Linköping Studies in Science and Technology

Thesis No. 1459

System-Level Techniques for Temperature-Aware Energy Optimization

by

Min Bao

Linköping University
INSTITUTE OF TECHNOLOGY

Submitted to Linköping Institute of Technology at Linköping University in partial fulfilment of the requirements for the degree of Licentiate of Engineering

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2010

System-Level Techniques for Temperature-Aware Energy Optimization

by

Min Bao

December 2010
ISBN 978-91-7393-264-6
Linköping Studies in Science and Technology
Thesis No. 1459
ISSN 0280-7971
LiU-Tek-Lic-2010:30

ABSTRACT

Energy consumption has become one of the main design constraints in today's integrated circuits. Techniques for energy optimization, from circuit-level up to system-level, have been intensively researched.

The advent of large-scale integration with deep sub-micron technologies has led to both high power densities and high chip working temperatures. At the same time, leakage power is becoming the dominant power consumption source of circuits, due to continuously lowered threshold voltages, as technology scales. In this context, temperature is an important parameter. One aspect, of particular interest for this thesis, is the strong inter-dependency between leakage and temperature. Apart from leakage power, temperature also has an important impact on circuit delay and, implicitly, on the frequency, mainly through its influence on carrier mobility and threshold voltage. For power-aware design techniques, temperature has become a major factor to be considered. In this thesis, we address the issue of system-level energy optimization for real-time embedded systems taking temperature aspects into consideration.

We have investigated two problems in this thesis: (1) Energy optimization via temperature-aware dynamic voltage/frequency scaling (DVFS). (2) Energy optimization through temperature-aware idle time (or slack) distribution (ITD). For the above two problems, we have proposed off-line techniques where only static slack is considered. To further improve energy efficiency, we have also proposed on-line techniques, which make use of both static and dynamic slack. Experimental results have demonstrated that considerable improvement of the energy efficiency can be achieved by applying our temperature-aware optimization techniques. Another contribution of this thesis is an analytical temperature analysis approach which is both accurate and sufficiently fast to be used inside an energy optimization loop.

This work has been supported by the Swedish Foundation for Strategic Research (SSF) under the Electronic Systems and Photonics Programme.

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

System-Level Techniques for Temperature-Aware Energy Optimization

Min Bao

Linköping University
INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science
Linköpings Universitet
SE-581 83 Linköping, Sweden

Linköping 2010

ISBN 978-91-7393-264-6, ISSN 0280-7971
Printed by LiU-Tryck, 2010

Electronic version available at:
`http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-60855`

Acknowledgements

During the time that I am working on this thesis, I have learned a lot about how to do and how to present research. There are many people who have, along the way, contributed to my progress. I would like to express my gratitude to them all.

First of all, I would like to thank my supervisor, Prof. Zebo Peng, for offering me the opportunity to pursue my postgraduate study here. He is extremely supportive and has given me countless valuable advice and help since the first day I came here.

Secondly, I would like to thank my second supervisor Prof. Petru Eles. Discussing my research with him is very enjoyable, and I can always get insightful and inspiring feedbacks. I deeply appreciate his patience and dedication in teaching and improving my technical writing and presentation skills.

Special thanks go to Dr. Alexandru Andrei, my previous colleague of Embedded Systems Laboratory. He was my mentor the first year I came here and has given me many valuable guidance both in research and life.

I would like to extend my thanks to all former and present members of Embedded Systems Laboratory. Because of them, the working atmosphere is so friendly and fun that I enjoy studying here every day.

Big thanks go to Eva Pelayo Danils, Inger Emanuelsson, Anne Moe, and Gunilla Mellheden who have been invaluable in their efforts to simplify all the administrative details.

I would like to thank all my friends who have made my life here interesting and memorable.

I am deeply grateful to my mother who always has faith in me. I could not finish this thesis without tremendous encouragement and unconditional support from her. This thesis is dedicated to her.

Min Bao
Linköping, Dec. 2010

Abstract

Energy consumption has become one of the main design constraints in today's integrated circuits. Techniques for energy optimization, from circuit-level up to system-level, have been intensively researched.

The advent of large-scale integration with deep sub-micron technologies has led to both high power densities and high chip working temperatures. At the same time, leakage power is becoming the dominant power consumption source of circuits, due to continuously lowered threshold voltages, as technology scales. In this context, temperature is an important parameter. One aspect, of particular interest for this thesis, is the strong inter-dependency between leakage and temperature. Apart from leakage power, temperature also has an important impact on circuit delay and, implicitly, on the frequency, mainly through its influence on carrier mobility and threshold voltage. For power-aware design techniques, temperature has become a major factor to be considered. In this thesis, we address the issue of system-level energy optimization for real-time embedded systems taking temperature aspects into consideration.

We have investigated two problems in this thesis: (1) Energy optimization via temperature-aware dynamic voltage/frequency scaling (DVFS). (2) Energy optimization through temperature-aware idle time (or slack) distribution (ITD). For the above two problems, we have proposed off-line techniques where only static slack is considered. To further improve energy efficiency, we have also proposed on-line techniques, which make use of both static and dynamic slack. Experimental results have demonstrated that considerable improvement of the energy efficiency can be achieved by applying our temperature-aware optimization techniques. Another contribution of this thesis is an analytical temperature analysis approach which is both accurate and sufficiently fast to be used inside an energy optimization loop.

Contents

1 Introduction **1**

1.1 Embedded Systems 1

1.2 Energy Issues 1

1.3 Dynamic Voltage/Frequency Scaling (DVFS) 2

1.4 Temperature Issues 2

1.5 Temperature Considerations in DVFS 3

1.5.1 Leakage/Temperature Dependency 3

1.5.2 Frequency/Temperature Dependency 3

1.6 Idle Time Distribution (ITD) 4

1.7 Related Work 4

1.7.1 Temperature Dependent Leakage Analysis 4

1.7.2 Architecture-Level Thermal Modeling 5

1.7.3 Thermal Sensing and Tracking 6

1.7.4 Temperature-Aware System-Level Design 6

1.8 Contributions 8

1.9 Thesis Organization 9

2 Preliminaries **11**

2.1 Power and Delay Models 11

2.2 Application Model 12

2.3 Architecture Model 12

2.4 Dynamic Voltage/Frequency Scaling 13

3 Temperature-Aware Dynamic Voltage/Frequency Scaling **15**

3.1 Motivational Example 15

3.1.1 Leakage/Temperature Dependency 15

3.1.2 Frequency/Temperature Dependency 16

3.1.3 On-line DVFS vs. Off-line DVFS 17

3.2 Temperature Analysis 18
3.2.1 Static Temperature Analysis 18
3.2.2 Dynamic Temperature Analysis 19
3.3 Static Temperature-Aware DVFS (SDVFS) 21
3.3.1 SDVFS with Leakage/Temperature Dependency (T-SDVFS) 21
3.3.2 SDVFS with Both Leakage/Temperature and Frequency/Temperature Dependencies (SDVFS-LF) 22
3.4 Dynamic Temperature-Aware DVFS (DDVFS) 23
3.4.1 Off-line and On-line Phases 23
3.4.2 LUT Generation 24
3.4.3 Temperature Bounds and Granularity 26
3.4.4 LUT Granularity Along the Time Dimension 27
3.4.5 Accounting for Analysis Accuracy and Ambient Temperature 28
3.5 Experimental Results 29
3.5.1 Static DVFS Approach 29
3.5.2 Dynamic DVFS Approach 32

4 Temperature-Aware Idle Time Distribution 37
4.1 Motivational Example 37
4.1.1 Static Idle Time Distribution 37
4.1.2 Dynamic Idle Time Distribution 39
4.2 Problem Formulation 41
4.2.1 ITD with Only Static Slack: SITD 42
4.2.2 ITD with Both Static and Dynamic Slack: DITD 44
4.3 Temperature Analysis 45
4.3.1 Temperature Model 45
4.3.2 SSDTC Estimation 47
4.3.3 TTC Estimation 48
4.4 ITD with Only Static Slack (SITD) 49
4.4.1 SITD without Overhead (SITDNOH) 49
4.4.2 SITD with Overhead (SITDOH) 52
4.5 ITD with Dynamic and Static Slack (DITD) 54
4.5.1 On-line Phase 55
4.5.2 Off-line Phase 56
4.5.3 DITDOH Algorithm 58
4.5.4 Time Bounds and Granularity 60
4.5.5 Temperature Bounds and Granularity 61
4.6 Experimental Results 63
4.6.1 Evaluation of The Thermal Model 63
4.6.2 Evaluation of ITD Heuristics 64

