follows: The VDCU receives the instantaneous power values from the PEUs. These power values are then scaled on the basis of the currently set DVFS parameters and forwarded to the internally instantiated SVEU. Within the SVEU, the calculation/estimation of the target hardware's supply voltage is performed. DVFS policies are fed with the instantaneous power and

supply voltage values. Based upon this information the slave cores' DVFS frequency and voltage parameters are modified correspondingly.

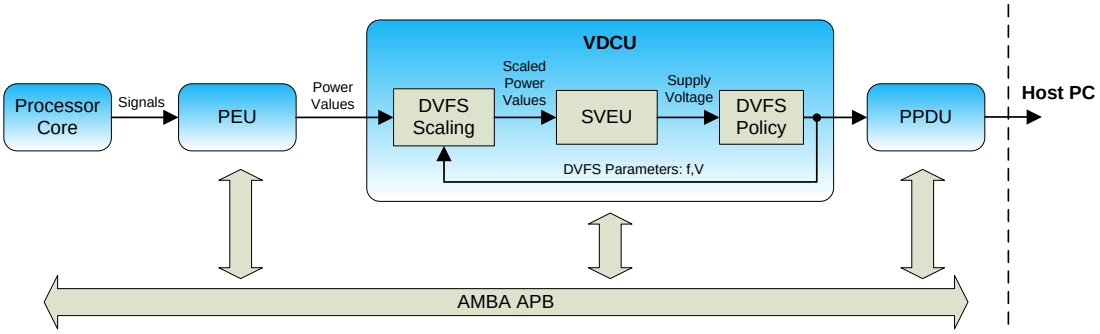

Figure 4.6: VDCU Architecture

An AMBA APB interface is supported for configuration tasks. The following configuration options are supported:

VDC_CTRL - Address 0x80000E00

This register defines the operation modes of the VDCU. Only one DVFS policy can be enabled at the same time.

Name	Bit	Direction	Description
VDC_EN	0	RW	Enables the complete VDCU
VDC_POLICY[1:12]	1	-	Reserved
	2	RW	Enables VDC Greedy DVFS Policy
	3	-	Reserved
	4	RW	Enables VDC Power DVFS Policy
	5	RW	Enables VDC Performance DVFS Policy
	6	-	Reserved
	7	RW	Enables VDC Gradient DVFS Policy
	8	RW	Enables VDC Gradient Delay DVFS Policy
	9	-	Reserved
	10	RW	Enables VDC Priority DVFS Policy
	11	-	Reserved
	12	RW	Enables VDC Hybrid Greedy Voltage/Power DVFS Policy

Table 4.8: Voltage Drop Compensation Unit Register - VDC_CTRL

VDC_VOLTAGE_SETPOINT - Address 0x80000E04

Name	Bit	Direction	Description
VDC_VOLTAGE_SETPOINT	0:15	RW	Defines the DVFS policies' control setpoint/target value

Table 4.9: Voltage Drop Compensation Unit Register - VDC_VOLTAGE_SETPOINT

VDC_VOLTAGE_VALUE - Address 0x80000E08

Name	Bit	Direction	Description
VDC_VOLTAGE_VALUE	0:15	R	Contains the instantaneous supply voltage level

Table 4.10: Voltage Drop Compensation Unit Register - VDC_VOLTAGE_VALUE

VDC_RESET - Address 0x80000E0C

Name	Bit	Direction	Description
VDC_RESET	1	W	Resets the VDCU

Table 4.11: Voltage Drop Compensation Unit Register - VDC_RESET

VDC_POWER_SETPOINT - Address 0x80000E10

Name	Bit	Direction	Description
VDC_POWER_SETPOINT	0:15	RW	Defines the DVFS policies' control target value

Table 4.12: Voltage Drop CompensationUnit Register - VDC_POWER_TARGET

4.3.5 Multiplexer

A multiplexer is implemented to pass only the analysis data of the currently active PMU or VDCU to the PPDU. The emulation platform is not designed to be operated with both units simultaneously.

4.3.6 Power Performance and Debug Unit

M. Lackner's PPDU is used to transmit the analysis data from the PMU and VDCU to the host PC (cf. Figure 4.5 and Figure 4.6). The original PPDU version supports several different operation modes and transmits huge amounts of analysis and performance data to the host PC. To reduce the FPGA area occupation and to increase the accurateness of the data (the less data transmitted, the less data is averaged), a lot of unneeded functionality has been removed. The newly designed PPDU supports only two operation modes and transmits only power consumption, supply voltage, DVFS frequency and DVFS voltage values to the host PC. The PPDU's AMBA APB interface can be used to configure the unit. The following table describes the supported features.

PPDU_MODUS - Address 0x80000D00

Name	Bit	Direction	Description
ENABLE	31	RW	Enables the complete PPDU
MODUS	30:29	RW	Defines the operation mode. '01': Power consumption and supply voltage information is transmitted. '10': DVFS frequency and DVFS voltage values are transmitted.

Table 4.13: Power Performance and Debug Unit Register - PPDU_MODUS

4.4 Software Components

4.4.1 Firmware

The emulation platform is designed to come along without any operating system. Due to the fact that a multi-core processor system is given and no operating system is used, the same firmware is executed on each processor core. Thus, the firmware must be designed in a way to run on both, the master and the slave cores properly. In order to simplify the design, functions related specifically to the hardware operation are separated from the benchmarking functions. This is done by using an application programming interface (API) structure. The basic flow of activities is presented by Figure 4.7: When the emulation system is powered on, the firmware is executed on the master core only. First of all, a check is executed to verify if the software runs on the master core or on slave cores. Because the program runs on the master core, the master core branch is pursued and its role is to configure the emulation platform (PEU, PMU, VDCU, PPDU). After the initialization procedure, the target hardware/slave cores are started. Now the same firmware is executed on the slave cores but this time, it is the slave-branch that is executed. This branch of the firmware finally implements the various benchmarks. To offer a basis of comparison with results from other publications, the MiBench benchmarking suite [GRE+01] is used. MiBench has been developed by the University of Michigan and is specialized in representative embedded applications.

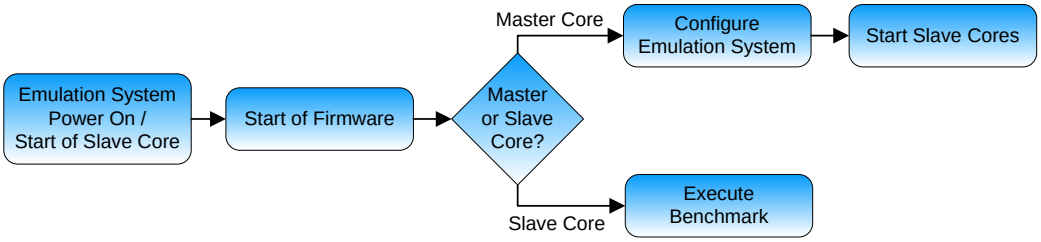

Figure 4.7: Firmware Process Flow

4.4.2 Data Capture Tool

M. Lackner's JAVA tool (see Section 3.5 for further information) is used to capture and save the incoming analysis data. It runs on a host PC and reads the data coming from the Ethernet interface. Once the tool has gathered all the data, it can present it on screen or save it to a log file.

4.4.3 Evaluation Software

MATLAB scripts as well as Excel are used to read the log files and perform analyses. Also, it can illustrate the analyses in a professional way. In general, there are two possible analysis data sources. The scripts are designed so that it can read either log files from Lackner's tool or log files coming from ModelSim simulations of the platform.

Chapter 5

Implementation of the Emulation Platform

In Chapter 3 and Chapter 4, the basic design of the power and supply voltage emulation platform has been presented. This chapter now explains in more details how this project is realized.

5.1 Design and Implementation Process

This section presents the major design and implementation process as well as the tools which are used during this project. Figure 5.1 gives an overview of the performed design and implementation tasks.

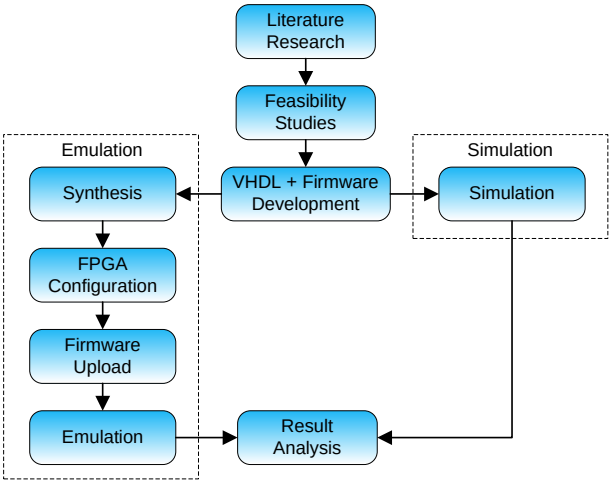

Figure 5.1: Design and Implementation Process

Literature Research and Feasibility Studies

Before any implementation or design activity can be accomplished, a theoretical background needs to be acquired. The topics which have been explored during this master

thesis are described and explained in Chapter 2. Design and architectural ideas are then formed and evaluated based on the gained experiences. Feasibility studies of CORDIC and other exponential algorithms are then programmed in C and analyzed with Visual Studio. MATLAB [Mat10] and LTSpice IV [Tec10] are fundamental tools which are used during the evaluation and development of the electrical charge based SVE model of the SVEU. LTSpice IV is a SPICE simulator, which can be downloaded from the Internet[1].

VHDL and Firmware Development

The development of the VHDL source code is one of the major activities during the implementation process. The development is based upon the LEON3 VHDL source code which is published through the Aeroflex Gaisler's GRLIB package. This library can be downloaded free of charge from the Gaisler homepage[2]. To get the GRLIB environment running, Linux or a Linux-like environment is required. Due to the fact that most of the development and result evaluation tasks are performed under Windows, the Linux-like environment Cygwin has been chosen [Cyg10]. It can be downloaded from the Internet[3].

The firmware source code as well as the VHDL source code are written and edited with Visual Studio editor. C++ based tools are developed to ease the VHDL development. Among other tasks, they are used for integer-to-fixed point conversions, lookup table generations, etc. Visual Studio does not support VHDL source code highlighting originally. Nevertheless, it is used because Visual Studio is one of the most comfortable and clearly arranged source code editors available.

Generally, whenever source code is compiled, the outcome must be mapped towards a specific platform (e.g., Windows, Linux, etc) or processor (e.g., ARM, SPARC V8, x86, etc). Thus, a LEON3 specialized cross-compiler is needed for this purpose. There are a few different compilers available, which can be all downloaded from the Gaisler homepage. This project utilizes the Bare-C Cross Compiler (BCC).

Simulation

The developed VHDL and firmware source code can be tested before the time intense hardware synthesis is performed. It's an alternative that saves time and money. In this project, simulations are performed with ModelSim 6.6 SE [Gra10]. ModelSim can be downloaded from the Internet[4] but a special licensing environment is needed to operate it properly. This issue can be resolved for example by established a virtual private network connection to Graz University of Technology.

Synthesis

In order to build the hardware integrated power and supply voltage emulation system, the VHDL source code needs to be converted into a netlist file that is used to configure the FPGA. This task is performed with the help of Xilinx ISE, which is available from

[1] http://www.linear.com/ [last access 2010-05-14].
[2] http://www.gaisler.com [last access 2010-05-14].
[3] http://www.cygwin.com [last access 2010-05-14].
[4] http://model.com [last access 2010-05-14].

the internet in the Xilinx's Webpack [5]. Basically, the netlist generation procedure can be subdivided into the following tasks:

- Synthesis: The source code is checked and the hierarchy of the design is analyzed. The outcome is a netlist, which is saved in an NGC file.
- Translation: The netlists and the design constraints are merged. At the end of this task a native generic database (NGD) file is generated.
- Mapping: The logic design is mapped with the help of the NGD file to the Xilinx Spartan 3 FPGA. As a result a native circuit design (NCD) file is created.
- Place and route: Takes the NCD file, places and routes the design and finally saves the result in another NCD file.
- Bitfile generation: Based on the routed NCD file a bitstream file (BIT or ISC) is produced. These files are used to configure the FPGA.

FPGA Configuration

During this task, the FPGA is configured with the developed VHDL design. The bitstream file is required for this purpose, which is generated during the synthesis process. The configuration is basically done with either Xilinx ISE or the GRLIB specific tool through the JTAG interface [IEE01].

Firmware Upload

The firmware now needs to be uploaded into the memory of the development board. This is performed with the help of the debug monitoring program GRMON and the JTAG interface. GRMON is available in the Internet[6] and can be downloaded free of charge.

Emulation

The emulation system now has been integrated into the FPGA and the firmware is uploaded. The emulation system is ready to be put into operation. This is done by executing the „run“ command within the GRMON tool.

Result Analysis

Figure 5.1 and Figure 5.2 depict the two possible approaches to generate and evaluate results. One method is emulation and the other one is simulation.

- Emulation: During the execution of the firmware, analysis data is gathered by the emulation system's internal PPDU and is sent to the host PC through the Ethernet interface. At PC side, a JAVA tool (see Section 3.5) is used to collect and save the data into log files. These log files contain averaged information of supply voltage and power consumption behavior as well as averaged DVFS parameters. Averaging

[5] http://xilinx.com [last access 2010-05-14].
[6] http://www.gaisler.com [last access 2010-05-14].

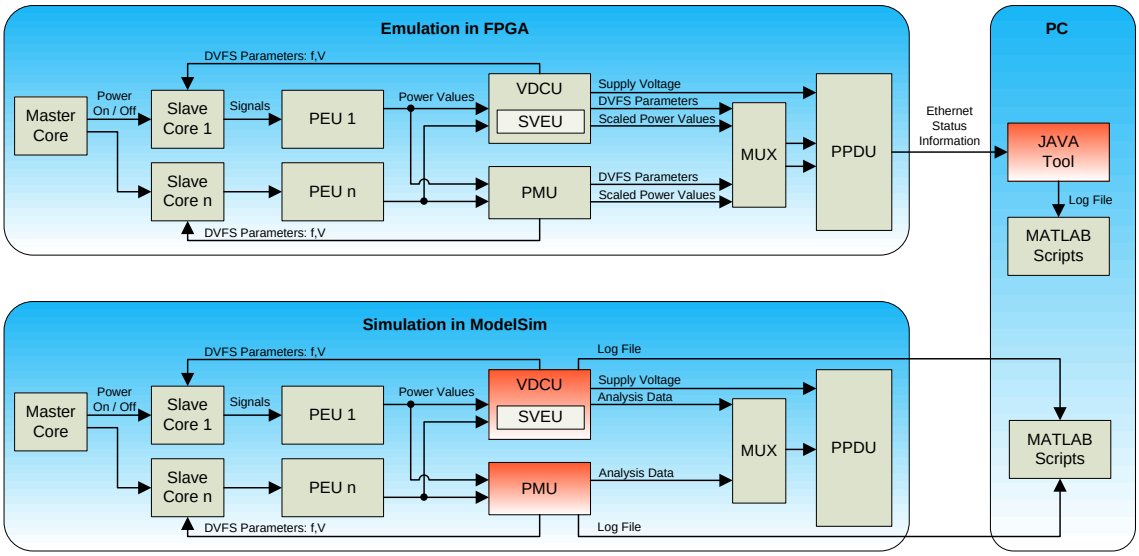

Figure 5.2: Result and Log File Generation

is necessary because the Ethernet interface is too limited in terms of bandwidth. For details refer to [Lac10].

- Simulation: The simulation of the platform is performed by ModelSim. The data generated by the PMU and VDCU are saved by ModelSim into log files. These files contain cycle accurate information regarding supply voltage and power consumption behavior as well as DVFS parameters.

MATLAB and Excel are used to analyze and visualize the content of the log files. Those scripts calculate execution time of the benchmark, mean and standard deviation for supply voltage and power consumption, as well as variation in time from desired value of supply voltage and power consumption. One can recognize a good DVFS implementation if it respects the following criteria:

- Faster execution time is an indication of economical energy consumption.
- Supply voltage must be above the minimal supply voltage value at all times in order to ensure proper operation.
- Lower deviation values are an indication that DVFS facilitates the operation of the smart card.

5.2 Power Values in Gate Level Simulation and Hardware Domain

The power values form the basis for any further power analysis and supply voltage estimation. They cross several representation domains and are depicted in Figure 5.3. This section describes the exact mathematical transformations of these values.

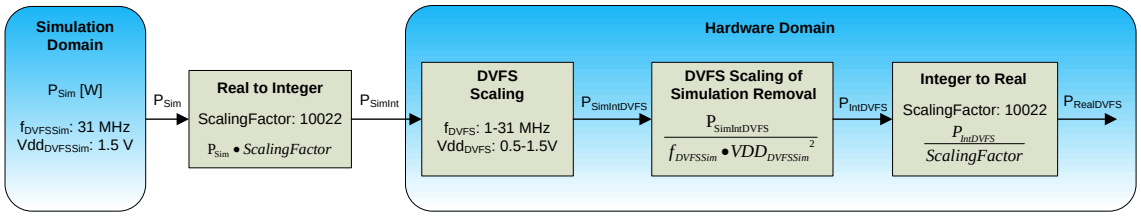

Figure 5.3: Power Value Scaling and Representation

A gate level simulation of the target hardware forms the origin for any further power/supply voltage analysis. The target hardware is operated at 31 MHz and 1.5 V. The states from the power model are perturbed and the dedicated power dissipation is determined. These power values, which are given in Watt (Equation (5.1) exemplifies the value of one power value), now need to be transformed into a representation suitable for further hardware computations. This is achieved by converting them into an integer data type. A multiplication factor, named ScalingFactor, is used for this purpose (5.2) and has a value of 15033. DVFS algorithms are then scaling the P_{SimInt} values with the specific DVFS voltage and DVFS frequency parameters according to Equation (5.3). After that, the original DVFS parameters (31 MHz and 1.5 V) from the gate level simulation need to be removed from the power value by division. This is done in Equation (5.4). The power value is then converted from the integer domain to real domain because the real supply voltage value is more meaningful to an hardware developer. Furthermore, the real voltage is needed so that one can verify at all times if the supply voltage constraint is respected. This is done in Equation (5.5).

$$P_{Sim} = 0,1W \tag{5.1}$$

$$P_{SimInt} = \lfloor P_{Sim} \cdot ScalingFactor \rfloor \tag{5.2}$$

$$P_{SimIntDVFS} = P_{SimInt} \cdot f_{DVFS} \cdot Vdd_{DVFS} \cdot Vdd_{DVFS} \tag{5.3}$$

$$P_{IntDVFS} = \frac{P_{SimIntDVFS}}{f_{DVFS_{Sim}} \cdot Vdd_{DVFS_{Sim}} \cdot Vdd_{DVFS_{Sim}}} \tag{5.4}$$

$$P_{RealDVFS} = \frac{P_{IntDVFS}}{ScalingFactor} \tag{5.5}$$

To compute the transformation of Equation (5.4) and Equation (5.5) easily in hardware, the following trick is used. Equation (5.6) shows the value of the divisor from Equation (5.4) and Equation (5.5). Equation (5.7) shows the nearest possible power of two value. Thus, the integer value $P_{SimIntDVFS}$ is divided by 2^{20}, which can be implemented by a hardware based shift bit operation. A calculation error of less than 0,03‰ is produced during these representation shifts.

$$ScalingFactor \cdot f_{DVFS_{Sim}} \cdot Vdd_{DVFS_{Sim}} \cdot Vdd_{DVFS_{Sim}} = 1048551.75 \tag{5.6}$$

$$2^{20} = 1048576 \tag{5.7}$$

A major prerequisite for a proper supply voltage estimation is the computation of the target hardware's drawn electrical current. According to Equation (5.8), this is done by dividing the instantaneous power consumption with the supply voltage (1.5 V), which is actually applied to the target hardware. Due to the fact that an implementation of a hardware based division is very costly, the ScalingFactor between simulation and hardware domain is divided by 1.5. Thus, the problem of the hardware based division is solved elegantly. Consequently, the final value of the ScalingFactor equals 10022.

$$P(t) = U(t) \cdot I(t) \tag{5.8}$$

The SVEU uses the VHDL support library[7] to ease the implementation of the described fixed point computations.