5 Conclusions **71**

References **73**

List of Figures

3.1 Leakage/Temperature Dependency Influence in DVFS 16
3.2 Motivational Example 17
3.3 Static Thermal Analysis Considering Leakage/Temperature Dependency 19
3.4 Dynamic Thermal Analysis Considering Leakage/Temperature Dependency 20
3.5 SDVFS with Leakage/Temperature Dependency 21
3.6 Typical Temperature Convergence Curve 22
3.7 On-Line Phase 24
3.8 LUT Generation 25
3.9 Energy Improvement with T-SDVFS Approach 30
3.10 Energy Improvement for The MPEG2 Decoder 31
3.11 Dynamic vs. Static Approach 33
3.12 Computation Time: Off-line Phase 34
3.13 Impact of Temperature Line Number 35
3.14 Impact of The Ambient Temperature 36

4.1 Motivational Example: Static Idle Time Distribution 38
4.2 Motivational Example: Idle Time Distribution 40
4.3 Thermal Circuit 46
4.4 Temperature Analysis 47
4.5 SITDNOH Heuristic 52
4.6 SITDOH Heuristic 54
4.7 DITD On-line Phase 55
4.8 DITD Off-line Phase 57
4.9 SSDTC Estimation with Our Approach vs. Hotspot 64
4.10 Leakage Energy Reduction with Low Switching Overheads 66
4.11 Leakage Energy Reduction with High Switching Overheads . . . 67

4.12 Leakage Energy Reduction with No Switching Overheads 68
4.13 Leakage Energy Reduction with Different Standard Deviations . . 68
4.14 Computation Time 69

List of Tables

3.1 DVFS without Frequency/Temperature Dependency 17
3.2 DVFS with Frequency/Temperature Dependency 17
3.3 Dynamic DVFS 18
3.4 Energy Reduction from Using SDVFS-LF Comparing with T-SDVFS 32
3.5 Energy Improvement by DDVFS with Frequency/Temperature Dependency 32
3.6 Energy Improvement Degradation by Simulation Accuracy 36

4.1 Motivational Example: Application Parameters 37
4.2 Static ITD: Leakage Energy Comparison 38
4.3 Motivational Example: An Activation Scenario 39
4.4 Dynamic ITD: Leakage Energy Comparison 40

List of Abbreviations

BNC	Best Number of Cycles
DDVFS	Dynamic Voltage/Frequency Scaling with Both Dynamic and Static Slack
DITD	Idle Time Distribution with Both Dynamic and Static Slack
DITDOH	DITD with Overheads Consideration
DVFS	Dynamic Voltage/Frequency Scaling
EFT	Earliest Finishing Time
ENC	Expected Number of Cycles
EST	Earliest Starting Time
ITD	Idle Time Distribution
ITDNOH	Idle Time Distribution with No Overheads Consideration
ITDOH	Idle Time Distribution with Overheads Consideration
LFT	Latest Finishing Time
LST	Latest Starting Time
LUT	Look-up Table
NT-DVFS	None Temperature-Aware Dynamic Voltage/Frequency Scaling
OS	Operating Systems

SDVFS	Dynamic Voltage/Frequency Scaling with Static Slack
SDVFS-LF	SDVFS with Both Leakage/Temperature and Frequency/Temperature Dependencies Consideration
SFA	Straightforward Approach
SITD	Idle Time Distribution with Static Slack
SITDNOH	SITD with No Overheads Consideration
SITDOH	SITD with Overheads Consideration
SSDTC	Steady State Dynamic Temperature Curve
T-DVFS	Temperature-Aware Dynamic Voltage/Frequency Scaling
TTC	Transient Temperature Curve
WNC	Worst Number of Cycles

Chapter 1

Introduction

1.1 Embedded Systems

Embedded systems are information processing systems that are embedded into a larger product and usually are not visible to users [1]. Embedded systems have a wide range of application areas and are one of the most rapidly growing segments of the computer industry [2]. New products appear with an explosive speed and, nowadays, embedded systems are used everywhere, e.g. in automotive systems, medical equipments, consumer electronics and tele-communication devices.

Unlike general purpose computer systems, such as personal computers (PC), embedded systems are designed for dedicated functionalities. A common characteristic of embedded systems is that real-time response is usually required. This means that delivering results within certain time constraints is important for a correct functionality of the system.

The design of embedded systems is challenging since the implementation has not only to produce correct functionalities but also to meet diverse competing constraints, e.g. physical size, cost, performance, reliability, flexibility, and testability [3]. The constraints can be addressed in different levels of abstraction: from circuits level up to system level. In this thesis, we focus on several aspects related to the *system-level design* of embedded systems [4].

1.2 Energy Issues

Energy consumption is one of the main design constraints in today's integrated circuits. For battery-operated devices, e.g. mobile consumer electronics, the available energy is of a fixed amount; the rate of power consumption determines the lifetime of the battery or the time between two recharges of the battery. The ever increasing computation complexity, which doubles every two years [5], results in elevated power and energy consumptions. However, the battery technology only improves around 3–7% per year [5], lagging far behind the increase of the required energy consumption.

Energy optimization techniques, from circuit level up to system level, are needed in order to close the gap between energy consumption and battery capacity. Extensive research has been performed on energy optimization for embedded systems. In this thesis, we focus on the system-level energy optimization techniques.

1.3 Dynamic Voltage/Frequency Scaling (DVFS)

At system level, dynamic voltage/frequency scaling (DVFS) is one of the preferred approaches for reducing the overall energy consumption [6], [7]. This technique exploits the available slack time in real-time applications to achieve energy efficiency by reducing the supply voltage and frequency such that the execution of tasks is stretched within their deadline.

There are two types of slacks.

- Static slack, which is due to the fact that, when executing at the highest (nominal) voltage level, tasks finish before their deadlines even when executing their worst case number of cycles (WNC).
- Dynamic slack, due to the fact that most of the time tasks execute less than their WNC.

Off-line DVFS techniques, such as those in [8] and [9], can only exploit static slack, while on-line approaches, e.g. [10], [11], [12] and [13] are able to further reduce energy consumption by exploiting the dynamic slack due to the variation of the workload generated by the tasks.

1.4 Temperature Issues

Junction temperature is one of the most important CMOS parameters [14]. Temperature has a strong impact on system reliability. Excessive high working temperature can lead to permanent faults due to electro-migration and other process failure, while frequent temperature variations can result in transient faults, e.g. transient voltage fluctuations [15].

Of most interest, in this thesis, is the strong influence of temperature on leakage current and circuit delay. The impact of temperature on circuit delay and, implicitly, on frequency, is mainly through its influence on carrier mobility and threshold voltage [16]. With high working temperature the carrier mobility decreases, which degrades the circuits' performance. Leakage current, which consists of various components among which the sub-threshold leakage current dominates, is strongly dependent on temperature due to the temperature's strong impact on sub-threshold leakage. Sub-threshold leakage is caused by the weak inversion conduction of transistors [17] and increases rapidly with temperature.

Technology scaling leads to high power densities in current circuits, which have resulted in a high working temperature. On the other hand, technology scaling continuously lowers threshold voltages in order to maintain the improvement of performance, leading to an exponential increase in sub-threshold current [17]. As a result, leakage energy is becoming the dominant energy consumption source of circuits [18]. Due to the strong inter-dependency between leakage current and temperature [19], growing temperature can lead to an increase in leakage current and, consequently, energy, which, again, produces higher temperatures. For power-aware techniques, temperature has therefore become an important parameter to be taken into consideration.