5.3 Power Management Hardware Components

Figure 5.4 depicts the emulation platform with the highlighted power management units, which are explained in this section.

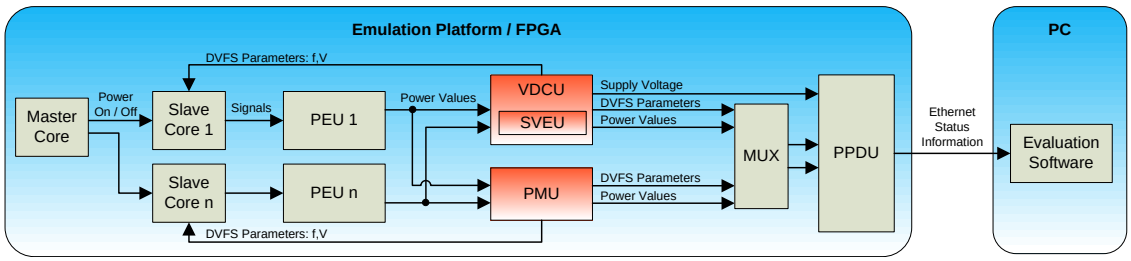

Figure 5.4: Emulation System Architecture with Highlighted Power Management Units

5.3.1 Improved DVFS Scaling

The original DVFS scaling approach used two architectural units, one lookup table and three multipliers (cf. Figure 3.4). Each processor core has one DVFS scaling unit assigned. A DVFS scaling computation is done within two clock cycles. Due to the fact, that the DVFS scaling unit is part of the controller loop, it influences the control delay directly and consequently the performance of all DVFS algorithms.

An improved DVFS scaling unit has been implemented during this master thesis. It features a lookup table between frequency and $f \cdot v^2$ as well as only one multiplier. This design is feasible because each frequency value (31 discrete values are fragmenting the frequency range between 1 and 31 MHz) is exactly one voltage value assigned. Furthermore, the two original architecture units are merged together improving the calculation speed by one clock cycle. This speed increase can be identified in Figure 5.5 if one compares the old implementation with the new one. Each architectural unit (delimited by a blue box) introduces a delay of one clock cycle. Thanks to the new, tighter and faster design the performance of all DVFS algorithms is improved which is directly reflected by better benchmarking results.

[7] http://www.vhdl.org/fphdl/ [last access 2011-04-22].

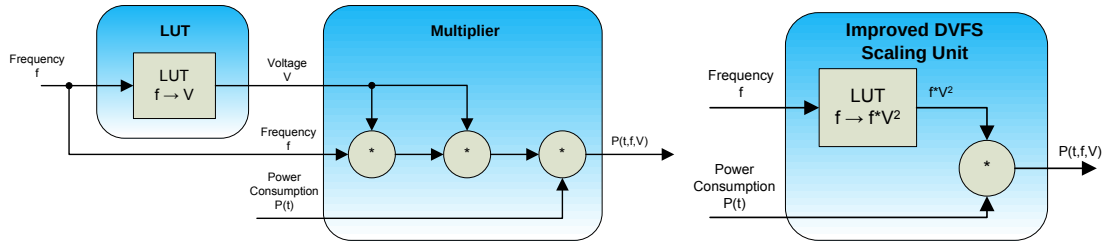

Figure 5.5: Comparison of Original and Improved DVFS Scaling Units

5.3.2 Supply Voltage Estimation Unit

The *supply voltage estimation unit* is the central part of this master thesis. Genser et al. have already proposed a version of a *supply voltage estimation unit* (see [GBH+11] and Chapter 3.4). Genser's voltage estimation architecture acts as a basis for this master thesis. However, the following issue of Genser's implementation is addressed and improved during this project: Term $e^{(-t \cdot R_i \cdot C)}$ of Equation (2.7) is expressed by a constant. The main advantage of this approach is that a very complex exponential function calculation is avoided. But in case the parameter t changes a calculation error is introduced. Due to the fact DVFS algorithms modify a processor's frequency, t is in general not constant.

CORDIC Approach

An implementation of the CORDIC algorithm, based on a feasibility study, has been integrated in hardware during this master thesis. The goal is to compute the exponential term $e^{(-t \cdot R_i \cdot C)}$ of the supply voltage estimation procedure more precisely. This CORDIC implementation is specialized in the hyperbolic - rotation mode to approximate the exponential function (see Chapter 2.5.1). Figure 5.6 from [EL04] depicts the utilized architecture.

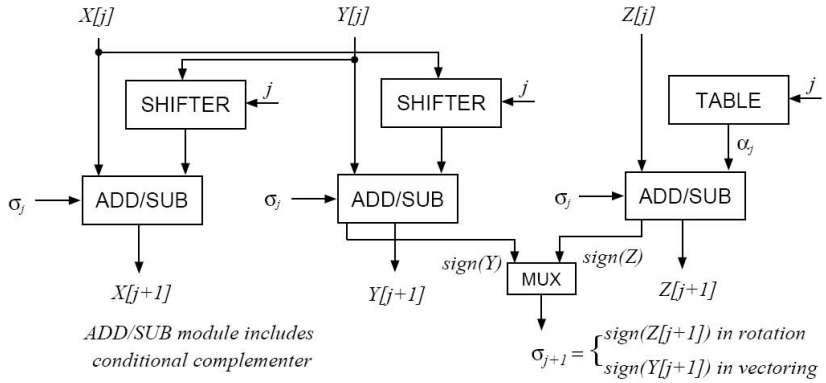

Figure 5.6: CORDIC Hardware Integration [EL04]

The CORDIC exponential algorithm supports only a very limited convergence radius of ± 1.11817. This convergence radius limits the exponent input parameter to the same value range. Therefore, a way to increase the convergence radius has been looked for. A solution has been found by taking advantage of exponential identities. Based upon Equation (5.9) [Kre93], an alternative formula (5.10) is being introduced, which is more

suitable for hardware integrations. After a simplification, the Equation (5.11) is divided by 2^{-k}. Equation (5.12) is then multiplied with e^x, resulting in (5.13). Finally, the simplified Equation (5.14) reduces the argument of positive x values. This approach can easily be adapted for negative x values and is very suitable to be integrated into hardware. Only one addition (if x < -1.11817) or subtraction (if x > 1.11817), a table lookup ($k\ln(2)$) and a bit shift operation are needed for this task.

$$x = e^{\ln(x)} \tag{5.9}$$
$$2^{-k} = e^{\ln(2^{-k})} \tag{5.10}$$
$$2^{-k} = e^{-k\cdot\ln(2)} \tag{5.11}$$
$$1 = 2^k \cdot e^{-k\cdot\ln(2)} \tag{5.12}$$
$$e^x = 2^k \cdot e^{-k\cdot\ln(2)} e^x \tag{5.13}$$
$$e^x = 2^k \cdot e^{x-k\cdot\ln(2)} \tag{5.14}$$

Figure 5.7 illustrates the three major states of the final hardware integrated module. At first, the input parameter is adjusted by the factor $k\ln(2)$ to comply with the limited convergence radius. Then, the CORDIC iterations are done. And finally, the outcome needs to be readjusted with the factor 2^k.

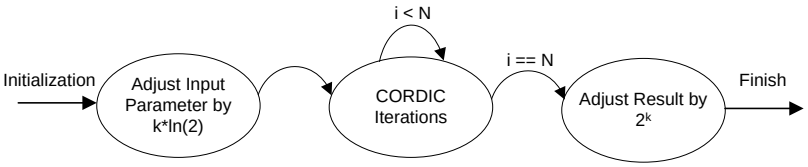

Figure 5.7: CORDIC Implementation State Diagram

The number of cycles needed for the computation depends mostly on the desired accuracy. n accuracy bits request n+1 CORDIC iterations. The final implementation was able to do one iteration within two clock cycles. Consequently, in case a 16-bit accuracy is aimed, 34 cycles for the CORDIC iterations and a few cycles for initializing, value adjustments and finishing tasks are needed. There are possibilities to increase the computation speed by parallelizing the design, but then the hardware costs increase and a latency of 17 clock cycles still remains. This kind of delay directly influences the control delay of all VDC DVFS algorithms. The higher the control the delay, the worse the control performance.

There are a few alternative CORDIC implementations available (e.g., from OpenCores[8] or Xilinx). The Xilinx IP Generator offers a comfortable way to customize the CORDIC unit's design regarding special needs. According to [Xil04], Xilinx's fastest possible implementation is able to compute n accuracy bits within n clock cycles with the help of pipelining and parallelization at the costs of 660 FPGA slices. Considering these facts the CORDIC SVEU design disqualifies.

[8] http://www.opencores.org/ [last access 2011-04-22].

Electrical Charge Approach

The fact that the CORDIC implementation is too slow, a completely different approach has been searched and developed. Figure 5.8 illustrates the simplified analytical model of a smart card. The main idea behind this method is the calculation of the electrical charges which affect the capacitor C. Table 5.1 presents the electrical charges, which are used during the calculations.

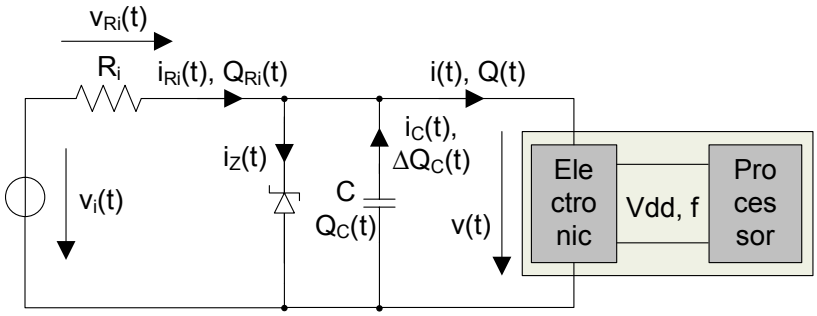

Figure 5.8: Smart Card Model

Variable	Interpretation
$Q_{Ri}(t)$	Electric charge provided by the magnetic field during one processor cycle, calculated with the help of $i_{Ri}(t)$
$Q_i(t)$	Electric charge consumed by the target hardware during one processor cycle, calculated with the help of $i(t)$
$Q_C(t)$	Electric charge level of the capacitor C
$\Delta Q_C(t)$	Capacitor C is charged or discharged by the amount of ΔQ_C

Table 5.1: Interpretation of the Electric Charges

The analysis of this model starts by defining the following facts: R_i (62.5Ω) and C (100nF) are given; capacitor C is fully charged. In addition, it is assumed that the value of $v_i(t)$ is known (2.5V). A voltage sensor comes into operation in smart card systems for this purpose. Based on these assumptions, $v_{Ri}(t)$ and $i_{Ri}(t)$ can be computed easily by Equation (5.15) and Equation (5.16). After that, the electrical charge $Q_{Ri}(t)$ is computed according to Equationn (5.17) with the help of the electrical current i_{Ri} and the reciprocal processor clock frequency. The processor's power consumption is estimated now by the *power estimation unit* (further explanations be found in Chapter 3.2 and Chapter 5.2). This power value is transformed into the electric current $i(t)$. Then, the corresponding charge $Q(t)$ is computed on the basis of $i(t)$ and the instantaneous processor's clock frequency. The difference between $Q(t)$ and $Q_{Ri}(t)$ is captured by $\Delta Q_C(t)$. The specific characteristic of $\Delta Q_C(t)$ and the corresponding electrical current $i_C(t)$ are their changing directions. Depending on the processor's power consumption the charge $\Delta Q_C(t)$ flows either into the capacitor C (the capacitor is charged, $Q_C(t+1)$ increases) or contrariwise (the capacitor is discharged, $Q_C(t+1)$ decreases). Finally, the target hardware's supply voltage $v(t+1)$ can be determined by Equation (5.21).

$$v_{Ri}(t) = v_i(t) - v(t) \tag{5.15}$$

$$i_{Ri}(t) = v_{Ri}(t) \cdot \frac{1}{R_i} \tag{5.16}$$

$$Q_{Ri}(t) = i_{Ri}(t) \cdot \Delta t \tag{5.17}$$

$$Q_(t) = i(t) \cdot \Delta t \tag{5.18}$$

$$\Delta Q_C(t) = Q_{iRi}(t) - Q_i(t) \tag{5.19}$$

$$Q_C(t+1) = Q_C(t) + \Delta Q_C(t) \tag{5.20}$$

$$v(t+1) = Q_C(t+1) \cdot \frac{1}{C} \tag{5.21}$$

The hardware-integrated model uses only one lookup table and four multipliers. The lookup table is used to translate clock frequencies to their reciprocal time values. To ease the computation of the fixed point values, a VHDL support library[9] is used. The resulting supply voltage can be computed with a latency of two clock cycles (two latches are added to reduce the critical path). Furthermore, a multi-core system can be supported elegantly: Equation (5.18) and Equation (5.17) are improved by Equation (5.22) and Equation (5.23) for this purpose. Thus, the influence of each processor core can be computed individually.

$$Q_i(t) = i_{Core1}(t) \cdot \Delta t_{Core1} + i_{Core2}(t) \cdot \Delta t_{Core2} \tag{5.22}$$

$$Q_{iRi}(t) = i_{Ri}(t) \cdot \frac{\Delta t_{Core1} + \Delta t_{Core2}}{2} \tag{5.23}$$

Magnetic Field Changes

The implemented SVEU is capable to model a changing magnetic field intensity. This effect occurs when a smart card is moved within the magnetic field. Thus, the voltage $v_i(t)$ alters inevitably. Power Management and Voltage Drop Compensation DVFS algorithms are evaluated with this special test method regarding the stability of the resulting supply voltage $v(t)$.

5.3.3 Power Management Unit

The PMU is designed and implemented in order to flatten the target hardware's power profile. Figure 5.9 shows the detailed implementation of this unit. PEUs deliver estimated power consumption values from their dedicated processor cores. The DVFS scaling units are fed with these values and compute the scaled power values based on the current set of DVFS voltage and frequency parameters (see Figure 5.5). These values are then forwarded to the various PM DVFS algorithms. The AMBA APB interface is used to transmit the control setpoint value and the DVFS policy used to PMU. The processor core frequencies are output by the policy units and a multiplexer unit decides upon the settings from the AMBA APB interface what parameters are further used. Finally, the frequency values are forwarded to the DVFS scaling units and to the PPDU. The PPDU is responsible

[9] http://www.vhdl.org/fphdl/ [last access 2010-05-14].

to transmit the DVFS frequency and voltage values as well as the instantaneous power consumption to the host PC for further analysis tasks.

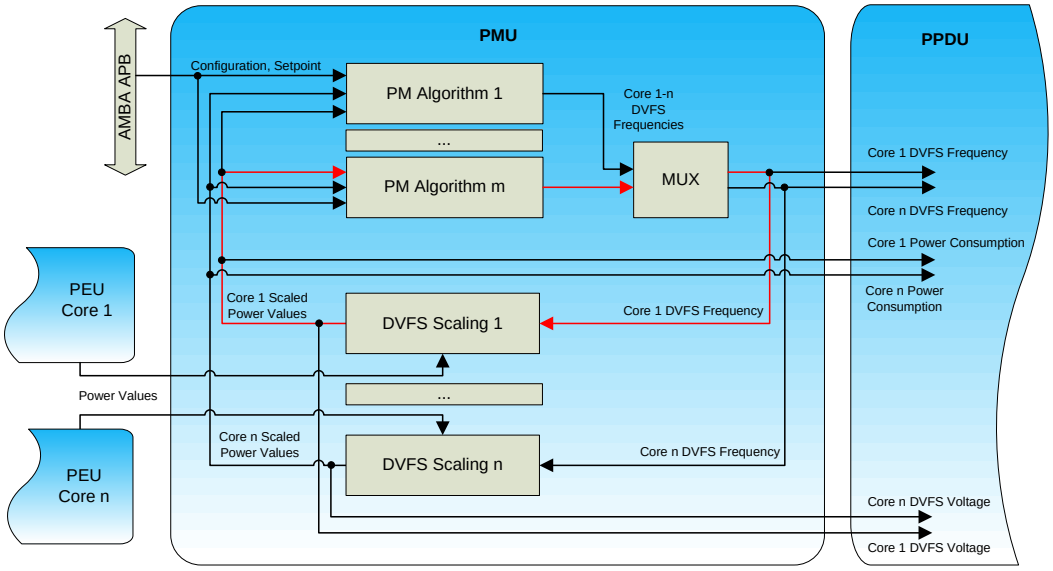

Figure 5.9: Power Management Unit Implementation

The delay of the PMU's control delay is very small. According to Figure 5.9, each unit (PM Algorithm, MUX, DVFS Scaling) introduces a one clock cycle delay. The whole control loop is shown by the red line in Figure 5.9. Therefore, the time needed to get a complete update on the PMU's output values is three clock cycles. Due to the small control loop delay, the PMU is very well suited to flatten the target hardware's power profile. The main disadvantage of the PMU is its unawareness of the target hardware's instantaneous supply voltage. In the following section, the PM related DVFS approach is explained and the advantages/disadvantages are depicted.

5.3.4 Power Management DVFS Algorithms

Power management algorithms perform DVFS modifications based upon the instantaneous power consumption of the target hardware and a predefined power setpoint.

On one hand, PM algorithms smooth the power profile very well and with low control delay, but on the other hand they are unaware of the target hardware's supply voltage. Basically, the PM algorithms are able to prevent voltage drops but a rather pessimistic power setpoint setup is needed for this purpose. So, a PM setup must deal with the following problems:

- At which value should the PM setpoint be set?
- If the PM setpoint is set too low, the performance of the benchmarking application is compromised but the voltage drop safety is increased.
- If the PM setpoint is set too high, voltage drops may occur more likely but the benchmarking application is executed faster.

In the following paragraphs, the various PM DVFS policies are presented in detail.

PM Greedy Algorithm

Greedy is a very simple but effective chip-wide DVFS algorithm. Figure 5.10 depicts the implementation of this DVFS method. An artificial control delay is implemented to cope with the PMU's control loop delay. If the control delay is reached, then the instantaneous power consumption and the power consumption setpoint are compared and DVFS frequency modification is ordered. For the case where the instantaneous total power consumption is higher than the power setpoint, all processor cores have their clock frequencies decreased by the same amount. Otherwise, if the total power consumption is lower than the power setpoint, all cores have their clock frequencies increased. The reason why there is a delay is actually to give the system time to adjust its power consumption according to the DVFS modifications. Once that the effect is significant, a new calculation of DVFS frequencies can be requested.

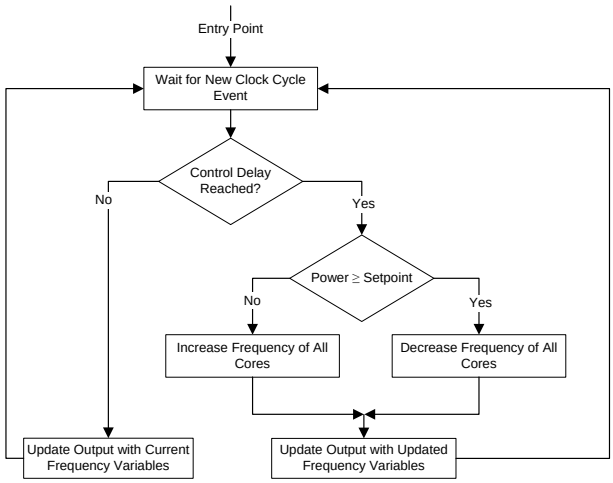

Figure 5.10: PM Greedy Algorithm

PM Gradient Algorithm

The Gradient algorithm is a per-core DVFS method, which controls the processor cores' frequencies based on their respective power consumption gradients and the instantaneous total power consumption. Figure 5.11 depicts this algorithm's sequence of activities. At first, power gradients are calculated. The number of cycles between power consumption measurements for this computation can be adapted easily by modifying a constant in the VHDL source code. Next, the list of processor cores is sorted ascendingly by means of their gradients. Power consumption gradients are compared:

- If the gradients are different:
  - If the total power consumption is below the power consumption setpoint then the core with the lowest power gradient gets an increase in its clock frequency.
  - If the total power consumption is above the power consumption setpoint then the core with the highest power gradient gets a decrease in its clock frequency.