1.5 Temperature Considerations in DVFS

1.5.1 Leakage/Temperature Dependency

Traditionally, the dependency of leakage on temperature is ignored in DVFS, due to the fact that leakage energy used to represent only a small percentage of the total energy consumption. To perform voltage selection for energy optimization, at design time an empirical assumed working temperature of the chip is used for leakage energy estimation. For example, the actual working temperature of the chip is assumed to be 70°C. However, as pointed out in the previous section that leakage is becoming the dominant power consumption as technology scales, ignoring the leakage/temperature dependency in DVFS can lead to very inaccurate leakage energy estimation and, hence, sub-optimal energy consumption.

1.5.2 Frequency/Temperature Dependency

As mentioned in Section 1.4 that temperature has also an important impact on the frequency of circuits. At the same time, frequency also depends on the supply voltage. In order to provide performance, the frequency is usually set to the maximum value allowed by the current supply voltage. However, traditionally, when calculating this maximum allowed frequency for a given supply voltage V, it is implicitly assumed that this is the frequency f corresponding to the maximum temperature T_{max} at which the chip is allowed to run. While this is a safe assumption, it is far from efficient. If we are aware that the chip is running at a temperature $T < T_{max}$, the frequency could be fixed at $f' > f$ and, thus, performance is increased for the same energy consumption. Or, maybe more important, the same frequency f could be achieved with a supply voltage $V' < V$ and, thus, further energy is saved.

With the strong impact of temperature on both leakage and frequency, temperature is an important aspect to be considered for DVFS. In this thesis we investigate the issue of DVFS techniques taking the temperature aspect into consideration.

1.6 Idle Time Distribution (ITD)

As mentioned in Section 1.3, DVFS reduces energy consumption by exploiting slack. However, very often, not all available slack should or can be exploited and certain amount of slack may still exist after DVFS. An obvious situation is when the lowest supply voltage is such that, even if selected, a certain slack interval is left. Another reason is the existence of a critical voltage [20]. To achieve the optimal energy efficiency, DVFS would not execute a task at a voltage lower than the critical one, since, otherwise, the additional static energy consumed due to the longer execution time is larger than the energy saving due to the lowered voltage. During the available slack interval, the processor remains idle and can be switched to a low power state. Due to the strong inter-dependence between leakage power and temperature, different distributions of idle time will lead to different temperature distributions and, consequentially, energy consumption. In this thesis, we take the temperature aspect into consideration and address the issue of optimizing energy consumption through efficient distribution of both static and dynamic slack.

1.7 Related Work

1.7.1 Temperature Dependent Leakage Analysis

As leakage current is strongly dependent on temperature [19], temperature-aware leakage models are needed to correctly estimate leakage power consumption. Liao et al. [19] proposed a temperature-aware leakage model which describes the exponential dependency of leakage current on temperature. In [21], the authors proposed to piece-wise linear approximating of the exponential leakage model with less than 1% error. A leakage model which describes leakage current as a quadratic function of temperature was proposed in [22]. Huang et al. [23] performed a comprehensive study of different temperature-aware leakage models, where exponential, quadratic, piece-wise linear, and linear leakage models were compared, and the trade-off between the complexity and the accuracy of the models was discussed.

1.7.2 Architecture-Level Thermal Modeling

Temperature-aware system-level design methods rely on the availability of temperature modeling and analysis approaches. Most temperature modeling tools are based on the duality between heat transfer and electrical phenomena [24]. There are two types of thermal analysis: (1) static temperature analysis and (2) dynamic temperature analysis. With static temperature analysis a temperature value, at which the circuit is supposed to function in steady state, is computed. With dynamic temperature analysis a temperature profile, which describes the temperature behaviour of the circuit as a function of time, is calculated. There has been research on architecture-level thermal analysis, e.g., Hotspot [25] and ISAC [26]. The basic idea of Hotspot is to build an equivalent circuit of thermal resistances and capacitances capturing both the architecture blocks and the elements of the thermal package. HotSpot can be used both for static analysis and dynamic analysis. ISAC, proposed in [26], is similar to Hotspot, and it speeds up the thermal analysis through dynamic adaptation of the resolution.

The thermal analysis used in Chapter 3 is based on Hotspot. For our purposes, the architecture is modeled at core level. Thus, from the architecture point of view, the actual blocks whose temperature is analyzed are the processors on which the tasks are executed. When provided with the physical/thermal parameters (size and placement of blocks, thermal capacitances and resistances, parameters of packaging elements) and the power profile capturing the power dissipation of the core, HotSpot produces the steady state temperature or the temperature profile of the processor. However, the temperature analysis does not support the case in which power dissipation is dependent on the temperature, which, obviously, is the situation with leakage. In Chapter 3, we propose modifications of Hotspot to overcome the above problem for static and dynamic temperature analysis, respectively.

The computation complexity of the architecture-level temperature analysis approaches like the two mentioned above is large. There has been research on establishing fast system-level temperature analysis techniques which are sufficiently efficient to be used inside an optimization loop of temperature-aware system-level design techniques, e.g. [27], [28], [29], and [30]. They also build on the duality between heat transfer and electrical phenomena and are based on very restrictive assumptions in order to simplify the model. In [27] the authors assumed that (1) no cooling layer is present, (2) there is no interdependency between leakage current and temperature, and (3) the whole application executes at a constant voltage. The models in [28] and [29] consider variable voltage levels but maintain the first two limitations above. The most general analytical model is proposed in [30] which considers cooling layers as well as the dependency between leakage and temperature. However, this approach is limited to the case of a unique voltage level throughout the application. In Chapter 4 we will introduce a fast and accurate temperature analysis technique which eliminates all three limitations mentioned above and can be efficiently used inside the optimization loop of temperature-aware system-level design techniques.

1.7.3 Thermal Sensing and Tracking

Many temperature-aware system-level design approaches are proposed in which decisions are taken on-line, based on the actual chip temperature information. In such cases, thermal sensors [31] are used together with techniques for collecting and analyzing their values with adequate accuracy. For example, the techniques for dynamic OS-level workload scheduling aiming at avoiding thermal hot spots and large temperature variations [32] are based on run time temperature sensor readings. In Chapter 3 and Chapter 4, in addition to off-line approaches, we also propose on-line DVFS and ITD approaches which rely on temperature sensing and tracking techniques.

Several approaches have been proposed in literature to improve the accuracy of temperature measurement and estimation. For example, in [33] and [34], the authors proposed techniques to determine the optimal locations and allocations of thermal sensors with the goal of accurate hot spot detection as well as full chip thermal characterization. In [35], [36], and [37], the authors addressed the issue of how to process/analyze readings from sparse and noisy thermal sensors to accurately estimate temperatures where various estimation schemes such as spectral methods and Kalman filters are utilized.

1.7.4 Temperature-Aware System-Level Design

Several approaches to system-level temperature-aware design have been discussed in literature.

Temperature management is utilized to control the temperature behavior of processors for improving system reliability [15]. In [38], the authors proposed a technique for temperature management by scaling the processor speed and, in [39], the authors addressed the issue of scheduling and mapping of a set of tasks with real-time constraints on multi-processors for peak temperature minimization. Techniques for task sequencing combined with DVFS to reduce the peak working temperature of the processor were proposed in [28]. Several approaches aiming at reducing temperature variations or temperature gradients across the chip, e.g. in [40], were proposed.

A considerable amount of work has been published on performance optimization under thermal and real-time constraints. For example, Zhang et al. [41] proposed voltage assignment techniques to optimize the performance of a set of periodic tasks working on a DVFS enabled processor under thermal constraints. In [42], the authors proposed approaches to optimize throughput by task sequencing under thermal constraints. An on-line speed adaptation technique for homogeneous multi-processors with the target of maximizing the total throughput was proposed by Rao et al. in [43].