- If the gradients are equal:
  - If the total power consumption is below the power consumption setpoint then the core with the lowest power consumption gets an increase in its clock frequency.
  - If the total power consumption is above the power consumption setpoint then the core with the highest power consumption gets a decrease in its clock frequency.

After the affected core has received its new clock frequency, the algorithm waits an amount of time equals to an user-defined control delay in order to be able to observe a significant change in power consumption before ordering a new calculation.

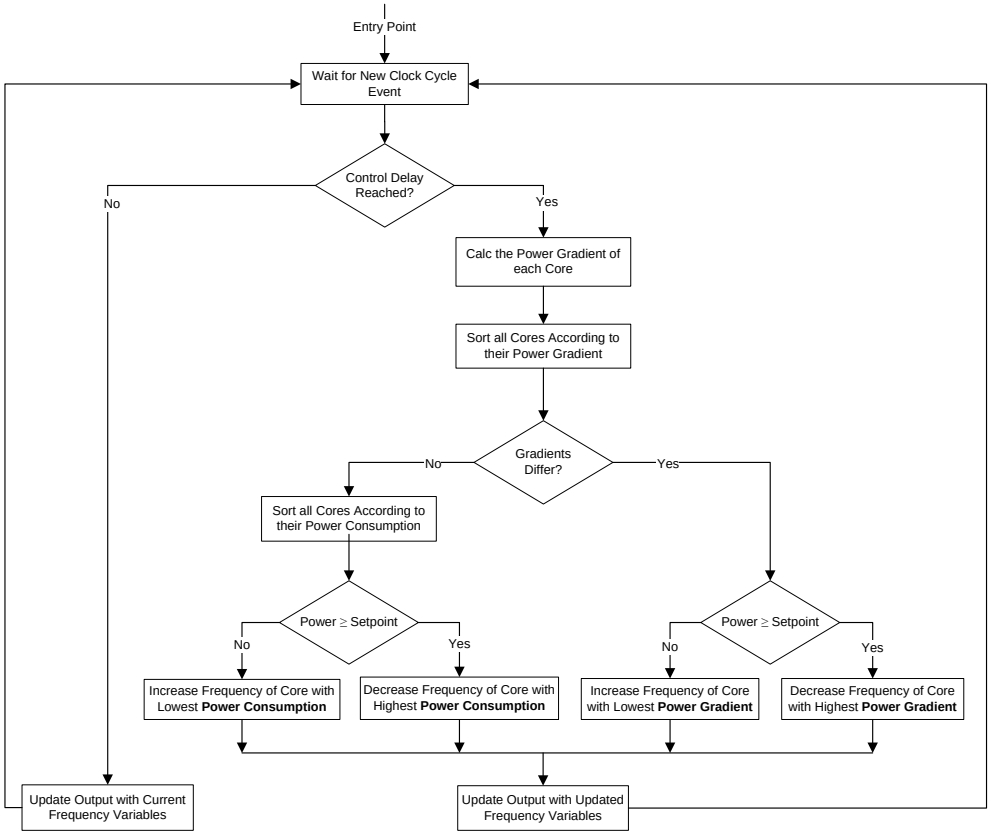

Figure 5.11: PM Gradient Algorithm

PM Power Algorithm

This is a per-core DVFS policy. Figure 5.12 depicts this algorithm's control sequence. At first, all cores are sorted according to their instantaneous power consumption. Then, the total power consumption is computed using the power consumption from each core. If the total power consumption is below the power setpoint, the core with the lowest power consumption gets an increase in its clock frequency. If the total power consumption is

above the setpoint value, then the processor core with the highest power consumption gets a decrease in its clock frequency.

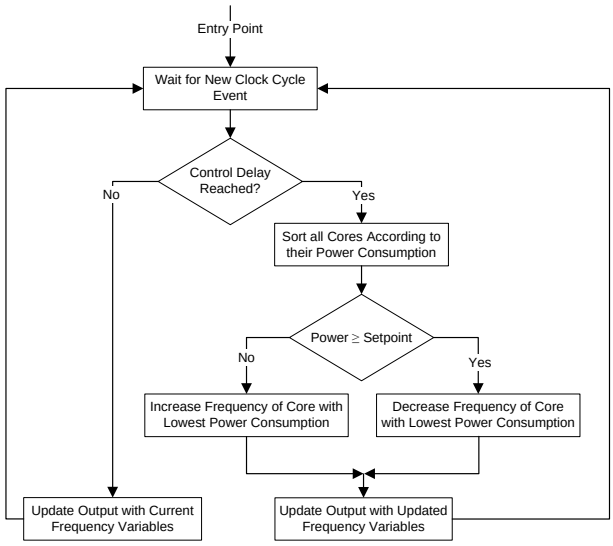

Figure 5.12: PM Power Algorithm

PM Performance Algorithm

This per-core algorithm decides with the help of performance information which core frequency must be altered. Figure 5.13 depicts this algorithm's control sequence. Each core has a performance index calculated as follows: At each current clock cycle, if the core executes a none-idle instruction it gets a pondered value according to its clock frequency. At each clock cycle, the sum of the last 100 pondered values is calculated: this is the performance metric. Then, all cores are sorted according to their current performance metric. Next, the total power consumption is computed using the power consumption from each core. If the total power consumption is below the power setpoint, the core with the highest performance gets an increase in its clock frequency. If the total power consumption is above the setpoint value, then the processor core with the lowest power performance gets a decrease in its clock frequency. This DVFS algorithm favors the best performing processor core in all cases.

5.3.5 Voltage Drop Compensation Unit

The VDCU is responsible for voltage drop detections and compensations. Figure 5.14 shows the detailed implementation scheme. This implementation works in the following way: An AMBA APB interface is utilized for configuration tasks (e.g., enabling the unit, selecting a DVFS policy, configuring the control setpoint value, etc). PEUs are delivering instantaneous power consumption values for each slave core. These values are then scaled in the improved DVFS scaling units with the currently set DVFS frequency parameters (explained in detailed in Section 5.3.1 and Figure 5.5). The scaled power values are then passed to the SVEU, which estimates the instantaneous target hardware's supply voltage. The power values as well as the supply voltage value are then passed to the DVFS units.

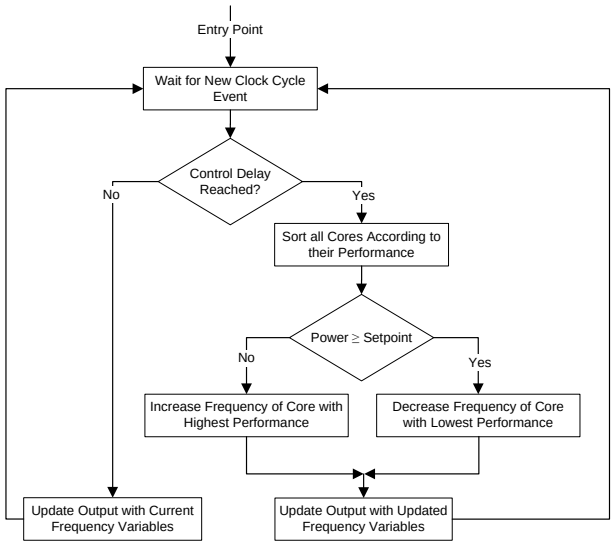

Figure 5.13: PM Performance Algorithm

What kind of DVFS policy is utilized is controlled through the AMBA APB interface. A multiplexer finally passes the appropriate DVFS frequencies to the DVFS scaling units to close the control loop. For analysis purposes, all information of interest is passed to the PPDU which is transferring the data to the host PC.

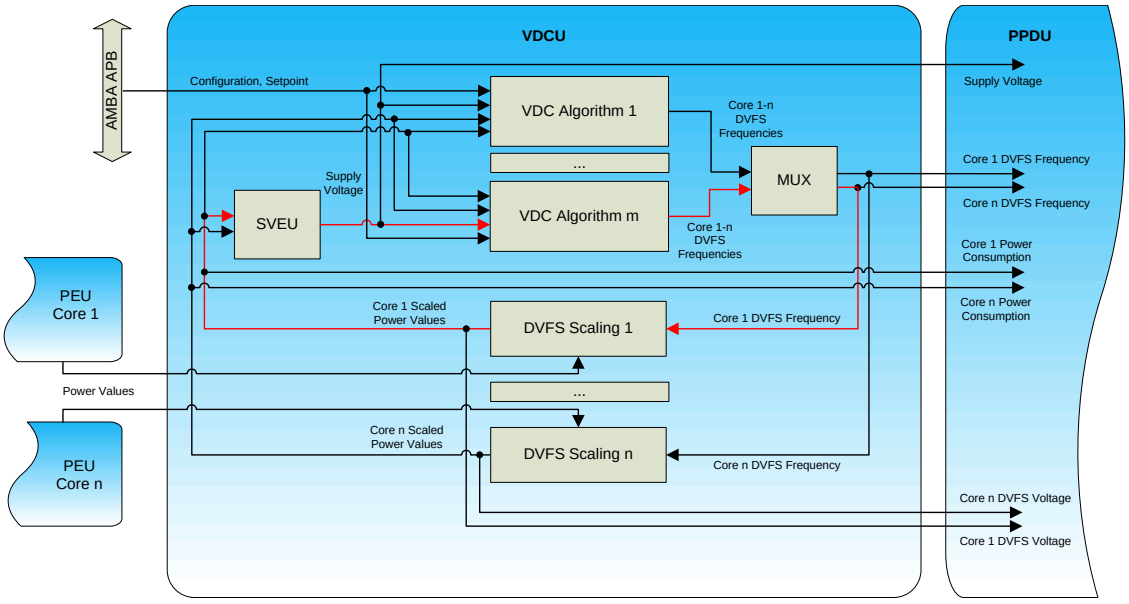

Figure 5.14: Voltage Drop Compensation Unit Implementation

Comparing the VDCU's control loop, which is shown by the red line in Figure 5.14, with the PMU's control loop, shown in the same way in Figure 5.9, shows that the VDCU implements a longer control delay. Each unit introduces one clock cycle delay. An additional two clock cycles delay is caused within the SVEU because of critical path optimizations. Furthermore, the SVEU introduces a delay caused by circuitry that mimics the behavior

of a charging/discharging capacitor. This kind of control delay (six clock cycles and the modeling of capacitor C) must be regarded by the VDC DVFS policies. In the following section, the VDC related DVFS approach is explained and the advantages/disadvantages are depicted.

5.3.6 Voltage Drop Compensation DVFS Algorithms

Voltage drop compensation algorithms perform DVFS modifications based upon the instantaneous supply voltage of the target hardware and a predefined voltage setpoint. Each policy decides on the basis of an unique strategy what processor core needs to be slowed/accelerated. All algorithms have in common the possibility to modify the DVFS step size and control delay individually. With the help of these DVFS policies, a multi-core processor system can be operated at very high clock rates and fatal supply voltage drops are prevented simultaneously. High inertia and control delays, mainly caused by a long control loop and the SVEU's modeled capacitor C, can be noted as the primary disadvantages.

Most of the currently implemented DVFS policies are allowed to change the frequency and voltage parameters for a given processor at each clock cycle. The emulation platform is operated at a clock frequency of about 30 MHz. Thus, modifications to a processor frequency can be done every 3.3ns. According to Kim et al. [KGWB08], the switching time of off-chip regulators is between $1\mu s$ and $10\mu s$. Contrariwise, on-chip regulators are capable to operate below a switching time of 10ns. The control delay at which the algorithms can update the core frequency is limited by the settling time of on-chip or off-chip regulators: this delay must be longer than the settling time and can be set by the system user.

VDC Greedy Algorithm

The principle of this algorithm is similar to the PM Greedy Algorithm, which has been presented in Section 5.3.4, but instead of observing the power consumption, this algorithm observes the supply voltage in order to make its decision. The instantaneous supply voltage level and the voltage setpoint are compared continuously. If the supply voltage drops below the setpoint, the frequencies of all cores get decreased by the same amount. Otherwise, if the supply voltage is above the setpoint, the frequencies of all cores get increased by the same amount.

VDC Power Algorithm

The principle of this algorithm is similar to the PM Power Algorithm, which has been presented in Section 5.3.4, but instead of observing the power consumption, this algorithm observes the supply voltage in order to make its decision. At first, all cores are sorted according to their instantaneous power consumption. Then, the instantaneous supply voltage level is analyzed. If it is below the voltage setpoint, the core with the highest power consumption is slowed. If the supply voltage is above the setpoint value, then the processor core with the lowest power consumption is accelerated.

VDC Priority Algorithm

The Priority algorithm corresponds to a per-core DVFS policy. Each core has a priority value assigned to itself. These values can be modified during runtime. This method favors the most prioritized core. Thus, for the case where a voltage emergency is recognized, the core with the lowest priority is slowed first to save electrical energy. If the supply voltage is above a defined setpoint value, then the high prioritized cores are accelerated ahead of all others. Figure 5.15 depicts the implementation of this DVFS method.

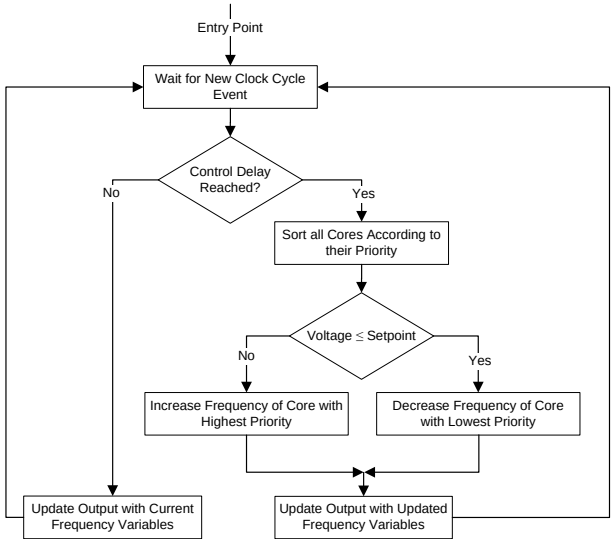

Figure 5.15: VDC Priority Algorithm

VDC Gradient Algorithm

The principle of this algorithm is similar to the PM Power Algorithm, which has been presented in Section 5.3.4, but instead of observing the power consumption, this algorithm observes the supply voltage in order to make its decision. The Gradient algorithm is a per-core DVFS method, which controls the processor cores' frequencies based on their respective power consumption gradients and the instantaneous supply voltage. At first, power gradients are calculated. The number of cycles between power consumption measurements for this computation can be adapted easily by modifying a constant in the VHDL source code. Next, the list of processor cores is sorted ascendingly by means of their gradients. Power consumption gradients are compared:

- If the gradients are different:
  - If the supply voltage is above the voltage setpoint then the core with the lowest power gradient gets an increase in its clock frequency.
  - If the supply voltage is below the voltage setpoint then the core with the highest power gradient gets a decrease in its clock frequency.

- If the gradients are equal:
  - If the supply voltage is above the voltage setpoint then the core with the lowest power consumption gets an increase in its clock frequency.
  - If the supply voltage is below the voltage setpoint then the core with the highest power consumption gets a decrease in its clock frequency.

After the affected core has received its new clock frequency, the algorithm waits an amount of time equals to an user-defined control delay in order to be able to observe a significant change in power consumption before ordering a new calculation.

VDC Gradient Delay Algorithm

This is a special version of the per-core VDC Gradient algorithm. DVFS parameters are only allowed to be changed after a certain amount of time that copes with off-chip regulators. Due to the fact that this algorithm is not allowed to change as often as a Greedy algorithm, a small offset is added to the voltage setpoint. This offset has the effect of reducing clock frequency ahead of time compared to other algorithms. Furthermore, the clock frequency for a given core can be increased by increments of 3 MHz or decreased by decrements of 16 MHz. Those combined actions does save more power in the long run at the expense of performance.

VDC Performance Algorithm

The principle of this algorithm is exactly the same as the PM Performance Algorithm, which has been presented in Section 5.3.4, except that instead of observing the power consumption, this algorithm observes the supply voltage in order to make its decision.

VDC Inverse Performance Algorithm

The principle of this algorithm is exactly the same as the VDC Performance Algorithm, which has been presented above. However, the main difference is that control actions performed for each case are reversed. If the supply voltage is below the voltage setpoint, the highest performing core is slowed. If the supply voltage is above the voltage setpoint, the lowest performing core is accelerated.

5.3.7 Hybrid Algorithms

Hybrid algorithms are neither power nor voltage management algorithms solely. They combine the best of both worlds: They are able to flatten the power consumption profile very well and additionally they are aware of the supply voltage.

Greedy Voltage/Power Algorithm

This type of chip-wide hybrid algorithm uses both voltage and power setpoint values. Figure 5.16 shows the detailed implementation scheme. It DVFS policy basically acts like the PM Greedy algorithm (see Section 5.10). Additionally, the current supply voltage level

is checked against the voltage setpoint continuously. The power setpoint is then modified slowly to direct the instantaneous supply voltage level to the desired voltage setpoint.

The main disadvantage of the power management techniques described in Section 5.3.4 is their unawareness of the supply voltage level. To ensure a maximum voltage drop safety, the power budget must be setup in these cases with very pessimistic settings. This hybrid methodology incorporates the advantage of power management algorithms (low control delay) while still being aware of the supply voltage. It permanently searches the optimal power setpoint for a previously specified voltage setpoint. Thus, the power setpoint can be initially setup very pessimistic without suffering a major performance degradation.

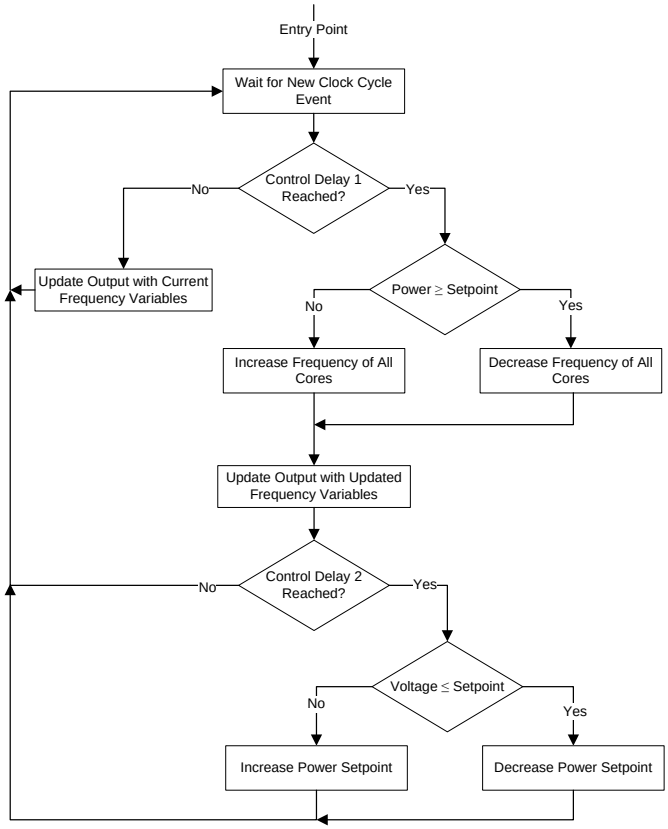

Figure 5.16: Greedy Voltage/Power Algorithm

5.4 Software Components

5.4.1 Firmware

The firmware is the so-called software that runs on the LEON3 processor cores. Embedded systems are sometimes run with an operating system like eCos [eCo10] or Linux. However, this project does not use any operating system because it represents an unneeded overhead.