As discussed in Section 1.4, temperature is an important issue to be considered for power-aware system-level design. Since DVFS techniques are supposed to reduce energy consumption by adapting voltage levels, leakage/temperature and frequency/temperature dependencies are important aspects to be taken into consideration at voltage selection. However, very few of the proposed DVFS techniques have considered the leakage/temperature and frequency/temperature dependencies. The DVFS approach proposed by Liu in [21] is a static DVFS scheme aiming at reducing peak temperature. An on-line DVFS approach with consideration of both leakage/temperature and frequency/temperature dependencies was proposed in [44] where the throughput is maximized within the constraint of a peak working temperature. The authors in [45] proposed an on-line DVFS approach which is based on a design time optimization procedure performed considering various start time temperatures and workloads. At run-time, frequency settings are based on actual temperatures received from sensors. The approach, however, ignores the leakage/temperature dependency and assumes (as in off-line DVFS techniques) that the number of cycles executed by a given task is fixed and known at design time. In Chapter 3, we propose off-line and on-line DVFS techniques which take both leakage/temperature and frequency/temperature dependencies into considerations.

As mentioned in Section 1.6, in this thesis,we address, the issue of optimizing leakage energy consumption through distribution of idle time. The only work, to our knowledge, previously addressing this issue is [46] and [22]. In [46], the authors proposed an approach to distribute idle time for applications consisting of one single task executing at a constant given supply voltage. Thus, their approach cannot optimize the distribution of idle time among multiple tasks which also execute at different voltages. The same limitation also holds for [22], where a pattern based ITD for leakage energy optimization considering one single task was proposed. The pattern based approach generates uniform idle time distribution over the whole application and, thus, is not appropriate for ITD among multi-task applications where tasks have different amounts of energy consumption and execute at different voltage levels.

1.8 Contributions

In this thesis, we make the following main contributions:

1. We propose a temperature simulation method, based on Hotspot, which considers the leakage/temperature dependency.

2. We propose an off-line temperature-aware DVFS approach for energy optimization, which takes both leakage/temperature and frequency/temperature dependencies into consideration.

3. We propose, based on our off-line temperature-aware DVFS approach, an on-line temperature-aware DVFS technique which can exploit both static and dynamic slack. This approach is look up table (LUT) based and is composed of two phases: (1) During the off-line phase, look up tables (LUT) are generated for each task. (2) At runtime, voltage/frequency settings are decided by checking the corresponding task's LUT according to temperature sensor readings.

4. We propose a fast and accurate analytical temperature model which eliminates all the three limitations mentioned in Section 1.7.2, by considering the following aspects: a) the interdependence between leakage power consumption and temperature; b) multiple thermal cooling layers of the chip; c) non-smooth power consumption generated due to multiple discrete supply voltage levels of the processor. Our model can be efficiently used for both static and dynamic temperature analysis.

5. We propose an off-line ITD approach to optimize leakage energy consumption for a set of periodic tasks. It distributes static slack globally among tasks which are executed at different discrete voltage levels. This off-line ITD is based on an iterative heuristic using a convex optimization which can be solved in polynomial time.

6. We propose, based on the off-line ITD approach, an on-line ITD technique where both static and dynamic slack are distributed. This approach is look up table (LUT) based and is composed of two phases: (1) the off-line phase prepares a LUT for each task; (2) at runtime, when a task is finished, the idle time length following the finished task is decided by checking the task's LUT.

7. For systems with DVFS features, the proposed ITD approaches can be combined with DVFS techniques, in which case additional energy reduction can be achieved.

Part of the content of this thesis has been presented in the following papers:

- M. Bao, A. Andrei, P. Eles, and Z. Peng, ”On-Line Temperature-Aware Idle Time Distribution for Leakage Energy Optimization”, the 6th International Symposium on Electronic Design, Test and Applications (DELTA11), Jan.15–17, 2011 [47].

- M. Bao, A. Andrei, P. Eles, and Z. Peng, ”Temperature-Aware Idle Time Distribution for Energy Optimization with Dynamic Voltage Scaling”, the 10th Swedish System-on-Chip Conference (SSOCC10), May 3–4, 2010 [48].

- M. Bao, A. Andrei, P. Eles, and Z. Peng, ”Temperature-Aware Idle Time Distribution for Energy Optimization with Dynamic Voltage Scaling”, Design Automation and Test in Europe (DATE 2010), Mar. 8–12, 2010 [49].

- M. Bao, A. Andrei, P. Eles, and Z. Peng, ”On-line Thermal Aware Dynamic Voltage Scaling for Energy Optimization with Frequency/Temperature Dependency Consideration”, Design Automation Conference (DAC 2009), Jul. 26–31, 2009 [50].

- M. Bao, A. Andrei, P. Eles, and Z. Peng, ”Temperature-Aware Voltage Selection for Energy Optimization”, The 9th Swedish System-on-Chip Conference (SSOCC09), May 4–5, 2009 [51].

- M. Bao, A. Andrei, P. Eles, and Z. Peng, ”An Energy Efficient Technique for Temperature-Aware Voltage Selection”, Technical Reports in Computer and Information Science, ISSN 1654-7233; Linköping University Electronic Press, 2009 [52].

- M. Bao, A. Andrei, P. Eles, and Z. Peng, ”Temperature-Aware Task Mapping for Energy Optimization with Dynamic Voltage Scaling”, IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS’08), Apr. 16–18, 2008 [53].

- M. Bao, A. Andrei, P. Eles, and Z. Peng, ”Temperature-Aware Voltage Selection for Energy Optimization”, Design Automation and Test in Europe (DATE 2008), Mar. 10–14, 2008 [54].

1.9 Thesis Organization

The rest of this thesis is organized as follows. Preliminaries are presented in Chapter 2. In Chapter 3 we present the temperature-aware DVFS approaches. We first present the modified thermal model used in our DVFS techniques. Then we describe the off-line and on-line DVFS approach. In Chapter 4 we present the temperature-aware ITD methods. We start with introducing our analytical system-level thermal model. Based on the proposed thermal model, we then discuss our off-line and on-line ITD approaches. Finally, conclusions are discussed in Chapter 5.

Chapter 2

Preliminaries

2.1 Power and Delay Models

Digital CMOS circuits have two major sources of power dissipation: (1) dynamic power P^{dyn}, which is dissipated whenever computations are carried out (switching of logic gates) and (2) leakage power P^{leak}, which is consumed whenever the circuit is powered, even if no computation is performed. For dynamic power we use the following equation [55]:

$$P^{dyn} \ = \ Ceff \cdot f \cdot V^2 \tag{2.1}$$

where $Ceff$, V, and f denote the effective charged capacitance, supply voltage, and frequency, respectively.

The leakage power is expressed as follows [19]:

$$P^{leak} = I_{sr} \cdot T^2 \cdot e^{\frac{\beta \cdot V + \gamma}{T}} \cdot V \tag{2.2}$$

where I_{sr} is the leakage current at a reference temperature, T is the current temperature, and β and γ are technology dependent coefficients. In Chapter 4, we will use a piece-wise linear approximation of this model, as proposed, for example, in [21]. According to it, the working temperature range $[T_a, T_{max}]$, where T_a and T_{max} are the ambient and the maximal working temperature of the chip, is divided into several sub-ranges. The leakage power inside each sub-range $[T_i, T_{i+1}]$ is modeled by a linear function:

$$P_i = M_i \cdot T + B_i \tag{2.3}$$

where M_i and B_i are constants characteristic to each interval.

The maximum frequency of the processor at a given reference temperature T_{ref} is calculated as follows [55]:

$$f = \frac{1}{d} = \frac{((1 + K_1) \cdot V - v_{th1})^{\alpha}}{K_6 \cdot Ld \cdot V} \tag{2.4}$$

where Ld is the logic depth. K_1, K_6, and v_{th1} are technology dependent coefficients. α reflects the velocity saturation imposed by the used technology (commonly, 1.4 $< \alpha <$ 2). The scaling of frequency with temperature is given by Eq. (2.5) [19]:

$$f \propto \frac{(V - (v_{th1} + k \cdot (T - T_{ref})))^{\xi}}{V \cdot T^{\mu}} \tag{2.5}$$

T_{ref} and T are the reference temperature and current temperature, while k, ξ, and μ are empirical technology dependent constants.