Listing A.1 shows the pseudo code of a standard firmware implementation. As already noted in Chapter 4, the same source code/firmware is executed on each core of the multi-core processor system simultaneously. There are methods available to cope with this

circumstance very elegantly. Additionally, the firmware must communicate with the internal power management units of the emulation system (PEU, PMU, VDCU and PPDU) and has to execute various benchmarks. Thus, a firmware framework is implemented in a way to encapsulate benchmarking, emulation platform and LEON3 specific multiprocessor functionalities to preserve a clear structure. Figure 5.17 depicts the architecture of the constructed framework. In the following paragraphs, each part of the software framework is described precisely.

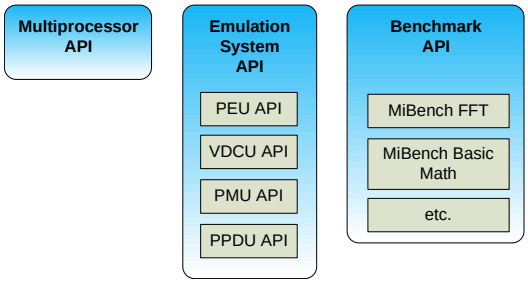

Figure 5.17: Firmware Framework

Multi Processor API

This part of the framework offers functionalities to control the LEON3 multiprocessor system. The LEON3 supports several application specific registers (asr) for control and analysis tasks. Detailed information can be found within the LEON3 SPARC functional manual [Aer10]. Very important are the asr17 and asr19 registers. With the help of asr17, the CPU index can be identified on which the program is currently running. Listing 5.1 shows the usage of the asr17.

Listing 5.1: Getting the CPU Index

```
unsigned char GetCPUIndex( void )
{
    int nCPUIndex = 0xF;

    // read asr17 and shift right
    __asm__ __volatile__( "rd     %%asr17,%0\n\t"
                          "srl    %0,28,%0" : "=&r" (nCPUIndex) : );
    return nCPUIndex;

} // GetCPUIndex
```

Register asr19 can be utilized to power down the current processor core. This is accomplished by writing a zero into it. The power down functionality is used by the firmware after a benchmark has finished. Listing 5.2 exemplifies the usage of the power down feature.

Listing 5.2: Powering down the current core

```
void PowerDownCurrentCore( void )
{
    __asm__ __volatile__( "wr %g0, %asr19" );

} // PowerDownCurrentCore
```

When the emulation system is powered up, only the core with index zero (master core) is active. Therefore, the slave processor cores must be started by the master core. For this purpose the multi processor status register of the multi processor interrupt controller unit is used (cf. [Gai09]). Its usage is exemplified in Listing 5.3. The multi processor status register can be accessed through the AMBA APB interface. A bit is assigned to each processor core and has the role to start the processor when the bit goes to one.

Listing 5.3: Powering up the slave cores

```
void PowerUpSlaveCores( void )
{
    volatile unsigned long* pMultiCoreStatusRegister;

    pMultiCoreStatusRegister = (volatile unsigned long*) 0x80000210;

    // bit nr 0 = core 0, master core
    // bit nr 1 = core 1, slave core
    // bit nr 2 = core 2, slave core

    *pMultiCoreStatusRegister |= 0x06;

} // PowerUpSlaveCores
```

Emulation System API

The emulation system API contains functions to access and configure the PEU, VDCU, PMU and PPDU. All these units support an AMBA APB interface. Therefore, a certain memory range is reserved for each unit. A unit is then accessed by performing a read or write operation within the dedicated memory range. Listing 5.4 exemplifies how the voltage setpoint value is transmitted to the VDCU.

Listing 5.4: Example of a Emulation System API Function

```
void VDCSetVoltageSetpointValue( unsigned long nSetpointValue )
{
    *((volatile unsigned long*) VDC_REGISTER_SETPOINT_SUPPLY_VOLTAGE) = nSetpointValue;

} // VDCSetVoltageTargetValue
```

Benchmark API

This API implements an abstraction layer for the underlying benchmarking framework. This framework consists of self written benchmarks and slightly modified functions of the MiBench suit. The MiBench suit can be downloaded from the Internet[10].

[10] http://www.eecs.umich.edu/mibench/ [last access 2010-05-14].

Chapter 6

Results

This chapter presents the results and experiences gained from this master project. First, the functionality of the implemented SVEU is compared to its equivalent simulation in order to verify the proper operation of this unit. Results generated by the emulation platform, named in this chapter „emulation results“, are presented. Then, the results from simulations with ModelSim, named in this chapter „simulation results“, are shown. The supply voltage, the power consumption and DVFS parameters of the target hardware are analyzed in order to determine which algorithms perform best according the criteria presented in Section 5.1.

6.1 Validation of Simulation Results by Experimentation for the SVE

The functionality of the SVEU has been checked intensively, because all voltage drop compensation DVFS policies rely on the correctness of these values. Several versions of the power supply network model have been implemented at different abstraction levels during this project (see also Figure 6.1):

- A reference SPICE model within the LT Spice IV simulator
- A MATLAB model
- A hardware integrated version

Figure 6.2 illustrates the way the final hardware integrated version is verified against the reference model. At first a simple benchmarking firmware program is developed and run within a ModelSim simulation of the emulation platform. The power consumption values gained from the PEU are converted to an electrical current. Then, the electrical current and SVEU's voltage values are written by ModelSim into log files. The log file, which contains the electrical current values, is imported by the LT Spice IV simulator. After performing a SPICE simulation of the smart card's supply power network with the given electrical current values, the resulting voltage values are exported into another log file. Next, the emulation platform voltage log file and the SPICE simulation voltage log file are analyzed and verified in MATLAB. Figure 6.3 illustrates both voltage curves as well as the computed mean squared error. The hardware implementation of the SVEU

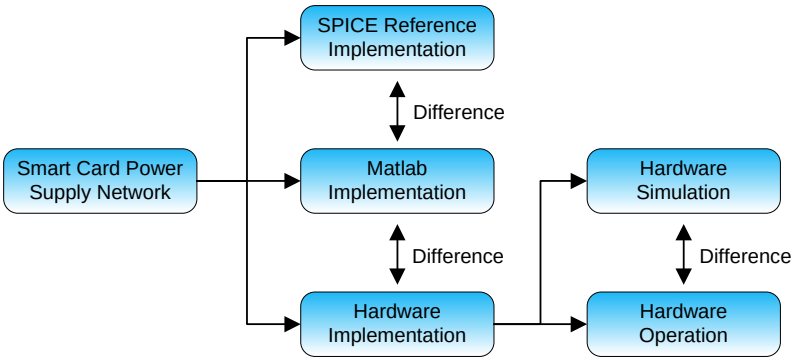

Figure 6.1: Supply Voltage Estimation Models

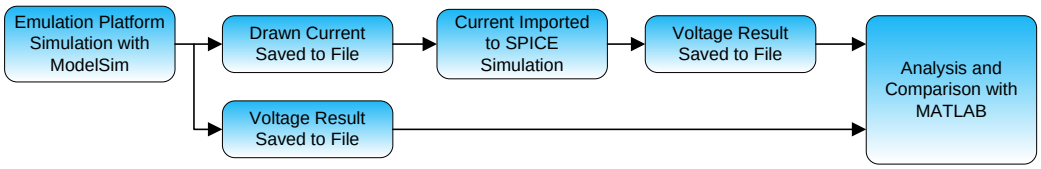

Figure 6.2: SVE Hardware Integrated Model Verification Flow

introduces a certain inexactness which is produced by rounding errors and computation delay. The delay is generated by latched signals between PE, SVE and DVFS scaling units. However, there is hardly any deviation observable, the mean squared error is in the range of 10^{-5}.

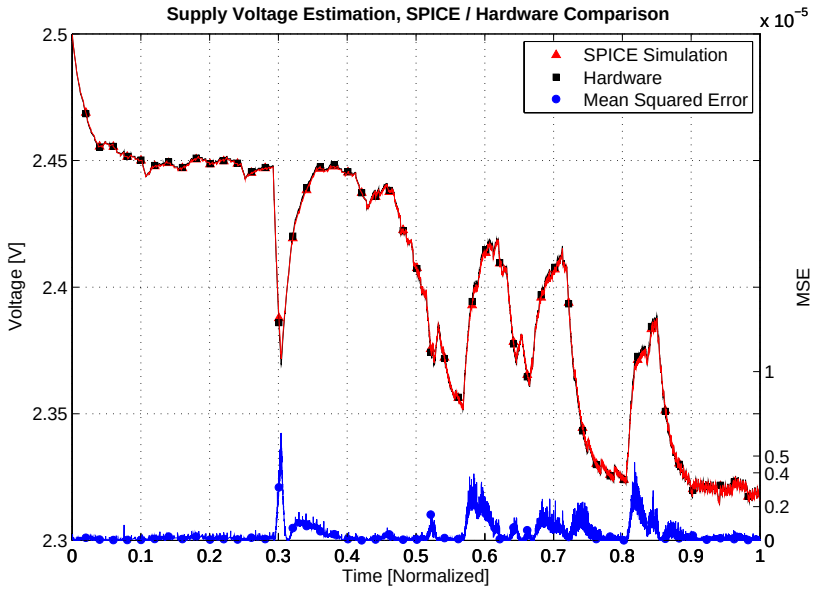

Figure 6.3: SVE Hardware Integrated Model Verification Result

The implementation of the SVEU occupies only 238 FPGA slices and computes a supply voltage value with a latency of 2 clock cycles.

6.2 Emulation Results

The hardware integrated smart card emulation platform has been tested with various benchmark programs. In these special test configurations, the analysis data is transmitted by the PPDU to the host PC. The JAVA based tool, which has been developed by M. Lackner, is used to gather, save and display the data. Figure 6.4 shows the tool in operation.

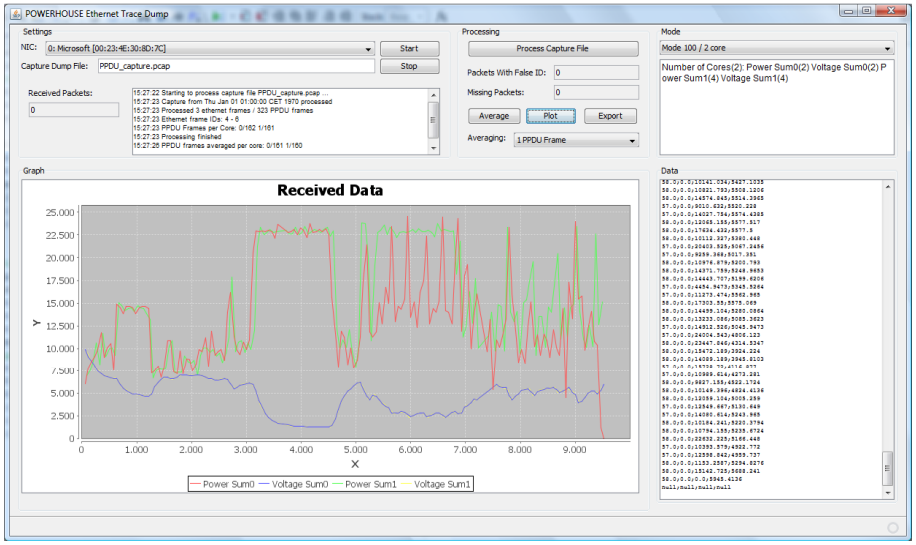

Figure 6.4: JAVA Based Data Reception and Display Tool

The received analysis data is then written by the JAVA tool into a comma-separated file. Due to the fact, that the PPDU is averaging the analysis data, this analysis data is less accurate. MATLAB scripts are used to read and post-process the content of these log files. Figure 6.5 shows the results of a benchmarking test while applying the VDC Greedy DVFS policy. The sub-figures on the left side illustrate the smart card running at a maximum speed of 31 MHz. Note the power peak, which is marked with a red arrow. This power peak causes the supply voltage to drop dramatically (also marked with a red arrow). In this certain case the functionality of the smart card would be corrupted. In contrast, the sub-figures on the right side show the smart card with the activated VDC Greedy DVFS algorithm. This DVFS policy recognizes the supply voltage drop and slows the processor cores in time, which is marked by a red arrow. As a result, the supply voltage setpoint level can be maintained and the smart card's functionality is properly preserved.

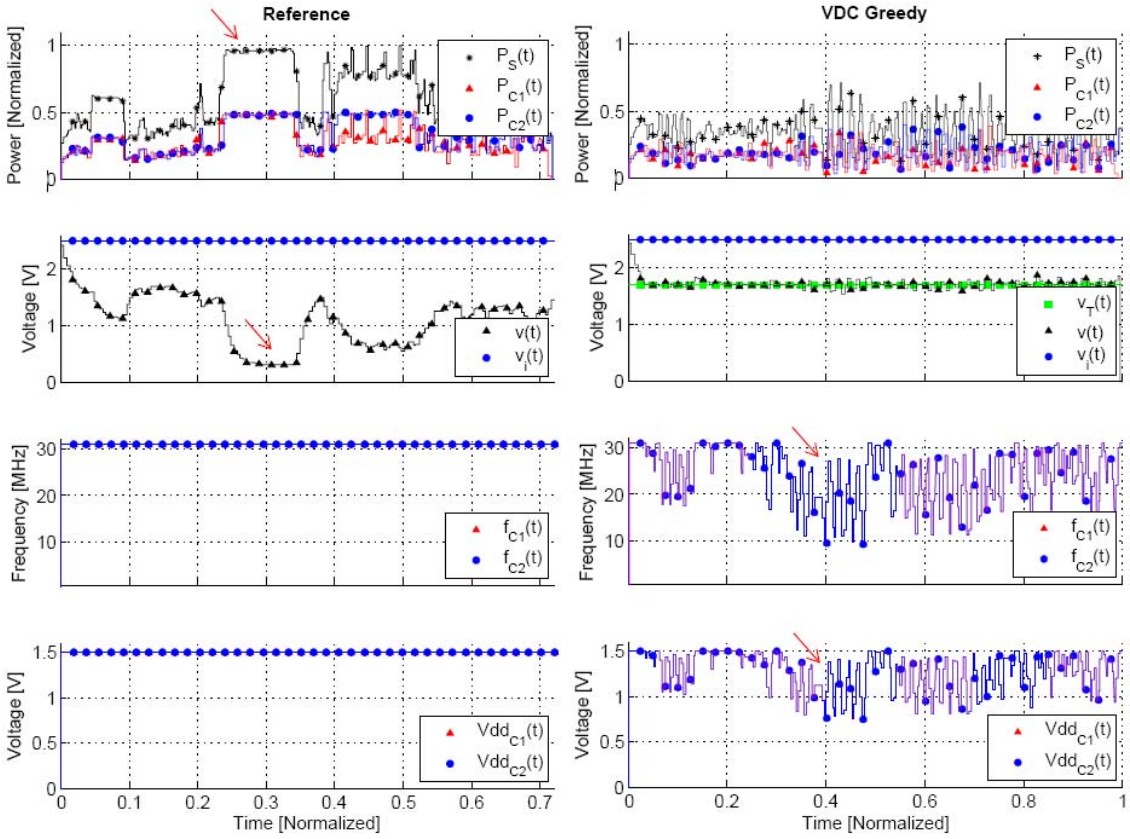

Figure 6.5: Emulation Platform in Operation

6.3 Simulation Results

This section presents the result of the emulation platform when DVFS policies are applied. In particular, the supply voltage of the target hardware as well as power consumption and DVFS voltage/frequency settings of the individual processor cores are examined. The gathered analysis data is compared to a reference target hardware setup that is operated at the maximum possible clock frequency of 31 MHz.

The evaluation of a DVFS policy is basically done in the following way: A benchmarking program is written. This program is compiled and run on the emulation platform twice. During the first run, the target hardware is operated at the maximum possible clock frequency and the analysis data is saved into a log file. This data represents the reference curves. The second run finally utilizes the specified DVFS policy with a predefined power or voltage setpoint. This gathered analysis data represents the comparison curves and is again saved to a log file. Both log files are then evaluated with MATLAB scripts in order to determine if the DVFS algorithm actually improves the reliability of the target hardware.

The diagrams presented in this section show curves which are named with certain variables. The meanings of these variables are defined as follows:

- $P_{C1}(t)$ and $P_{C2}(t)$: Power consumption of the smart card's processor core 1 and 2.

- $P_S(t)$: Power consumption sum of both processor cores. The higher the power consumption, the faster is the smart card's emergency capacitor discharged.
- $P_T(t)$: Power control setpoint/target value which is used by PM DVFS algorithms for DVFS control decisions.
- $v(t)$: Supply voltage which is applied to the target hardware, cf. Figure 2.14.
- $v_i(t)$: Voltage generated by coil L2 and the electromagnetic field, cf. Figure 2.14 and Figure 2.13.
- $V_T(t)$: Voltage control setpoint/target value which is used by VDC DVFS algorithms for DVFS control decisions.
- $I_{C1}(t)$ and $I_{C2}(t)$: Number of instructions executed by both processor cores during a specific amount of time.
- $f_{C1}(t)$ and $f_{C2}(t)$: DVFS frequencies applied to the processor cores, cf. Figure 2.14. The cores' clock frequencies have an impact on the the power consumption and the execution time of the application.
- $Vdd_{C1}(t)$ and $Vdd_{C2}(t)$: DVFS voltage applied to the processor cores, cf. Figure 2.14. A squared impact on the processor cores' power consumption is caused by these values.

6.3.1 PM DVFS Algorithms

In this section, the simulation results of the PM algorithms are presented. Basically, power management algorithms perform DVFS modifications based upon the instantaneous power consumption of the target hardware and a predefined power setpoint.

PM Greedy Algorithm

This is a very simple but effective chip-wide DVFS algorithm. If power emergencies are recognized, the clock frequencies of all cores are modified by the same amount. Figure 6.6 illustrates this fact. The power profile is flattened very well and the benchmark program is executed very quickly. The DVFS switching steps are not as precise as the PM Power's steps, because of its chip-wide strategy. However, due to the fact that only two processor cores are used during these benchmarks, this drawback is hardly noticeable.

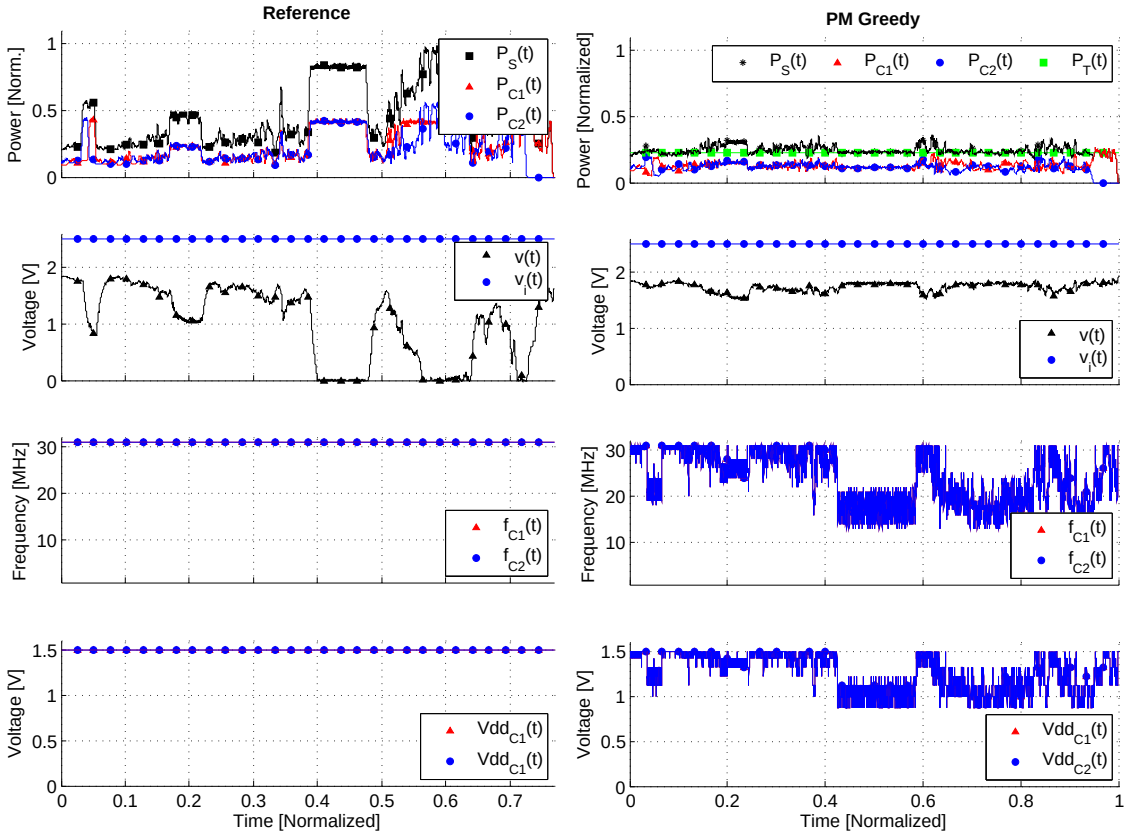

Figure 6.6: PM Greedy Algorithm Simulation Results

PM Gradient Algorithm

This per-core DVFS algorithm decides based upon the individual power gradients which processor core is accelerated or throttled. Figure 6.7 depicts the power consumption, supply voltage and DVFS parameter curves of the target hardware while this DVFS policy is applied. Basically, the PM Gradient method flattens the power profile well and executes the benchmark very fast. However, the FPGA area occupation (see Figure 6.23) is very high and the benefits compared to the other DVFS policies are relatively low. Considering these facts, it is not recommended to use this power management method.

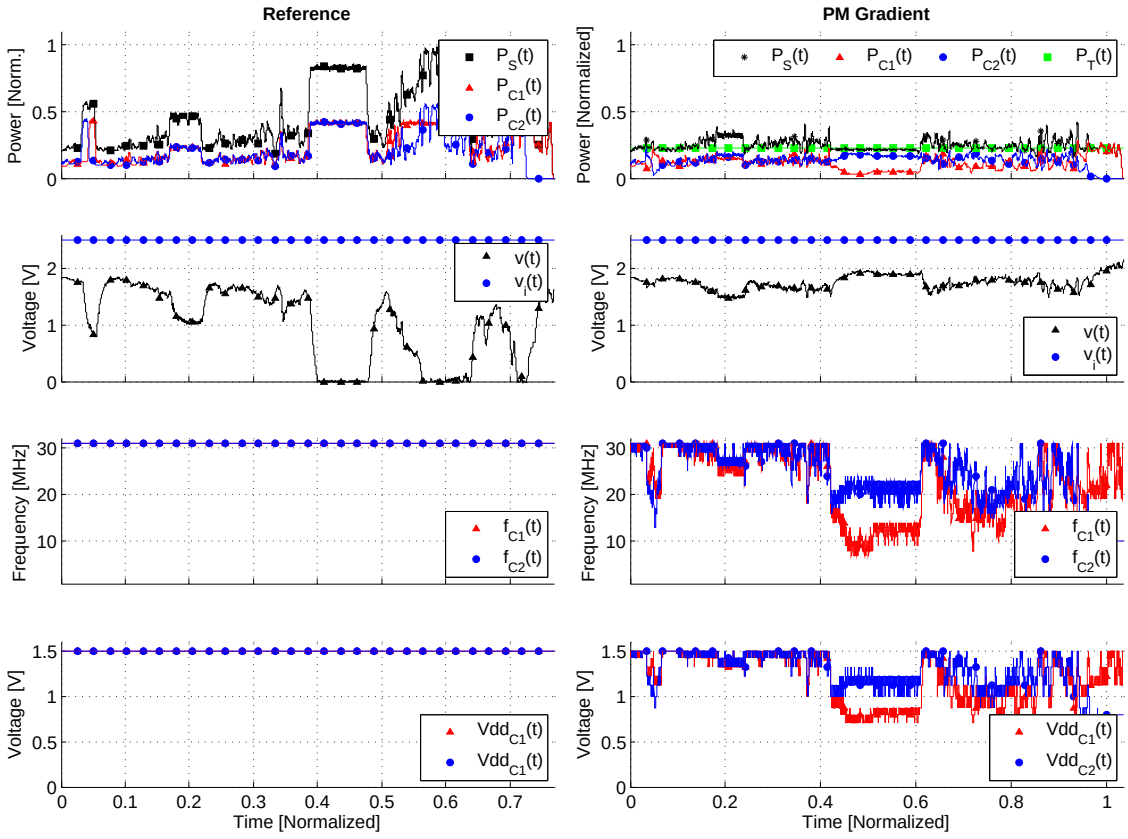

Figure 6.7: PM Gradient Algorithm Simulation Results

PM Power Algorithm

The target hardware's behavior, when applying the PM Power policy, is depicted in Figure 6.8. The fact that this is a per-core method, the frequency and voltage parameters of the processor cores are controlled individually and the total system's power consumption can be controlled very precisely. In case of an power emergency, the core with the highest power consumption is throttled. Otherwise, the core with the lowest power consumption is accelerated. Due to this special strategy, this DVFS policy generates a low power and supply voltage variation, which is shown by the sub-figures on the right side of Figure 6.8.

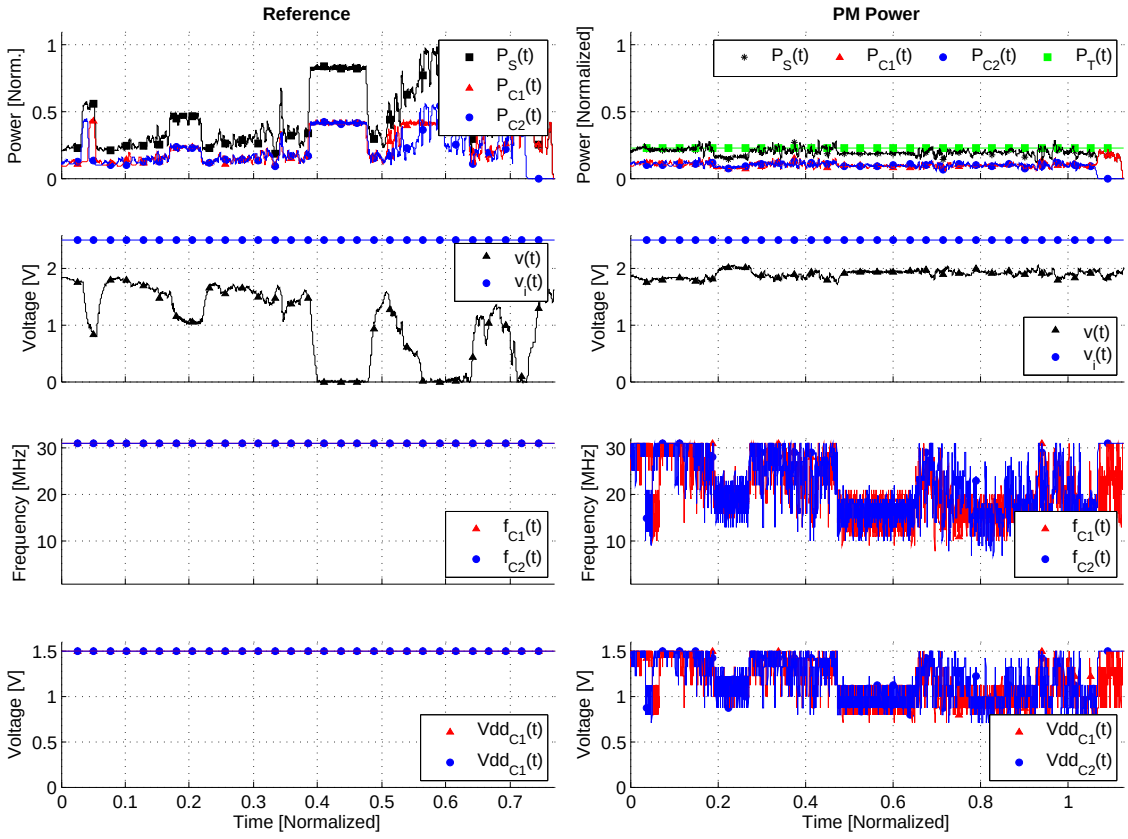

Figure 6.8: PM Power Algorithm Simulation Results

PM Performance Algorithm

Like its counterpart the VDC Performance method, DVFS decisions are performed based upon the utilization of the individual processor cores. The core with the highest utilization is favored most. Figure 6.9 illustrates the resulting power and supply voltage behavior of the smart card device. The DVFS voltage and frequency charts show that one processor core is operated mostly at high speed while the other core is throttled. Due to this specific characteristic, the available electrical energy is utilized very inefficiently. This kind of system behavior results in a bad benchmark execution time and bad power/supply voltage deviation values. These benchmarking results heavily depend on the type of application/benchmark. This DVFS policy would achieve better results if an application with huge idle time would be executed.

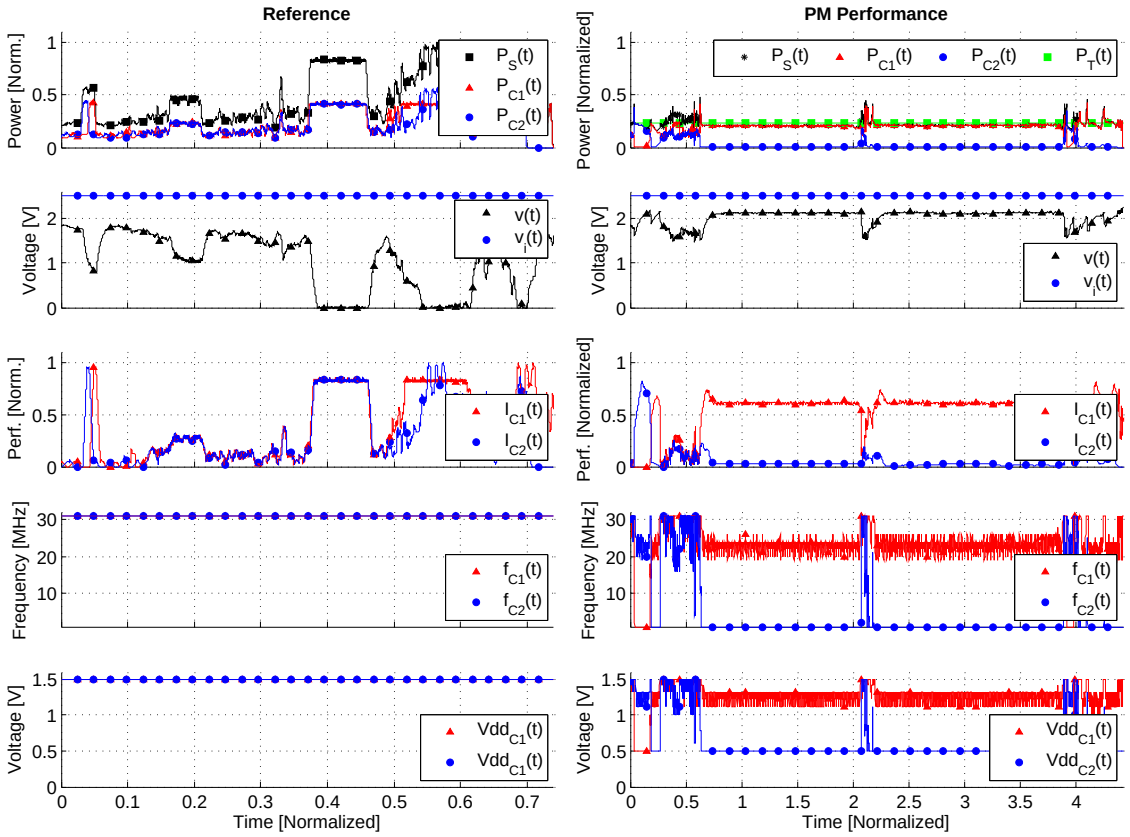

Figure 6.9: PM Performance Algorithm Simulation Results

6.3.2 VDC DVFS Algorithms

The VDC DVFS algorithms control the processor cores' voltage and frequency settings based on the estimated supply voltage from the SVEU and a predefined supply voltage setpoint. They are aware of the supply voltage and should be able to compensate any lethal drop. The emulation platform supports several distinct VDC DVFS policies. In the following paragraphs, the simulation results of the VDC DVFS algorithms are presented in detail.

VDC Greedy Algorithm

This chip-wide policy constantly searches for the optimal DVFS voltage and frequency parameters for a given supply voltage setpoint. If DVFS modifications need to be performed, then all cores are affected by the same DVFS voltage and frequency settings. Due to this special feature and the long VDCU's control loop delay, the power consumption of the target hardware can not be controlled as precisely as the per-core VDC Power algorithm or the PM algorithms do. However, the voltage setpoint can be maintained and the available electrical energy is used very economically (see Figure 6.10). Thus, the benchmarking program is executed very quickly.

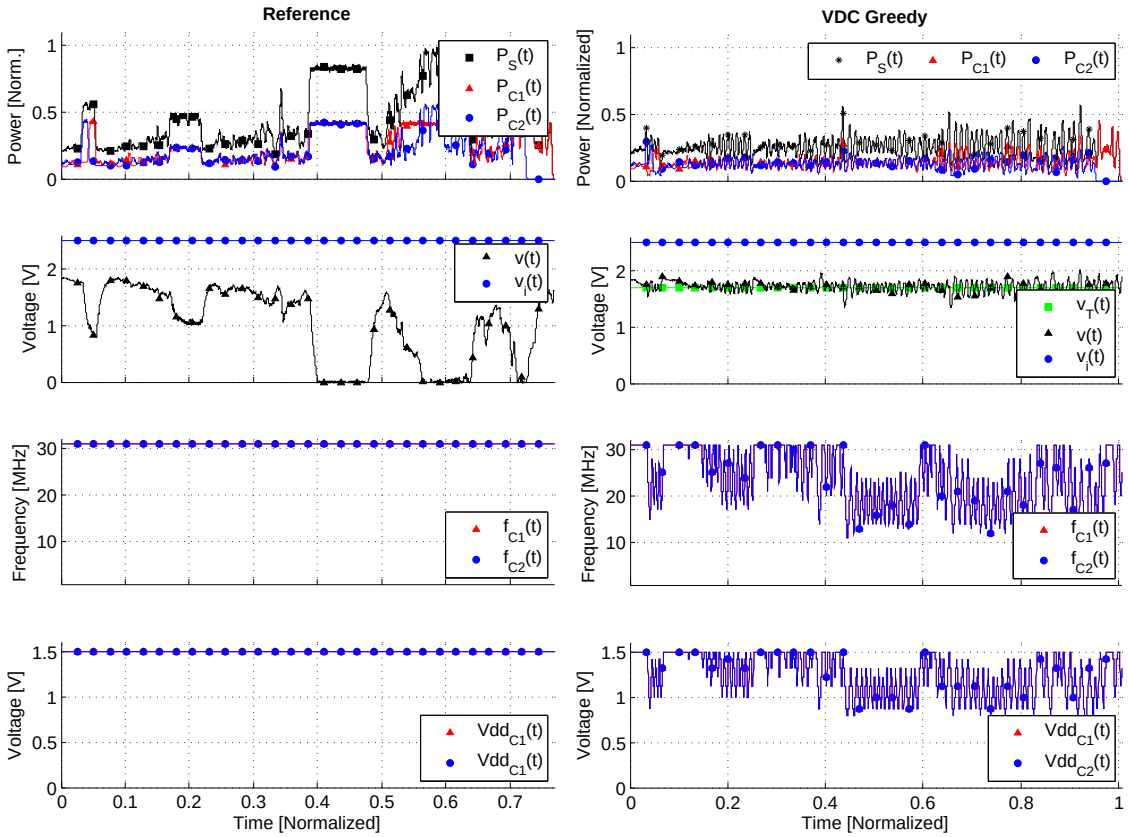

Figure 6.10: VDC Greedy Algorithm Simulation Results

VDC Power Algorithm

This is a per-core DVFS policy. Each processor core is controlled individually depending on its power consumption. If the instantaneous supply voltage is above the specified setpoint value, then the core with the lowest power consumption is accelerated. Otherwise, if a supply voltage drop below the setpoint value is recognized, the core with the highest power consumption is slowed. This per-core strategy results in a very precise power consumption and supply voltage control. Due to this fact, the power consumption variation is one of the lowest of all VDC policies (cf. Figure 6.19). The advantage of this DVFS algorithm would further increase if more processor cores were needed for a given application.

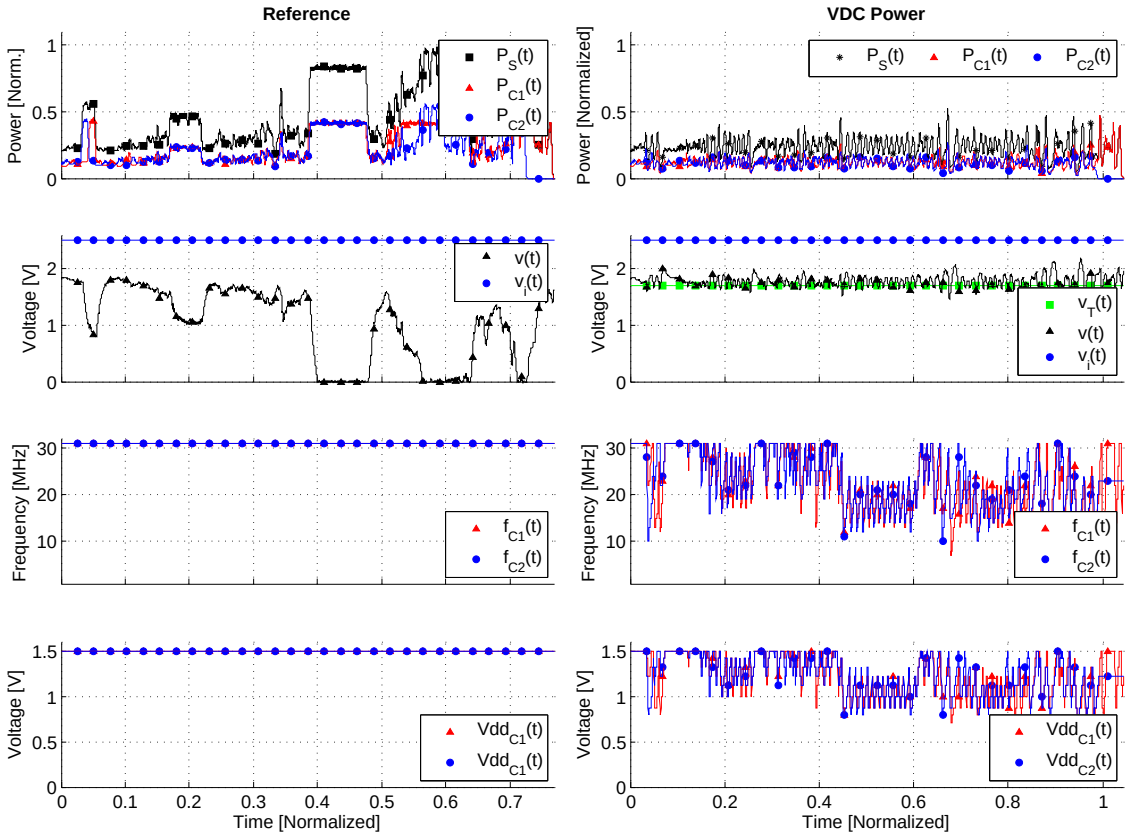

Figure 6.11: VDC Power Algorithm Simulation Results

VDC Gradient Algorithm

Figure 6.12 depicts the target hardware's power consumption, supply voltage and DVFS parameter curves when the VDC Gradient DVFS policy is utilized. This per-core policy calculates the power consumption gradient of each processor core and decides, according to this information, which core must be accelerated (lowest gradient) or throttled (highest gradient). The hardware-synthesized algorithm occupies most of the FPGA area of all integrated DVFS policies. The benchmark is run very quickly, within approximately the same amount of time as the VDC Greedy or VDC Power policies. Additionally, the power and supply voltage variations are very low. Summing up, the minor benefits gained by this DVFS algorithm do not justify the major increase in hardware costs.