2.2 Application Model

The application is captured as a task graph $G(\Pi, \Gamma)$. A node $\tau_i \in \Pi$ represents a computational task τ_i, while an edge $e \in \Gamma$ indicates the data dependency between two tasks. Each task τ_i is characterized by the following six-tuple:

$$< WNC_i, BNC_i, ENC_i, dl_i, Ceff_i >$$

where WNC_i, BNC_i, and ENC_i are task τ_i's worse case, best case, and expected number of clock cycles to be executed. The expected number of clock cycles ENC_i is the arithmetic mean value of the probability density function of the actual executed cycles ANC_i, i.e. $ENC_i = \sum_{j=BNC_i}^{WNC_i}(j \cdot p^i(j))$, where $p^i(j)$ is the probability that a number j of clock cycles are executed by task τ_i. We assume that the probability density functions of the execution cycles of different tasks are independent. Further, dl_i and $Ceff_i$ represent the deadline and the effective switched capacitance.

2.3 Architecture Model

The application is mapped and scheduled on a processor which has two power states: active and idle. In the active state the processor can operate at several discrete supply voltage levels. When the processor does not execute any task, it can be put to the idle state, consuming a very small amount of leakage power. We assume this leakage power P_{idle} to be constant due to its small amount. Switching the processor between the idle and active state as well as between different voltage levels incurs both time and energy overheads. The processor has internal temperature sensors that can be accessed during execution.

2.4 Dynamic Voltage/Frequency Scaling

Our temperature-aware DVFS approach proposed in this thesis is based on the DVFS approach presented in [7]. Given an architecture and a mapped and scheduled application as described above, the DVFS algorithm in [7] calculates the appropriate supply voltages for each task, such that the total energy consumption is minimized. Another input to the algorithm is the dynamic power profile of the application, which is captured by the average switched capacitance of each task. This information will be used for calculating the dynamic energy consumed by the task at a certain supply voltage level, according to Eq. (2.1). Leakage energy, during the optimization process, is calculated based on Eq. (2.2). However, the dependence of leakage on temperature has been ignored in this voltage scaling algorithm [7]. To perform voltage selection, designers need to introduce an assumed temperature which is used at energy optimization. This, as discussed in Chapter 1.5.1, leads to suboptimal results.

Another limitation of this DVFS approach is, as mentioned at Chapter 1.5.2, that the dependency of the frequency on temperature is ignored. Thus, the produced solutions are excessively conservative. Finally, this DVFS approach is a static technique, assuming that tasks always execute their WNC and, thus, cannot exploit the dynamic slack.

In Chapter 3, we will take both leakage/temperature and frequency/temperature dependencies into consideration, and develop DVFS techniques considering both static and dynamic slack.

Chapter 3

Temperature-Aware Dynamic Voltage/Frequency Scaling

3.1 Motivational Example

3.1.1 Leakage/Temperature Dependency

Let us consider the following example. A periodic application which contains only one task τ is to be executed on a DVFS-enabled processor. The number of clock cycles to be executed in the worst case (WNC) is 2.3×10^6, and the average switched capacitance (in F) is 5×10^{-9}. The deadline of task τ is equal to its period, which is 0.035s. The processor has 8 discrete voltage levels from 0.5V to 1.2V, with a step of 0.1V.

For the above example, we need to assign a voltage level to execute task τ such that the total energy consumed is minimized. The execution time of task τ at each supply voltage level is shown by the curve marked with triangles in Fig. 3.1. The horizontal dashed line indicates the deadline. As can be seen from Fig. 3.1, in order for the deadline to be satisfied, we can choose voltage levels in the interval 0.6V–1.2V. The total energy consumption of task τ working at each voltage level is dependent on the temperature, as leakage is strongly dependent on temperature. We have computed the total energy consumption of task τ working at each voltage level considering two different working temperature values of the task τ. As shown in Fig. 3.1, the line marked with squares shows the total energy consumption of task τ executed at the temperature of 45°C for each supply voltage level, while the line marked with dots shows the total energy consumption when task τ is executed at the temperature of 90°C.

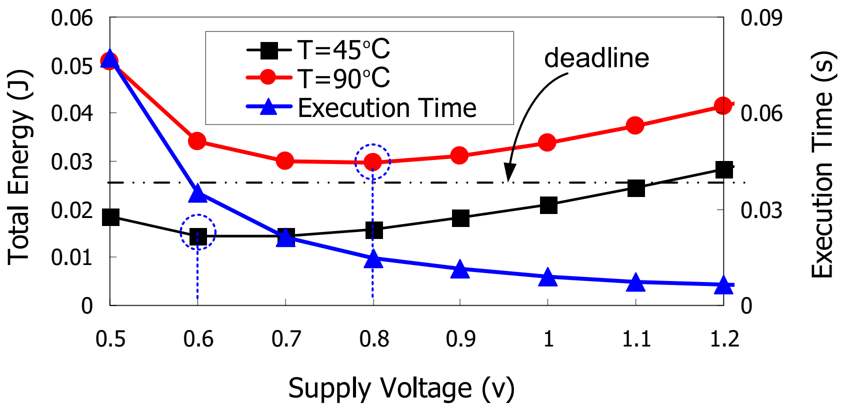


Figure 3.1: Leakage/Temperature Dependency Influence in DVFS

We can observe that the optimal supply voltage levels (marked with dashed-line circles) are different for the two working temperatures. If we blindly assume the task to be working at 45°C while in reality it is working at 90°C, we would choose to execute the task at 0.6V with an energy consumption of 0.034J. This will lead to an energy loss of 13% compared to the real optimal energy consumption (0.029J) achieved at 0.8V, at a working temperature of 90°C. Thus, to minimize the energy consumption with DVFS, temperature consideration and, implicitly, the consideration of the leakage/temperature dependency is of huge importance.

3.1.2 Frequency/Temperature Dependency

In this section, we will use an example to describe the importance of considering the frequency/temperature dependency for DVFS. Let us consider an application consisting of three tasks as shown in Fig.3.2. The number of clock cycles to be executed in the worst case (WNC) for τ_1, τ_2, and τ_3 is 2.85×10^6, 1.0×10^6, and 4.30×10^6, respectively, and their average switched capacitance (in F) is 1.0×10^{-9}, 0.9×10^{-10}, and 1.5×10^{-8}, respectively. The application has a global deadline of 0.0128s. We assume that the three tasks are executed on a processor which has 9 discrete voltage levels from 1.0V to 1.8V with a step of 0.1V. The chip size is $7\times7\text{mm}^2$ with a maximum allowable working temperature of T_{max} = 125°C.

For the above example, we perform energy minimization using a temperature-aware DVFS method which ignores the frequency/temperature dependency. When

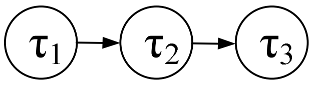


Figure 3.2: Motivational Example

calculating the maximum allowed frequency for a certain supply voltage, the maximum allowed working temperature for the chip, T_{max}= 125°C, is considered. In Table 3.1 we show the actual voltages and frequencies for each task, as calculated by the DVFS algorithm and the consumed energy. We also show the peak temperature for each task when executed with the calculated voltage and frequency, obtained with dynamic thermal analysis. As can be observed, this peak temperature is far below the T_{max} of the chip.

Table 3.1: DVFS without Frequency/Temperature Dependency

Task	Peak Temp(°C)	Voltage(V)	Freq(MHz)	Energy(J)	Total(J)
τ_1	79.9	1.7	638	0.059	
τ_2	78.1	1.7	638	0.020	
τ_3	81.2	1.7	638	0.264	0.343

From Eq. (2.4) and Eq. (2.5), it is obvious that, by taking into consideration the actual temperature at frequency calculation, there is a large margin for reducing the supply voltage without compromising on performance. We have performed a DVFS based energy optimization, similar to the one above, but with the difference that frequencies corresponding to the different voltage settings are calculated by taking into consideration the peak temperature at which the task actually runs. Table 3.2 shows the results. We can see that an energy reduction of 25% has been obtained.

Table 3.2: DVFS with Frequency/Temperature Dependency

Task	Peak Temp(°C)	Voltage(V)	Freq(MHz)	Energy(J)	Total(J)
τ_1	69.9	1.7	726	0.049	
τ_2	69.0	1.6	661	0.015	
τ_3	70.1	1.5	593	0.194	0.258

3.1.3 On-line DVFS vs. Off-line DVFS

The DVFS approach used in Section 3.1.2 is an off-line, static one which assumes that tasks always execute their WNC and, thus, can only exploit the static slack.