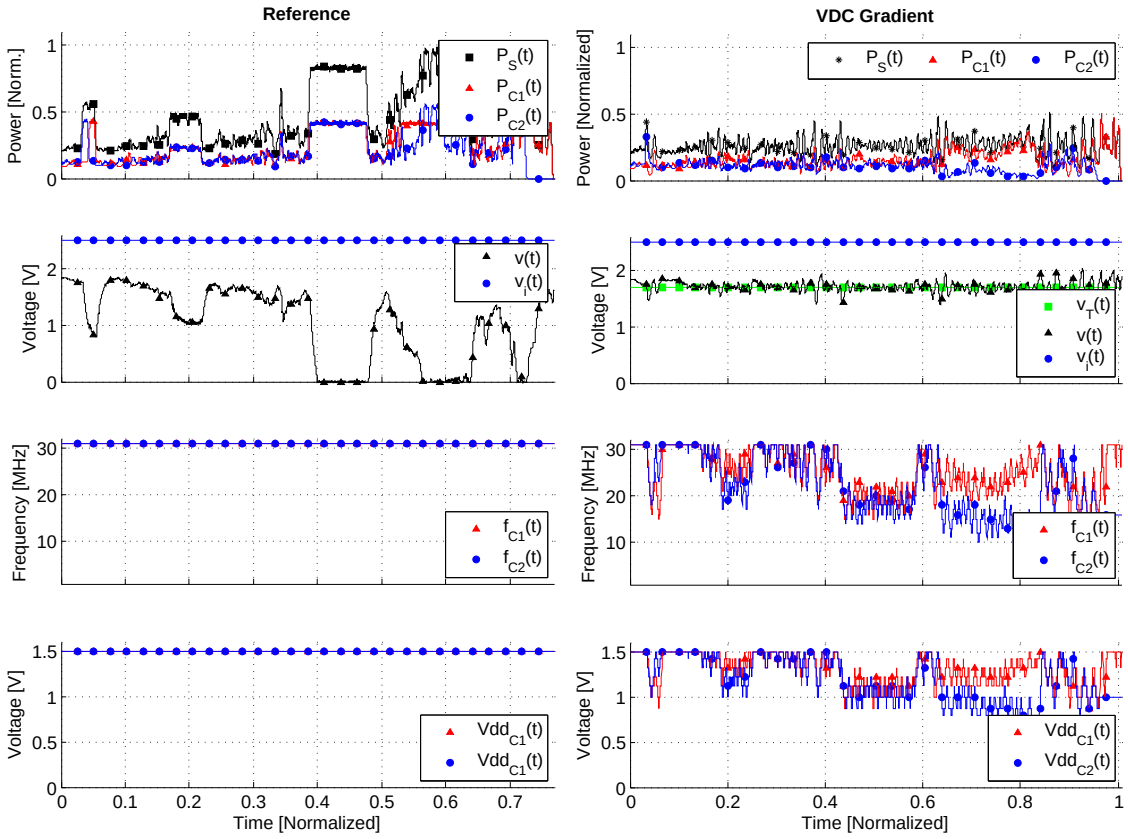

Figure 6.12: VDC Gradient Algorithm Simulation Results

VDC Gradient Delay Algorithm

This per-core DVFS algorithm simulates voltage and frequency regulators more realistically. DVFS modifications can only be performed after a certain amount of time that copes with off-chip regulators. Despite this restriction, this policy is able to prohibit a major supply voltage drop below the supply voltage setpoint. Figure 6.13 depicts the corresponding supply voltage, power consumption and DVFS parameter curves of the target hardware. Observable are the typical slow DVFS transitions. Comparing this algorithm with others reveals the impact of the delay restrictions: The benchmarks are performed at a significantly slower speed plus the supply voltage as well as the power consumption variations are the highest of all DVFS policies.

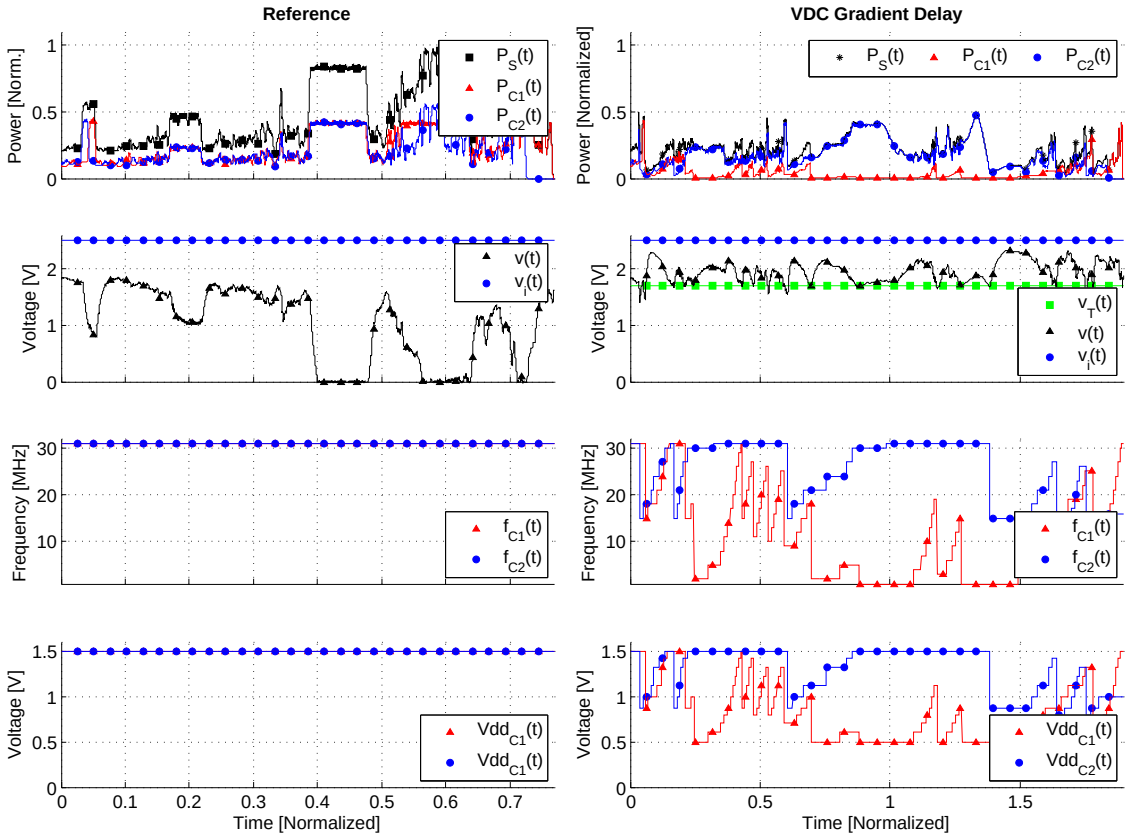

Figure 6.13: VDC Gradient Delay Algorithm Simulation Results

VDC Performance Algorithm

This per-core DVFS policy uses a performance metric (number of non-idle instructions executed during a certain period of time) to decide which core to accelerate or to throttle: The core with the highest performance is favored, the core with lowest performance is penalized. Due to the fact that a core's performance is influenced by DVFS changes, the following situation may occur: The DVFS settings are distributed between the cores very differently, like it is shown in Figure 6.14. Core 1 runs nearly the whole time at maximum speed, whilst core 2 is throttled. Due to the cubic power consumption impact of voltage and frequency (see Equation 2.6), this kind of configuration is very uneconomically way to utilize the available electrical energy (cf. Equation 2.6). This is observable in very poor benchmark result, which is depicted in Figure 6.14. These benchmarking results heavily depend on the type of application/benchmark. This DVFS policy would achieve better results if an application with huge idle time would be executed.

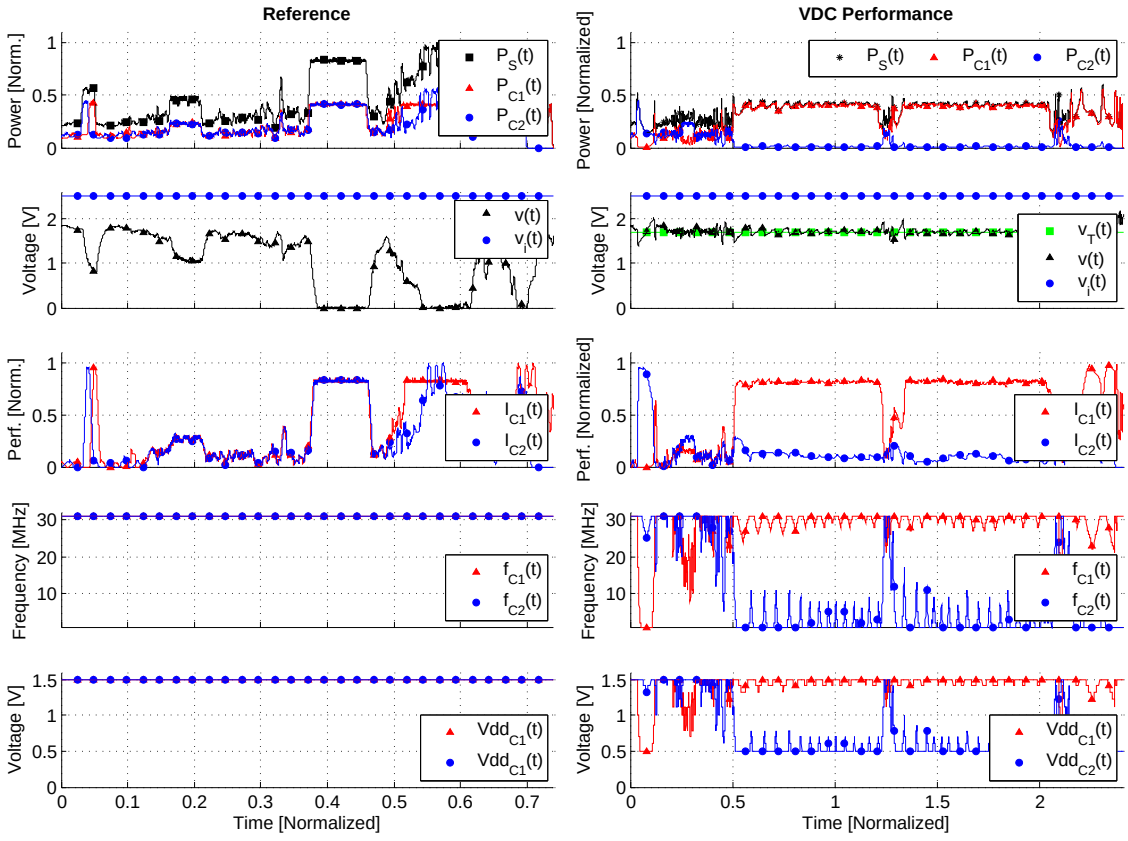

Figure 6.14: VDC Performance Algorithm Simulation Results

VDC Inverse Performance Algorithm

The Inverse Performance DVFS implementation tries to solve the VDC Performance's problem of its wasteful utilization of electrical energy. Basically, it works contrariwise to the Performance policy: The core with the lowest performance is favored, the core with highest performance is penalized. Figure 6.15 illustrates the resulting power consumption, supply voltage and DVFS parameter curves of the target hardware. With this algorithm, both cores tend to have the same level of performance. This kind of characteristic has a good impact on the benchmark results. The benchmark applications, that run on both processor cores, are executed quickly and the available electrical energy is utilized economically (cf. Figure 6.15).

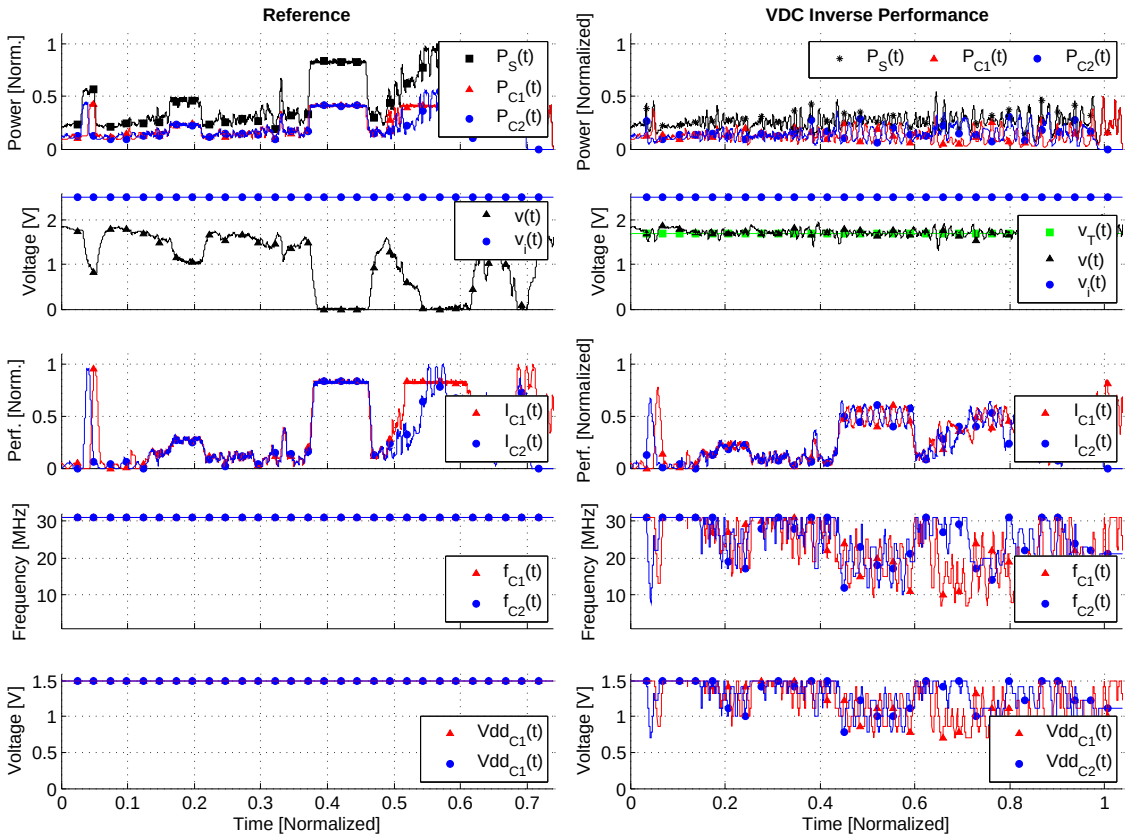

Figure 6.15: VDC Inverse Performance Algorithm Simulation Results

VDC Priority Algorithm

Figure 6.16 illustrates the power consumption, supply voltage and DVFS parameter curves of the smart card when the VDC Priority policy is applied. In this presented example, core two is assigned the lowest priority. Thus, when a supply voltage emergency is detected, core 2 is throttled. Core one is throttled only if core two is already running at the lowest possible clock frequency. The DVFS voltage and frequency curves from Figure 6.16 show that core one is operated nearly the whole time at the maximum frequency of 31 MHz whilst core two is operated in a very low frequency range. Due to the fact that voltage and frequency have a cubic impact on the power consumption (cf. Equation 2.6), the available electrical energy usage is very inefficient. Consequently, the execution time of the benchmark firmware is very high. Regardless of this underperformance, the supply voltage setpoint is maintained properly.

The VDC Priority strategy may be especially used for time-critical applications, which run on high prioritized processor cores and do not tolerate clock frequencies that are too low.

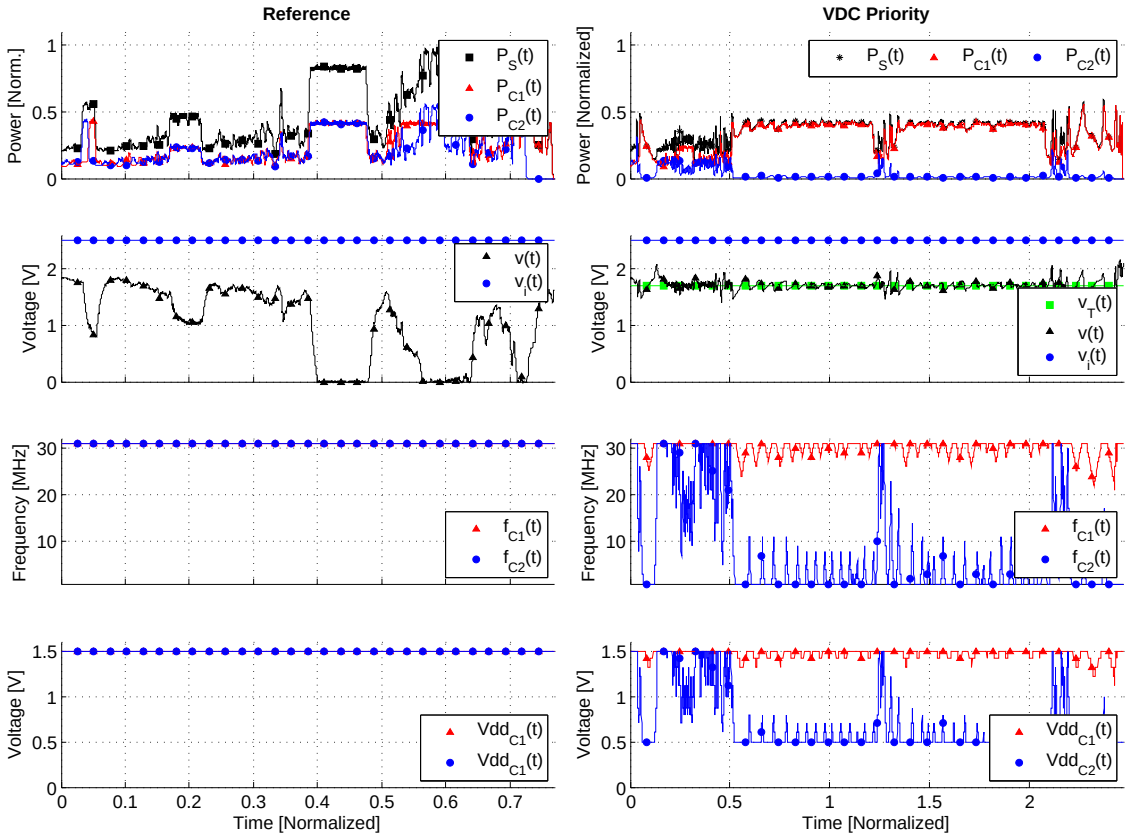

Figure 6.16: VDC Priority Algorithm Simulation Results

6.3.3 Hybrid DVFS Algorithms

Greedy Voltage/Power Algorithm

This hybrid DVFS algorithm uses both, a voltage and a power setpoint value. It tries to maintain both setpoints simultaneously. Due to the fact that this DVFS policy acts basically like the PM Greedy, the power profile flattening is done very well. Additionally, the voltage setpoint is checked on a regular basis. If the instantaneous supply voltage level is not in the range of the voltage setpoint, the power setpoint is adapted accordingly. Figure 6.17 shows the target hardware's power consumption, supply voltage and DVFS parameter curves when the Greedy Voltage/Power policy is applied. Figures 6.19 and 6.20 reveal the very low standard deviation and setpoint deviation values that are achieved. Furthermore, the algorithm's FPGA area consumption is quite low and the benchmarking firmware is executed very quickly. Considering all these facts, this DVFS policy performs best.

If per-core DVFS decision would be implemented instead of chip-wide decisions, this policy would perform even better.

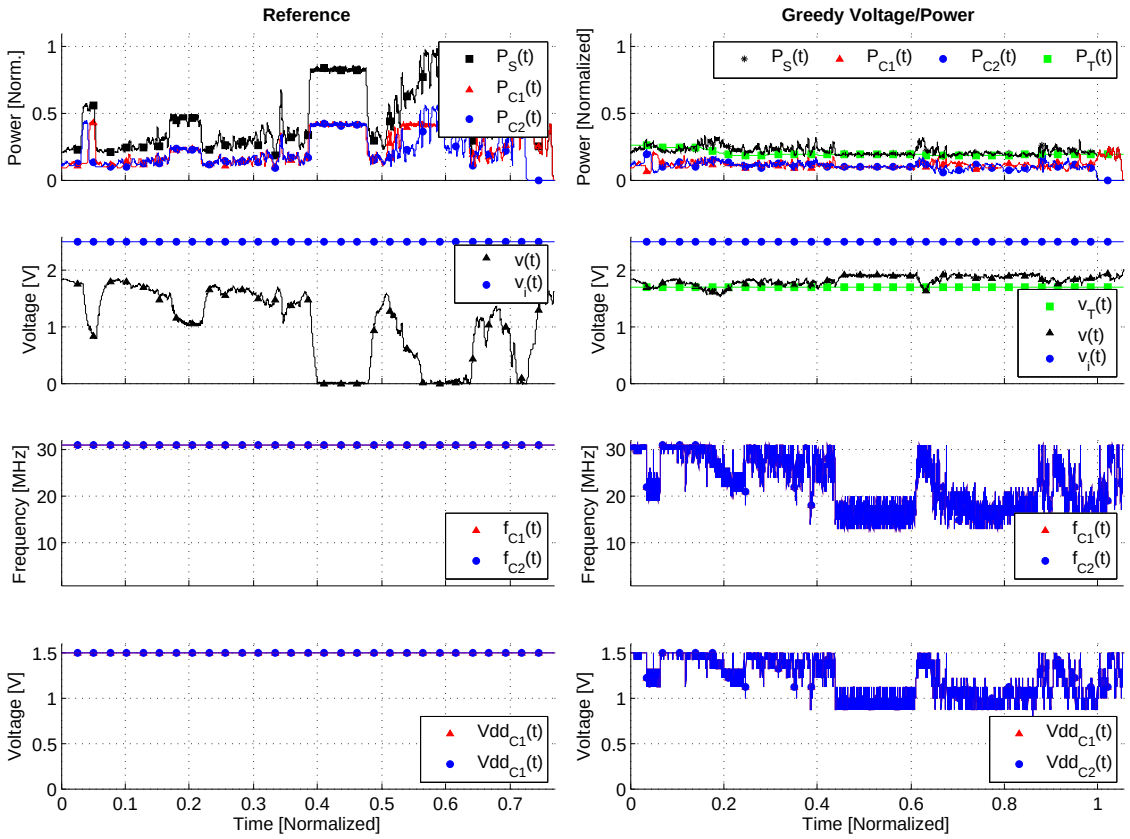

Figure 6.17: Greedy Voltage/Power Algorithm Simulation Results

6.3.4 VDC and PM DVFS Algorithms Comparison

In this section the VDC and PM algorithms are compared directly with each other. In order to increase the comparability of the presented results, the MiBench FFT and Basicmath benchmarks are executed on the individual processor cores.

Benchmark Execution Time

This test has been performed to visualize the development of the benchmark's execution time when the voltage and power setpoint values are varied. Figure 6.18 depicts these performance curves. There is one important chart area in both curves:

- The voltage setpoint equals zero and the power setpoint equals a maximum. In these operation modes all processor cores are always run at the maximum speed of 31 MHz. If the cores are operated at that speed, the supply voltage may drop to 0V continuously (illustrated for example in Figure 6.17). However, the resulting benchmark execution time acts as a theoretically lower bound and is compared to the other results.

Both curves show a similar characteristic. If the voltage setpoint is increased or the power setpoint is decreased, the execution time of the benchmark application increases exponentially. In addition, the PM Power and VDC Power DVFS policies show the best performing benchmark results. This is due to the fact, that these per-core strategies control the power consumption very precisely and therefore use the limited available electrical energy very economically. If the number of cores would increase, this per-core strategy advantage would further increase.

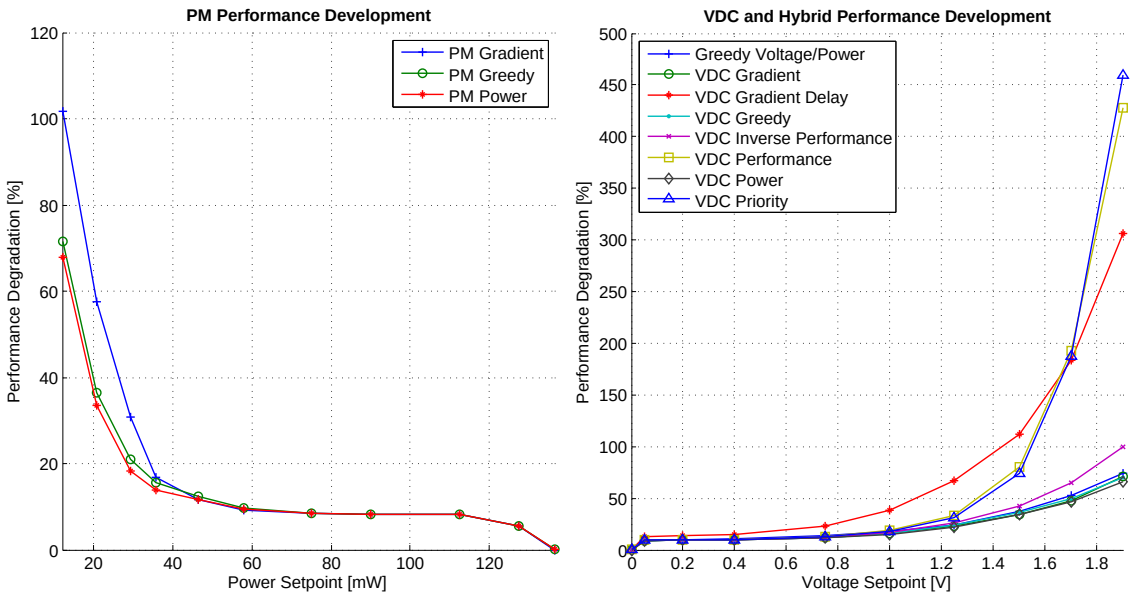

Figure 6.18: Performance Development of PM, VDC and Hybrid DVFS Policies

Standard Deviation and Deviation From Power/Supply Voltage Setpoint

Figure 6.19 shows the standard deviation of the target hardware's supply voltage and power consumption when DVFS policies are applied. These diagrams are of special interest, because the functionality of a smart card systems' RF communication may be corrupted if big power consumption changes take place (see Section 2.4). Concerning this matter, the PM policies perform better than their VDC counterparts because of the smaller control loop delay. The results show that the chip-wide VDC Gradient Delay algorithm presents the worst supply voltage deviation of all algorithms. This is mainly due to the restriction that DVFS control decisions are only allowed to be performed after a certain delay. However, this kind of behavior represents a more realistic simulation of voltage and frequency regulators.

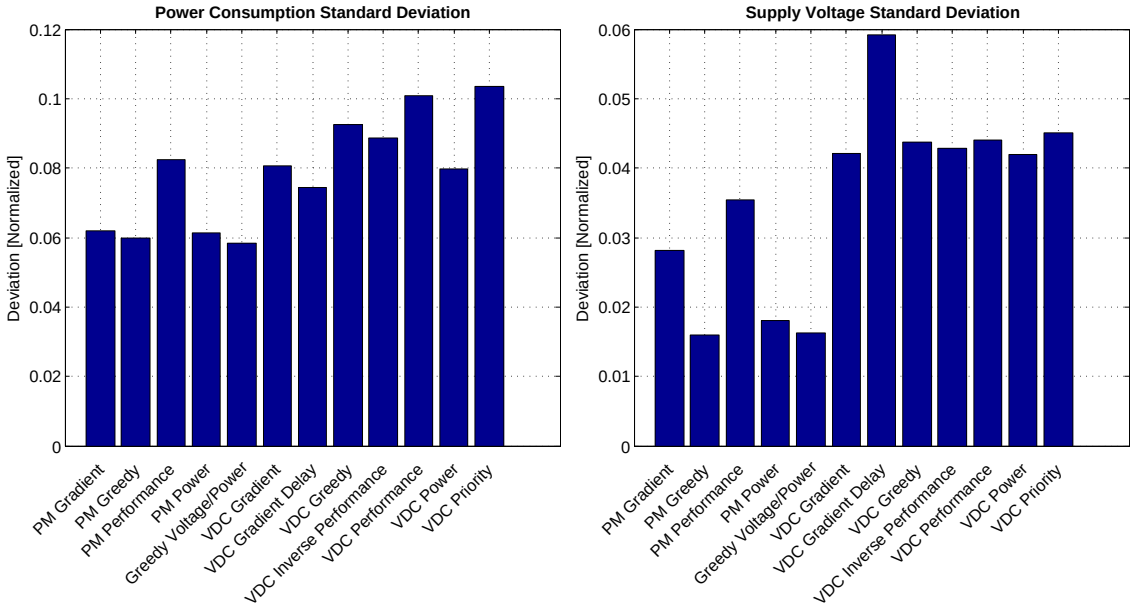

Figure 6.19: Supply Voltage and Power Consumption Standard Deviation Comparison

Figure 6.20 compares the ability of the various DVFS algorithms to maintain their power and supply voltage control setpoint values. The VDC Gradient Delay method performs poorly again due to its DVFS switching delay restrictions.

Figures 6.19 and 6.20 show, that the hybrid algorithm Greedy Voltage/Power performs well. The power and supply voltage deviation values are the lowest of all algorithms. Furthermore, the voltage control setpoint value is maintained extremely well. This algorithm combines the ability to flatten the power profile like the PM DVFS methods and simultaneously maintains a preset supply voltage level like the VDC DVFS implementations.

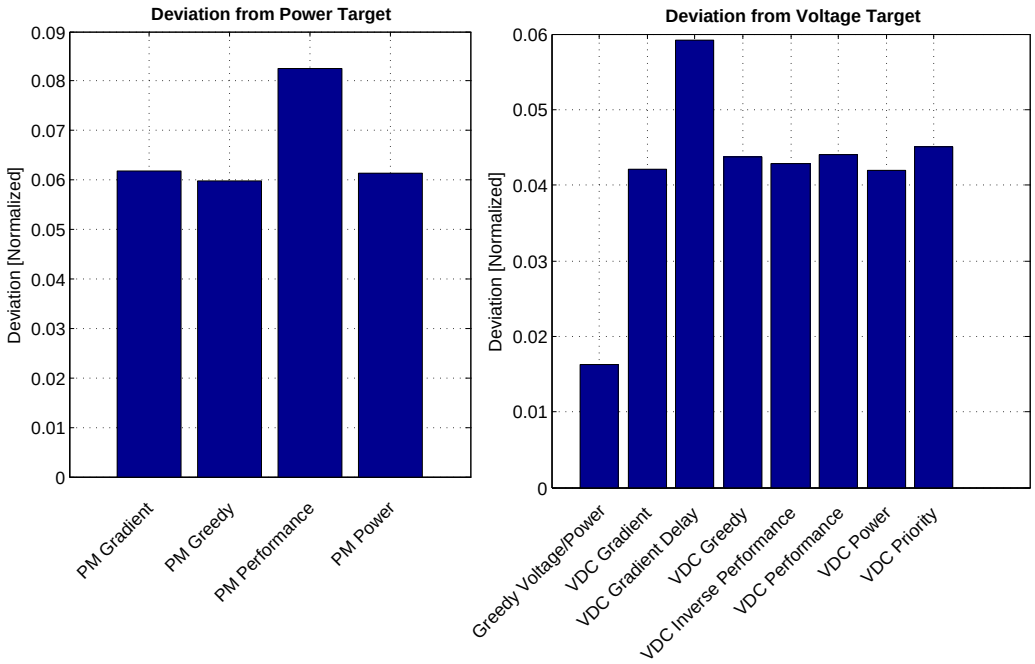

Figure 6.20: Deviation from Voltage and Power Setpoints

Magnetic Field Intensity Changes

In order to explicitly present the pros and cons of PM and VDC DVFS policies, the following test is being carried out. For a more comprehensible illustration, the voltage $v_i(t)$ is altered. $v_i(t)$ is generated by the electromagnetic field. If the intensity of the electromagnetic field changes, then voltage $v_i(t)$ varies according to Lorenz law (cf. Figure 2.14). Figure 6.21 compares the behavior of the target hardware when PM and VDC DVFS policies are applied and the voltage $v_i(t)$ changes. It is observable that the VDC Greedy algorithm manages to compensate the magnetic field changes perfectly. In contrast, the PM Greedy algorithm is unable to cope with this situation. In this case, variations of voltage $v_i(t)$ influence $v(t)$ directly. Thus, $v(t)$, which is applied to the target hardware, drops periodically below the lethal threshold of 1.4V. This behavior is explained by the supply voltage unawareness of all PM DVFS policies. To cope with these voltage drops, the PM control setpoint value needs to be set very pessimistically low, which results in performance degradations.

This test shows clearly, that PM DVFS algorithms are unable to maintain a minimum voltage setpoint if environment properties change or unpredictable situations occur.

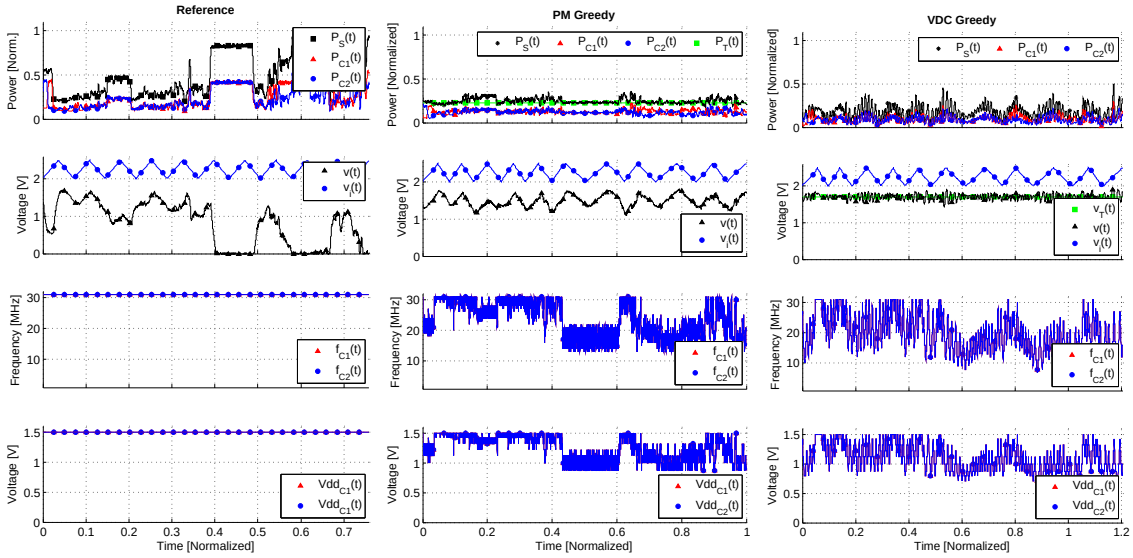

Figure 6.21: Simulation Results of Changing Magnetic Field Intensity Test

6.4 FPGA Area Consumption

Figure 6.22 depicts the FPGA area consumption of the total emulation platform. 37,7% of the FPGA area is occupied by additional hardware components: the master core (18%) and PEU, PMU, VDCU and PPDU (19,7%). This overhead is quite high. However, during this master project a power consumption and supply voltage emulation/exploration platform is constructed, which is not integrated into a final product. This emulation platform is used to evaluate and explore a target hardware's power and supply voltage behavior in early design stages. Therefore, the high amount of additional die area is actually not a problem at all.

The FPGA area consumption of the individual DVFS policies is compared and illustrated in Figure 6.23. This chart shows that PM and VDC Gradient algorithms occupy a huge amount of slices but provide hardly any matchable advantages compared to the PM and VDC Greedy algorithms. In contrast, the PM and VDC Greedy algorithms require hardly any FPGA area and additionally generate well performing benchmarking results.

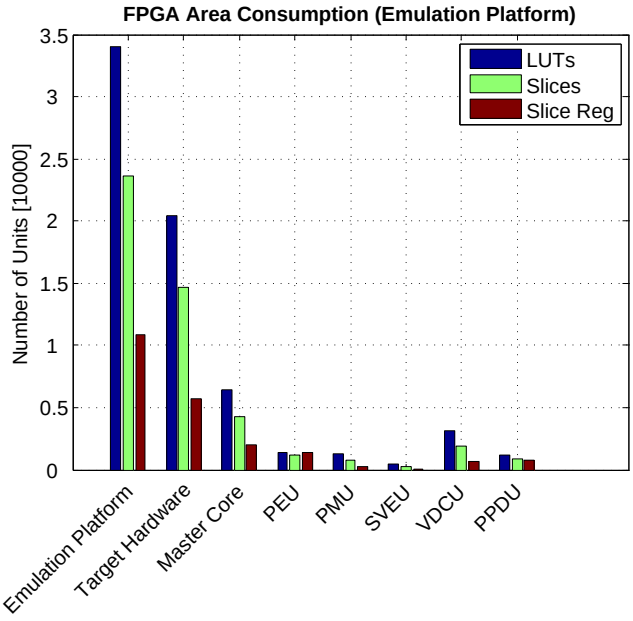

Figure 6.22: FPGA Area Consumption of the Emulation Platform

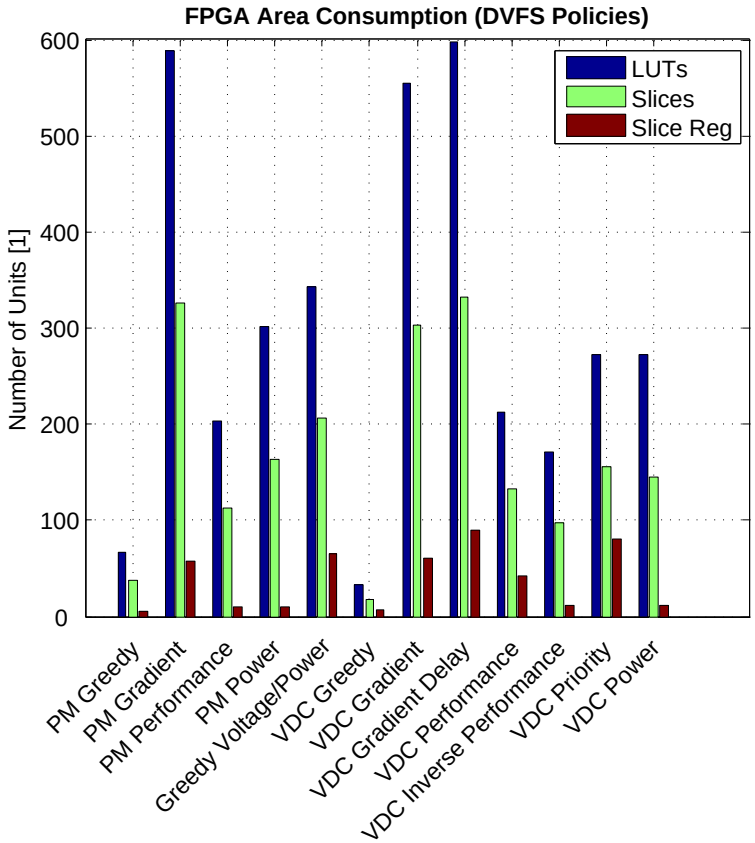

Figure 6.23: FPGA Area Consumption of the DVFS Policies

Chapter 7

Conclusion

Power-aware computing addresses the problem that electronic circuits and algorithms are growing exponentially in their complexity. Thus, the power dissipation of electronic circuits also increases rapidly, which is especially problematic for mobile or battery operated applications. Power and supply voltage analysis techniques are used to measure or estimate the instantaneous dissipated power of devices and their supply voltage level. The gathered information can then be used to optimize the system's power consumption regarding certain constraints. Power and supply voltage emulation is an estimation based analysis technique utilizing hardware acceleration to deliver cycle accurate power and supply voltage values in real time.