In reality, however, there are huge variations in the number of cycles executed by a task, from one activation to the other, which leads to a considerable amount of dynamic slack. Imagine an activation scenario for which each of the three tasks in Fig. 3.2 executes a number of cycles equal to 60% of their WNC. If we use the above off-line DVFS approach and run at the voltages and frequencies calculated as in Table 3.2, the total energy consumption would be 0.149J. However, much more can be done by also exploiting the dynamic slack. This implies that, at run-time, whenever a task terminates, the voltage level and the frequency for the next task are calculated by taking into consideration the current time and the current chip temperature. Table 3.3 shows the voltage and frequency levels determined in this way as well as the corresponding energy consumption. The total energy consumed is 0.133J, which means a reduction of 10% compared to the off-line DVFS approach.

Table 3.3: Dynamic DVFS

Task	Peak Temp(°C)	Voltage(V)	Freq(MHz)	Energy(J)	Total(J)
τ_1	52.5	1.5	620	0.018	
τ_2	50.4	1.2	417	0.004	
τ_3	51.4	1.5	582	0.111	0.133

The examples presented in this section demonstrate that (1) considering frequency/temperature as well as leakage/temperature dependency at DVFS can lead to substantial energy savings and (2) an on-line temperature-aware approach is needed in order to make use of the dynamic slack created due to variable number of clock cycles executed at different activations.

3.2 Temperature Analysis

Temperature analysis in our proposed DVFS technique is based on HotSpot [25]. As mentioned in Chapter 1.7.2, the temperature analysis by Hotpsot does not consider the dependency of leakage on temperature. In the following two sections, we describe our solutions to overcome the above problem for static and dynamic temperature analysis respectively.

3.2.1 Static Temperature Analysis

In the case of static temperature analysis, some solutions have been proposed in [56], [57] and [58]. A similar solution is used by us and is outlined in Fig. 3.3. As mentioned in Chapter 1.7.2, corresponding to an input power profile of the

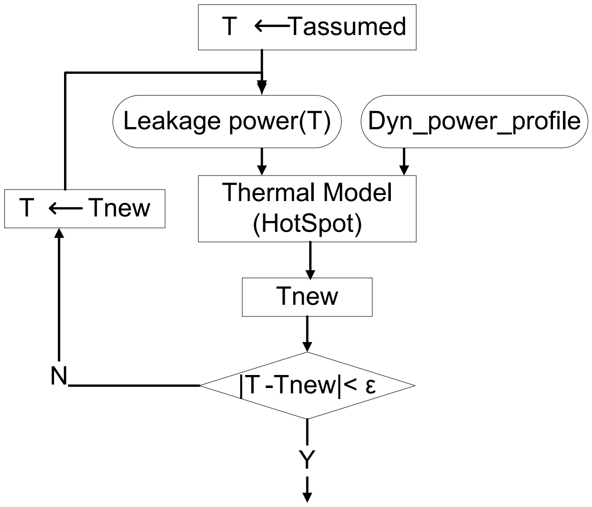


Figure 3.3: Static Thermal Analysis Considering Leakage/Temperature Dependency

processor, HotSpot will produce a steady state temperature at which the core is supposed to work. However, to input the leakage component of the power profile, the working temperature in steady state has to be known. In order to overcome this cyclic dependency, the process is started with an ”assumed” temperature and then continued iteratively until the produced temperature converges. At the obtained steady state temperature, the dissipated heat is in balance with the heat removal capacity of the package. However, it can happen that such a balance is not achieved, due to insufficient heat removal, and the temperature is increasing, potentially, to infinite. In such a case, the iterations in Fig. 3.3 will not converge. This phenomenon, called thermal runaway, is detected and indicates that the design is incorrect from the thermal point of view. Detecting thermal runaway is an important part of a thermal-aware design process.

3.2.2 Dynamic Temperature Analysis

Static analysis assumes that, eventually, the chip will function at one constant temperature. This, however, is usually not necessarily the case in reality. In the context of a variable power profile, the chip will not reach a constant steady state temperature but a steady state in which temperature is varying according to a certain pattern. In order to obtain the steady state temperature profile, we need to use dynamic thermal analysis. For dynamic analysis, HotSpot is calculating temperatures at successive time steps [25]. At each step a new temperature is calculated for each block by solving the equations describing the thermal model, based on a fourth-order Runge-Kutta method. The power consumption during the time interval between two steps is extracted from the power profile for the respective block. However, leakage power is a function of the temperature and, thus, cannot be delivered as an input to the analysis.

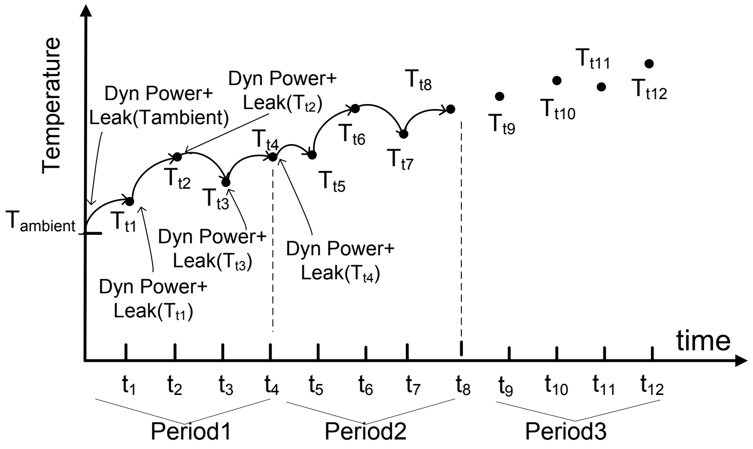


Figure 3.4: Dynamic Thermal Analysis Considering Leakage/Temperature Dependency

In order to solve the above problem we have extended the thermal analysis such that the power consumption during a time step is calculated as the sum of two components: (1) the dynamic power extracted from the input power profile and (2) the leakage power calculated at the temperature level of the previous step. The process is illustrated in Fig. 3.4. Temperature analysis is repeated for successive periods of the application. In order to detect convergence, temperature values at corresponding time steps of these successive periods are compared.

For both static and dynamic analysis, convergence is reached efficiently unless thermal runaway occurs. Since dynamic thermal analysis itself is much more time consuming than static analysis, obtaining a steady state temperature profile is much slower than calculating a constant steady state temperature.

3.3 Static Temperature-Aware DVFS (SDVFS)

3.3.1 SDVFS with Leakage/Temperature Dependency (T-SDVFS)

In Fig. 3.5 we show the overall flow of our static temperature-aware DVFS approach taking leakage/temperature dependency into consideration. Given is a scheduled and mapped task graph, and the average switched capacitance for each task. A so called

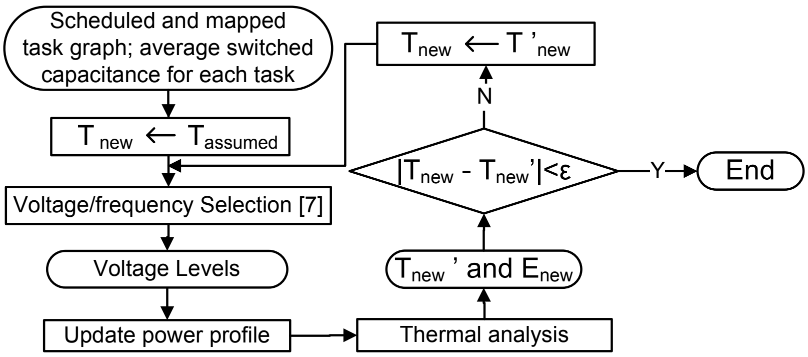

Scheduled and mapped task graph; average switched capacitance for each task
$T_{new} \leftarrow T_{assumed}$
Voltage/frequency Selection [7]
Voltage Levels
Update power profile
Thermal analysis
T_{new}' and E_{new}
$|T_{new} - T_{new}'| < \varepsilon$
Y
End
N
$T_{new} \leftarrow T'_{new}$

Figure 3.5: SDVFS with Leakage/Temperature Dependency

”assumed” temperature, at which each task is supposed to run, is also fixed as input. The voltage selection algorithm (outlined in Chapter 2.4 and [7]) will determine, for each task τ_i, the voltage level V_i such that energy consumption is minimized. Based on the determined voltage V_i (and the switched capacitances known for each task) the dynamic power profiles are calculated and the thermal analysis is performed as discussed in Section 3.2. Depending on what the designer selects, a unique temperature (produced by static temperature analysis, see Section 3.2.1) or a dynamic temperature profile (produced by dynamic temperature analysis, see Section 3.2.2) is determined for each task in the steady state, and the corresponding actual energy consumption E_{new} is computed. The new temperature/temperature-profile obtained from simulation in the current iteration is, then, used again for voltage selection in the next iteration and the process is repeated until the temperature/temperature-profile converges. Convergence means that the actual temperature values used at voltage selection correspond to the temperature at which the chip will function when running with the calculated voltages. It is also important to notice that during thermal analysis, potential thermal runaway is detected.