During this master project, a power and supply voltage emulation system for a smart card target hardware has been built and integrated into an FPGA. For this purpose, several LEON3 processor cores are used to emulate the multi-core processor system of a smart card. A *supply voltage estimation unit* has been developed based on a smart card's supply network model. Together with an already developed *power estimation unit*, a smart card power and supply voltage emulation and exploration platform has been constructed. Operating the smart card emulation and exploration platform can clearly reveal power bugs in the design under test. These design problems can thereupon be corrected by the development team even in early design stages, long before the chip's tape-out. In the second part, this exploration platform has been enhanced with dynamic voltage and frequency scaling functionalities to increase the target hardware's robustness against power peaks and supply voltage drops. Benchmarks with different power and supply voltage setpoint values have been carried out. MATLAB scripts have been written to visualize and evaluate the power and supply voltage behavior of the target design as well as the performance of the benchmarking applications while DVFS policies are applied. Based on this data, the best fitting DVFS strategy can be chosen for a specific smart card application. Tests have shown that a DVFS enhanced SMP smart card design can be operated at a stabilized supply voltage of 1.7V while degrading the application's execution time by only about 50%. This value of 50% is compared to the smart card system running continously at 31 MHz and allowing supply voltage drops down to 0 V. The presented emulation platform supports twelve different power management strategies. It incurs average estimation errors of 8.4% (PEU) and 2% (SVEU power network model) while occupying 37.7% of the total FPGA area. Furthermore it is depicted that an ASIC or final smart card hardware can be enhanced with the presented power management

methods (see Section 7.1). 10.1% of the die area would be occupied by power analysis and management units.

7.1 Future Work

The current version of the smart card emulation and exploration platform performs well in detecting power bugs. However, there is still a lot of improvement potential available. The most promising ideas are presented in the following sections.

Power and Supply Voltage Management Enhanced ASIC

The main idea of this approach is to enhance an ASIC or final smart card product with power management and supply voltage drop compensation functionalities, which are presented in this master thesis. To make this approach feasible, only the most important power management parts of the emulation platform are used and integrated. All other insignificant and area intense components are omitted. Therefore, the hardware overhead is reduce dramatically. The design of the final smart card product consists of:

- The smart card target hardware, which consists of two processor cores and peripherals (14676 slices).
- Two PEUs to estimate the power consumption of each processor core (1156 slices).
- A VDCU which estimates the supply voltage and implements only the VDC Greedy DVFS policy to cope with voltage drops (494 slices).
- Alternatively another, more sophisticated DVFS policy can be chosen, which is better suited for the designated target application.

With this pessimistic laid-out design (a lot of area improvements are still possible) a hardware overhead of 10.1% is achieved, but the hardware's robustness against power peaks and supply voltage drops is increased enormously. Therefore, the smart card's very area intense emergency capacitors could be reduced drastically.

Considering all these facts, the accumulated hardware amount and consequently the manufacturing costs would decrease and the product's application robustness would increase.

Additional DVFS Algorithms

There are several promising DVFS algorithms and power management strategies, that have been proposed by the scientific community but have not been implemented in this project. Among others, there are:

- An algorithm based on control theory, which implements a proportional-integral-differential controller.
- Oracle based solutions, which are able to calculate the future power consumption.
- Neuronal network controller approaches.
- Policies which are based on adaptive algorithms.

PPDU Improvements

The current version of the PPDU is used in conjunction with a 100 MBit Ethernet core. Due to the fact that the 100 MBit Ethernet is not able to transfer all analysis data produced during each clock cycle, the data must be averaged. In case a faster 1 GBit core would be used, the analysis data could be sent more accurately to the host PC.

Appendix A

Source Code

Listing A.1: firmware.c

```
int main( int argc, char *argv[] )
{
    unsigned int nCPUIndex = GetCPUIndex();

    if( nCPUIndex == 0 )
    {
        // master core
        StartUnitsOfEmulationSystem();
        PowerUpSlaveCores();

        // wait until all benchmarks are finished
        WaitUntilAllSlaveCoresArePoweredDown();

        // e.g. the PPDU needs to be stopped
        StopUnitsOfEmulationSystem();

        // finally power down the master core
        PowerDownCurrentCore();
    }
    else if( nCPUIndex == 1 )
    {
        // slave core, execute the benchmark
        FFTBenchmark();

        // benchmark finished, power down the slave core
        PowerDownCurrentCore();
    }
    else if( nCPUIndex == 2 )
    {
        // slave core, execute the benchmark
        BasicmathBenchmark();

        // benchmark finished, power down the slave core
        PowerDownCurrentCore();
    }

} // main
```

Appendix B

Tables

Signal Name	Power [W]	Power [1]
/testbench/cpu/l3/cpu__0/u0/cmem0/dme/dd0__0/ddata0/write	0,0038914	39
/testbench/cpu/l3/cpu__0/u0/cmem0/ime/im0__0/idata0/write	0,0015259	15
/testbench/cpu/l3/cpu__0/u0/cmem0/ime/im0__0/itags0/write	0,0011244	11
/testbench/cpu/l3/cpu__0/u0/p0/m1/c0mmu/dcache0/r.dstate(2)	0,0016642	17
/testbench/cpu/l3/cpu__0/u0/p0/m1/c0mmu/dcache0/r.dstate(3)	0,0017490	18
/testbench/cpu/l3/cpu__0/u0/p0/m1/c0mmu/dcache0/r.read	0,0003930	4
/testbench/cpu/l3/cpu__0/u0/p0/m1/c0mmu/icache0/r.istate(0)	0,0021274	21
/testbench/cpu/l3/cpu__0/u0/p0/m1/c0mmu/icache0/rl.write	0,0001293	1
/testbench/cpu/l3/cpu__0/u0/p0/m1/c0mmu/icache0/v_pe_signal.holdn	0,0001077	1
/testbench/cpu/l3/cpu__0/u0/p0/m1/c0mmu/mmudci.transdata.read	0,0006370	6
/testbench/cpu/l3/cpu__0/u0/p0/mgen/div0/holdn	0,0048107	48
/testbench/cpu/l3/cpu__0/u0/p0/mgen/div0/r.state(0)	0,0015865	16
/testbench/cpu/l3/cpu__0/u0/p0/mgen/div0/r.state(2)	0,0005605	6
/testbench/cpu/l3/cpu__0/u0/p0/mgen/div0/divo.nready	0,0016493	17
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.a.ctrl.rett	0,0003364	3
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.a.ctrl.wreg	0,0000718	1
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.a.rsel1(2)	0,0002039	2
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.a.rsel2(0)	0,0002209	2
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.a.rsel2(1)	0,0004415	4
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.e.alusel(0)	0,0002676	3
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.e.alusel(1)	0,0008184	8
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.e.ctrl.wicc	0,0011431	11
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.e.ctrl.wreg	0,0005500	6
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.e.cwp(1)	0,0004244	4
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.e.cwp(2)	0,0011237	11
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.e.ldbp2	0,0004602	5
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.m.ctrl.rett	0,0023060	23
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.m.ctrl.wicc	0,0015579	16

Table B.1: PEU Power Model, Part 1/2

Signal Name	Power [W]	Power [1]
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.m.divz	0,0010553	11
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.m.mul	0,0010763	11
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.x.ctrl.rd(5)	0,0005725	6
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.x.ctrl.rd(7)	0,0002313	2
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.x.ctrl.rett	0,0060096	60
/testbench/cpu/l3/cpu__0/u0/p0/iu0/r.x.ctrl.wicc	0,0012771	13
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.a.ctrl.pv	0,0005603	6
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.a.ctrl.rd(0)	0,0008125	8
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.a.ctrl.rd(2)	0,0005009	5
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.a.rsel1(0)	0,0001729	2
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.a.rsel1(1)	0,0009330	9
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.a.rsel2(1)	0,0002502	3
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.a.rsel2(2)	0,0001496	1
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.e.ctrl.annul	0,0001781	2
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.e.ctrl.rd(0)	0,0006293	6
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.e.ctrl.rd(2)	0,0005878	6
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.m.ctrl.ld	0,0002350	2
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.m.ctrl.rd(0)	0,0002956	3
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.m.ctrl.rd(2)	0,0002619	3
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.m.ctrl.rd(7)	0,0000538	1
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.m.dci.write	0,0011780	12
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.x.ctrl.rd(0)	0,0002727	3
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.x.ctrl.rd(3)	0,0001177	1
/testbench/cpu/l3/cpu__0/u0/p0/iu0/v_pe_signal.x.ctrl.rd(7)	0,0001116	1
/testbench/cpu/l3/cpu__0/u0/rf0/we	0,0010513	11
I-Cache Read Hit (4kB)	0,0246170	247
I-Cache Read Miss (4kB)	0,0268500	269
I-Cache Write Hit (4kB)	0,0258000	259
I-Cache Write Miss (4kB)	0,0268500	279
D-Cache Read Hit (4kB)	0,0246170	247
D-Cache Read Miss (4kB)	0,0268500	269
D-Cache Write Hit (4kB)	0,0258000	259
D-Cache Write Miss (4kB)	0,0268500	279
RFMem Read	0,0037583	38
RFMem Write	0,0039389	39

Table B.2: PEU Power Model, Part 2/2

Bibliography

[Aer10] Aeroflex. *UT699 LEON 3FT/SPARC V8 MicroProcessor Functional Manual*, 2010.

[Age09] International Energy Agency. Gadgets and Gigawatts, Policies for Energy Efficient Electronics, 2009.

[And98] R. Andraka. A survey of CORDIC algorithms for FPGA based computers. In *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, 1998.

[ARM99] ARM. *AMBA Specification (Rev 2.0)*, 1999.

[ASH05] E. Alon, V. Stojanovic, and M. Horowitz. Circuits and Techniques for High-Resolution Measurement of On-Chip Power Supply Noise. In *IEEE Journal of Solid-State Circuits*, volume 40, pages 820–828, 2005.

[BBDM00] A. Bogliolo, L. Benini, and Giovanni. De Micheli. Regression-based RTL power modeling. In *Transactions on Design Automation of Electronic Systems*, pages 337–372, 2000.

[BBH01] G. Bai, S. Bobba, and I.N. Hajj. Static Timing Analysis Including Power Supply Noise Effect on Propagation Delay in VLSI Circuits. In *Proceedings of the 38th Design Automation Conference*, pages 295–300, 2001.

[BE99] A. Bellaouar and M. Elmasry. *Low-Power Digital VLSI Design: Circuits and Systems*. Kluwer Academic Publishers, 1999.

[BGKM04] A. Boudabous, F. Ghozzi, M.W. Kharrat, and N. Masmoudi. Implementation of Hyperbolic Functions Using CORDIC Algorithm. In *The 16th International Conference on Microelectronics*, pages 738–741, 2004.

[BGS+10] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid. Automated Power Characterization for Run-Time Power Emulation of SoC Designs. In *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD)*, pages 587–594, 2010.

[BHB+08] R. Bergamaschi, G. Han, A. Buyuktosunoglu, H. Patel, I. Nair, G. Dittmann, G. Janssen, N. Dhanwada, Zhigang Hu, P. Bose, and J. Darringer. Exploring Power Management in Multi-Core Systems. In *Asia and South Pacific Design Automation Conference*, pages 708–713, 2008.

[Bor99] S. Borkar. Design challenges of technology scaling. In *Micro, IEEE*, volume 19, pages 23–29. IEEE Computer Society, 1999.

[BTD+02] M. Badaroglu, K. Tiri, S. Donnay, P. Wambacq, I. Verbauwhede, G. Gielen, and H. De Man. Clock Tree Optimization in Synchronous CMOS Digital Circuits for Substrate Noise Reduction Using Folding of Supply Current Transients. In *Proceedings of the 39th annual Design Automation Conference*, pages 399–404, 2002.

[CRA05] J. Coburn, S. Ravi, and Raghunathan A. Power Emulation: A New Paradigm for Power Estimation. In *Design Automation Conference, Proceedings*, pages 700–705, 2005.

[CRR05] J. Coburn, S. Ravi, and A. Raghunathan. Hardware Accelerated Power Estimation. In *Design, Automation and Test in Europe, Proceedings*, volume 1, pages 528–529, 2005.

[Cyg10] Cygwin. Cygwin. `http://www.cygwin.com/`, 2010.

[DBS06] J-P. Deschamps, G.J.A Bioul, and G.D. Sutter. *Synthesis of Arithmetic Circuits - FPGA, ASIC and Embedded Systems*. John Wiley & Sons, Inc., 2006.

[Dru10] N. Druml. Power Emulation on a LEON3 Platform. Technical report, Graz University of Technology, 2010.

[eCo10] eCos. eCos. `http://ecos.sourceware.org/`, 2010.

[EL04] M.D. Ercegovac and T. Lang. *Digital Arithmetic*. Morgan Kaufmann Publishers, 2004.

[Fin03] K Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, 2nd edition, 2003.

[Gai09] Aeroflex Gaisler. *GRLIB IP Core Users Manual Version 1.0.21*, 2009.

[Gai10] Aeroflex Gaisler. Aeroflex Gaisler. `http://www.gaisler.com/`, 2010.

[GAT02] E. Grochowski, D. Ayers, and V. Tiwari. Microarchitectural simulation and control of di/dt-induced power supply voltage variation. In *Proceedings of the 8th International Symposium on High Performance Computer Architecture*, pages 7–16, 2002.

[GBH+09] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss. An Emulation-Based Real-Time Power Profiling Unit for Embedded Software. In *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, pages 67–73, 2009.

[GBH+11] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss. Supply Voltage Emulation Platform for DVFS Voltage Drop Compensation Explorations. In *IEEE International Symposium on Performance Analysis of Systems and Software*, 2011.

[GH96] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. In *IEEE Journal of Solid-State Circuits*, volume 31, pages 1277–1284. IEEE Solid-State Circuits Society, 1996.

[Gra10] Mentor Graphics. ModelSim SE 6.6. `http://model.com/`, 2010.

[GRE+01] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *IEEE International Workshop on Workload Characterization*, pages 3–14, 2001.

[HKLS] J. Haid, W. Kargl, T. Leutgeb, and D. Scheiblhofer. Power Management for RF-Powered vs. Battery-Powered Devices.

[HM07] S. Herbert and D. Marculescu. Analysis of Dynamic Voltage/Frequency Scaling in Chip-Multiprocessors. In *Proceedings of the 2007 international symposium on Low power electronics and design*, pages 38–43, 2007.

[HNB08] M. Holtz, S. Narasimhan, and S. Bhunia. On-Die CMOS Voltage Droop Detection and Dynamic Compensation. In *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, pages 35–40, 2008.

[HTHR94] H. Hahn, D. Timmermann, B.J. Hosticka, and B. Rix. A Unified and Division-Free CORDIC Argument Reduction Method with Unlimited Convergence Domain Including Inverse Hyperbolic Functions. In *IEEE Transactions on Computers*, volume 43, pages 1339–1344, 1994.

[Hu92] Y.H. Hu. CORDIC-Based VLSI Architectures for Digital Signal Processing. In *IEEE Signal Processing Magazine*, volume 9, Issue:3, pages 16–35, 1992.

[IBC+06] 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 *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 347–358, 2006.

[IEE01] IEEE. *IEEE Std 1149.1-2001, IEEE Standard Test Access Port and Boundary-Scan Architecture*, 2001.

[Int09] Intel09. Realizing Data Center Savings with an Accelerated Server Refresh Strategy. 2009.

[Int10] Intel Corporation. *Intel Core i7-900 Desktop Processor Extreme Edition Series and Intel Core i7-900 Desktop Processor Series on 32-nm Process Datasheet*, 2010.

[Int11] Intel11. Intel Xeon Processor-based Server Refresh Savings Estimator. `http://www.intelsalestraining.com/xeonestimator/2B904B09/index.htm`, 2011.

[ITR11] ITRS. International Technology Roadmap for Semiconductors. `http://www.itrs.net`, 2011.

[JPC11] JPCAP. JPCAP Library. `http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html/`, 2011.

[KGWB08] W. Kim, M.S. Gupta, G. Wei, and D. Brooks. System Level Analysis of Fast, Per-Core DVFS using On-Chip Switching Regulators. In *IEEE 14th International Symposium on High Performance Computer Architecture*, pages 123–134, 2008.

[KM08] S. Kaxiras and M. Martonosi. *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool Publisher, 2008.

[Koo07] Jonathan G. Koomey. Estimating Total Power Consumption by Servers in the U.S. and the World. 2007.

[Kre93] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, Inc., 7th edition, 1993.

[Lac10] M. Lackner. Design and Implementation of a Multi-Core Power and Performance Emulation Platform. Master's thesis, Graz University of Technology, 2010.

[Mat10] MathWorks. MATLAB. `http://www.mathworks.com/`, 2010.

[MHK+99] T. Meincke, A. Hemanil, S. Kumar, P. Ellerveel, J. Oberg, T. Olsson, P. Nilsson, D. Lindqvist, and H. Tenhunen. Globally Asynchronous Locally Synchronous Architecture for Large High-Performance ASICS. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, pages 512–515, 1999.

[NIA03] T. Nakura, M. Ikeda, and K. Asada. Theoretical Study of Stubs for Power Line Noise Reduction. In *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference*, pages 715–718, 2003.

[NIA04] T. Nakura, M. Ikeda, and K. Asada. Preliminary Experiments for Power Supply Noise Reduction using Stubs. In *Proceedings of 2004 IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, pages 286–289, 2004.

[Pok07] K. Pokhrel. Physical and Silicon Measures of Low Power Clock Gating Success: An Apple to Apple Case Study. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*. SNUG, 2007.

[RGH+09] V.J. Reddi, M.S. Gupta, G. Holloway, G. Wei, M.D. Smith, and D. Brooks. Voltage Emergency Prediction Using Signatures to Reduce Operating Margins. In *IEEE 15th International Symposium on High Performance Computer Architecture*, pages 18–29, 2009.

[SK94] K. Scott and K. Keutzer. Improving Cell Libraries for Synthesis. In *Custom Integrated Circuits Conference, 1994., Proceedings of the IEEE 1994*, pages 128–135, 1994.

[SMB+02] G. Semeraro, G. Magklis, R. Balasubramonian, D.H. Albonesi, S. Dwarkadas, and M.L. Scott. Energy-Efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling. In *Eighth International Symposium on High-Performance Computer Architecture, Proceedings*, pages 29–40, 2002.

[SP93] R. Senthinathan and J.L. Prince. Application Specific CMOS Output Driver Circuit Design Techniques to Reduce Simultaneous Switching Noise. In *IEEE Journal of Solid-State Circuits*, volume 28, pages 1383–1388, 1993.

[SSN02] H. Su, S.S. Sapatnekar, and S.R. Nassif. An Algorithm for Optimal Decoupling Capacitor Sizing and Placement for Standard Cell Layouts. In *Proceedings of the 2002 international symposium on Physical design*, 2002.

[TDMG97] V. Tiwari, R. Donnelly, S. Malik, and R. Gonaalea. Dynamic Power Management for Microprocessors: A Case Study. In *Proceedings Tenth International Conference on VLSI Design*, pages 185–192, 1997.

[Tec10] Linear Technology. LTSpice. `http://www.linear.com/`, 2010.

[TM05] E. Talpes and D. Marculescu. Toward a Multiple Clock/Voltage Island Design Style for Power-Aware Processors. In *IEEE Transactions on Very Large Scale Integration Systems*, volume 13, pages 591–603, 2005.

[Vol59] J. E. Volder. The CORDIC Trigonometric Computing Technique. In *IRE Transactions on Electronic Computers*, volume EC-8, Issue:3, pages 330–334, 1959.

[WGSW08] M. Wendt, C. Grumer, C. Steger, and R. Weiss. System Level Power Profile Analysis and Optimization for Smart Cards and Mobile Devices. In *SAC '08 Proceedings of the 2008 ACM symposium on Applied computing*, pages 1884–1888, 2008.

[Xil04] Xilinx. *CORDIC v3.0, Product Specification*, 2004.

[ZDBT10] J. Zhao, B. Datta, W. Burleson, and R. Tessier. Thermal-aware Voltage Droop Compensation for Multi-core Architectures. In *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, 2010.

[ZSSS04] V. Zaccaria, M. Sami, D. Sciuto, and C. Silvano. *Power Estimation and Optimization Methodologies for VLIW-Based Embedded Systems*. Kluwer Academic Publishers, 2004.