Fig. 3.6 shows a typical temperature convergence curve for the process. The squares marked with circles indicate the temperature produced after each iteration. As a basic technique, this new temperature (in the case of dynamic analysis, this new temperature profile) is used as input to the voltage selection in the next iteration. The squares represent successive temperatures in the inner iteration loop for temperature analysis (Fig. 3.3). As convergence criterion, a maximal temperature difference of 0.2° has been used. Based on our experiments (Section 3.5), up to 90% of the cases reach convergence after less than five iterations (both for static and dynamic temperature analysis).

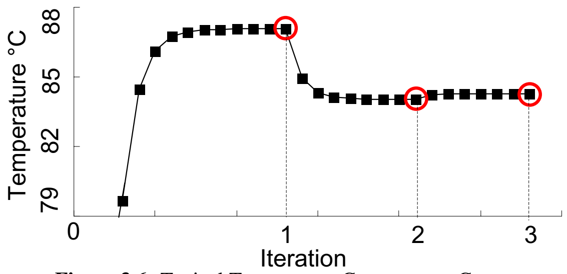


Figure 3.6: Typical Temperature Convergence Curve

Since the voltage levels available for selection are discrete and limited, our iteration approach is not guaranteed to reach a convergence. There are situations in which the temperature oscillates and the temperature updating technique described above leads to an infinite loop. This has happened for around 2.5% of our experiments. To overcome this problem, oscillations are detected and are solved by changing the temperature update rule: instead of using the just produced temperature for the next iteration, a middle value between the new temperature and the one produced in the previous iteration is used (in the case of dynamic temperature analysis, the points on the temperature profile are recalculated accordingly). By using this technique, all infinite loops occurring in our experiments have been solved.

3.3.2 SDVFS with Both Leakage/Temperature and Frequency/Temperature Dependencies (SDVFS-LF)

Our static DVFS approach which also considers the frequency/temperature dependency is based on the above iterative technique. The successive iterations lead, after convergence, to a temperature profile which corresponds to the one at which the chip will work. For each task τ_i the above voltage/frequency selection algorithm calculates a certain supply voltage V_i such that energy consumption is minimized and deadlines are satisfied. To take frequency/temperature dependency into consideration, when calculating the frequency setting for τ_i, we now consider the thermal profile of the task and determine the maximum temperature at which that task runs. At voltage/frequency selection, the frequency is calculated based on Eq. (2.4) and Eq. (2.5) (instead of being fixed, in a conservative way, considering the worst case temperature T_{max} for which the chip is designed).

3.4 Dynamic Temperature-Aware DVFS (DDVFS)

The above static approach determines start times for tasks and their voltage/frequency levels assuming that they execute their WNC. By this, only static slack is considered for energy minimization [1]. In order to exploit the dynamic slack, at the termination of each task and before starting the next one, voltage and frequency settings have to be determined based on the values of the current time and temperature. In principle, calculating the appropriate voltage/frequency settings implies the execution of the temperature-aware DVFS algorithm from Section 3.3. Running this algorithm on-line, after each task execution, implies a huge time and energy overhead which can be even higher than the execution time and energy consumption of the actual application.

3.4.1 Off-line and On-line Phases

To overcome the above problem, we have divided our dynamic DVFS approach into two phases. In the first phase, performed off-line, voltage/frequency settings for all tasks are pre-computed, based on possible start times of the tasks and the possible temperatures at that start time. The resulting voltage/frequency settings are stored in look-up tables (LUTs), one for each task. In Fig. 3.7 we show two such tables. They contain voltage and frequency settings for combinations of possible start time ts and start temperature Ts of a task. For example, the line in LUT$_2$ with start time 1.3ms and start temperature 55°C stores the voltage and frequency setting for the situation when τ_2 starts in the time interval (1.2ms, 1.3ms] and the start temperature is in the interval (45°C, 55°C]. In Section 3.4.2 we will present the generation procedure of the LUTs.

The second phase is performed on-line and is illustrated in Fig. 3.7. Each time a task terminates and a new voltage/frequency level has to be fixed for the next task, the on-line scheme looks up the appropriate setting from the LUT, depending on the actual time and temperature reading. If there is no exact entry in the LUT

[1]It should be mentioned that, as opposed to the dynamic one, the static approach can be used even in the case that no temperature sensors are available on the chip.

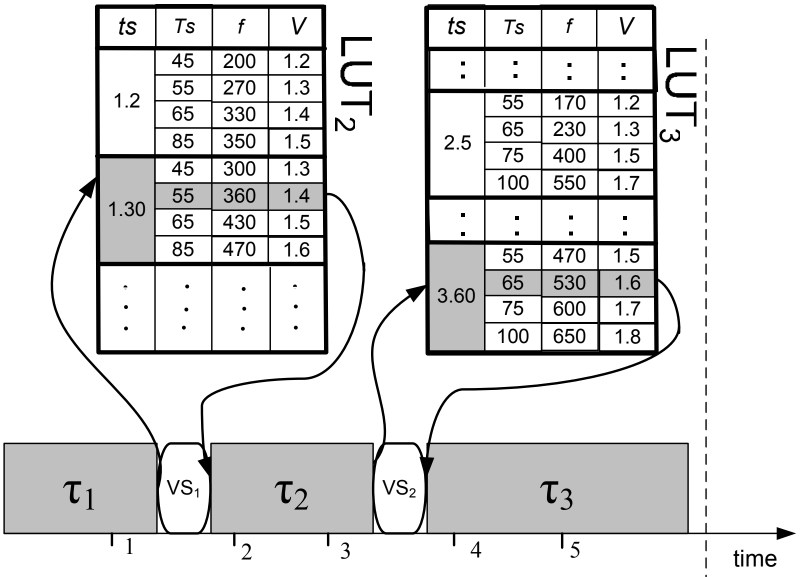


Figure 3.7: On-Line Phase

corresponding to the actual time/temperature, the entry corresponding to the immediately higher time/temperature is selected. For example, in Fig. 3.7, τ_1 finishes at time 1.25ms with a temperature 49°C. To determine the appropriate voltage and frequency for τ_2, LUT$_2$ is accessed based on these time and temperature values. There is no exact entry for 1.25ms and 49°C, so the entry corresponding to start time 1.3ms and start temperature 55°C is chosen. This on-line phase indicated with VS in Fig. 3.7 is of very low time complexity O(1) and, thus, very efficient.

3.4.2 LUT Generation

Given a set of tasks $(\tau_1, \tau_2, \ldots, \tau_n)$ (as described in Chapter 2.2 and Chapter 2.3) which are executed sequentially in the order, τ_1, τ_2,..., τ_n, on a DVFS enabled processor, our goal is to generate a LUT for each task τ_i, such that the energy consumption during execution is minimized. It is important to notice that the voltage levels and frequencies are calculated such that the energy consumption is optimal when the tasks execute their expected number of cycles ENC (which, in reality, happens with a much higher probability than e.g. the WNC). Nevertheless, voltages and frequencies are fixed such that, even in the worst case (tasks execute their WNC), deadlines are satisfied.

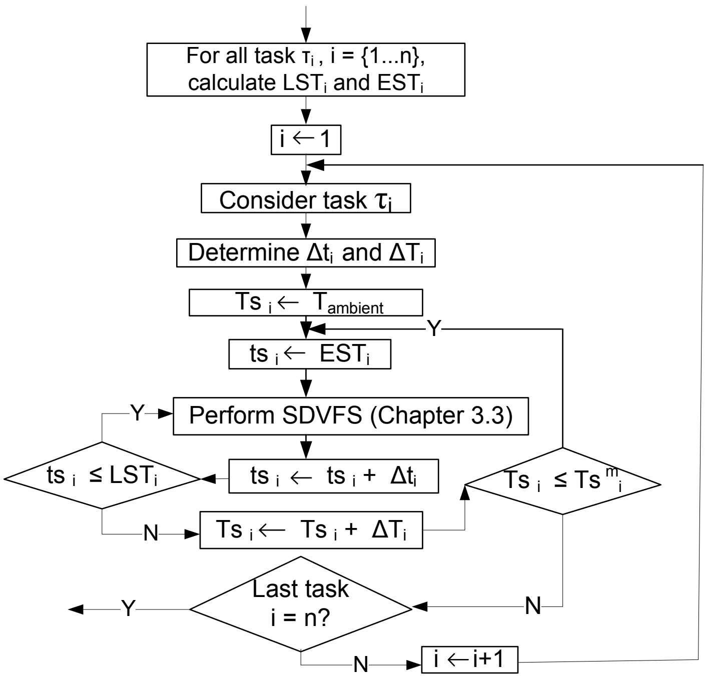


Figure 3.8: LUT Generation

The LUT generation algorithm is presented in Fig. 3.8. The outermost loop iterates over the set of tasks and successively constructs the table LUT$_i$ for each task τ_i. The next loop generates the entries of LUT$_i$ corresponding to the various start temperatures Ts_i of τ_i. Finally, the innermost loop iterates, for each possible start temperature, over all considered start times of task τ_i, ts_i. The algorithm starts by computing the earliest and latest possible start times for each task. The earliest start time of task τ_i, EST_i, is calculated based on the situation that all tasks execute with their best case number of cycles, BNC, at the highest voltage setting and lowest temperature (the ambient temperature). The latest start time LST_i is calculated as the latest start time of τ_i that still allows to satisfy the deadlines for future tasks in the current iteration, $\tau_j, j \geq i$, when executed with the worst case number of cycles, WNC, at the highest voltage and the maximum temperature T_{max} allowed for the chip.

Considering the intended granularity of the LUT, the time and temperature quanta $\triangle t_i$ and $\triangle T_i$ are determined. Thus, for task τ_i, the number of time entries (the number of different start times considered) will be $\lceil (LST_i - EST_i)/ \triangle t_i \rceil$, while, for each possible start time, the number of temperature entries is $\lceil (Ts_i^m -$

$T_a)/ \triangle T_i\rceil$, where Ts_i^m is the maximum possible temperature at the start time of τ_i. In Sections 3.4.3 and 3.4.4 we will further elaborate on the granularity and size of the LUTs.

When calculating the actual LUT entries for a task τ_i, the calculation of the voltage and frequency setting is performed by running the SDVFS algorithm outlined in Section 3.3, for all tasks τ_j, $j \geq i$, considering ts_i and Ts_i as start time and starting temperature, respectively, for τ_i.

3.4.3 Temperature Bounds and Granularity

As discussed before, the number of entries generated in LUT$_i$ along the temperature dimension is $\lceil (Ts_i^m - T_a)/ \triangle T_i\rceil$. The basic idea is that the lowest possible temperature is the temperature of the ambient, while Ts_i^m is the highest possible temperature, in the worst case, at the start time of task τ_i. But what is the value of Ts_i^m ? One solution is to consider for Ts_i^m the maximum temperature T_{max} at which the chip is allowed to work. While this assumption is safe, it leads to unnecessarily large tables since, during the execution of most of the tasks, the chip will never reach temperatures close to T_{max}. In order to avoid unnecessarily large tables, we need a safe but tighter upper bound on the temperature Ts_i^m . In order to achieve this goal, our LUT generation algorithm in Fig. 3.8 is executed several times in successive iterations before the final LUT tables are obtained.

We start by considering that for the first task the maximal starting temperature is the ambient temperature ($Ts_1^m = T_a$). The two inner loops in Fig. 3.8 will generate LUT$_1$. As part of the SDVFS procedure executed during generation of LUT$_1$ (see Section 3.3 and Fig. 3.5), we obtain the possible temperature profiles of τ_1 and, thus, also the peak temperature T_1^{peak} reached during execution of this task. The worst case starting temperature of task τ_2 is $Ts_2^m = T_1^{peak}$. Considering this value for Ts_2^m, table LUT$_2$ is generated and the procedure is continued for all tasks τ_i. After the algorithm in Fig. 3.8 has been executed once, we have all LUT tables, based on the assumption that the maximal possible temperature at the start of τ_1 is equal to T_a. This, however, is not the case, since the application is executed periodically and τ_1 is started again after the last task τ_n. Thus, in fact, the maximal starting temperature of τ_1 is, in the worst case, equal to the worst case peak temperature of τ_n. Therefore, we repeat the LUT generation algorithm, this time considering that $Ts_1^m = T_n^{peak}$. This will lead to a higher T_1^{peak} than in the previous iteration and, thus, a new larger $Ts_2^m = T_1^{peak}$. Thus, new lines will be generated in the LUTs. The procedure is continued iteratively, until, for a certain task, the peak temperature over two successive iterations does not change, which means that no new entries into the LUT tables will be generated. Our experiments have shown that convergence is reached after no more than 5 iterations. This procedure also allows to detect if there exists a possibility for the design to reach, in the worst case, a thermal runaway situation (in which case the iterations do not converge) or if the maximum allowed temperature can be violated (the process convergence but there are peak temperatures which are beyond T_{max}).

The above technique leads to a tightening of the range of temperatures in the LUT. There are two more questions to be answered regarding the number of temperature entries: (1) What should be the granularity of the temperature investigation and (2) how to reduce the number of entries if only a limited amount of memory is available at run-time?

It is obvious that a finer granularity and larger number of entries will, potentially, produce better energy savings at the cost, however, of increased memory consumption. With regard to the granularity $\triangle T_i$, our experiments have shown that values around 15°C are appropriate, in the sense that finer granularities will only marginally improve energy efficiency. If, due to memory limitations, we only can afford a certain number NT_i of temperature entries to be stored for a task τ_i, we have to decide which lines of LUT$_i$ to preserve and which to eliminate. One straightforward approach would be to maintain an even distribution of the selected NT_i lines over the range $[T_a, Ts_i^m]$. However, start temperatures of tasks, during execution, do not spread evenly over this range. Thus, it is more efficient to have the NT_i lines more dense around the temperature values that are more likely to happen, and sparse towards the extremes. This means that less pessimistic voltage/frequency settings will be used for the most likely cases, while cases that are much less likely to happen are handled in a more pessimistic way. Thus, after the LUT tables have been generated, in order to select the appropriate NT_i lines along the temperature dimension for each task τ_i, we run a temperature analysis session in which all tasks are executed for their expected number of cycles ENC. From this analysis, we can observe which is the most likely starting temperature for each task and we select the NT_i lines among those close to this most likely temperature.

3.4.4 LUT Granularity Along the Time Dimension

A straightforward approach would be to allocate the same number of entries, along the time dimension, to each task (Nt_i is the same for all tasks τ_i, $i = 1..N$). However, the start time interval sizes $LST_i - EST_i$ can differ very much between tasks, which should be taken into consideration when deciding on the number of time entries. Therefore, given a total number of entries along the time dimension NL_t, we determine the number of time entries in each LUT$_i$, as shown in Eq. (3.1)[2]:

[2]Let us mention that, while the start time intervals' sizes, $LST_i - EST_i$, are very different from task to task, this is much less the case with the size of the temperature interval, $Ts_i^m - T_a$. Therefore,

$$Nt_i = \left\lceil NL_t \cdot \frac{(LST_i - EST_i)}{\sum_{i=1}^{N} (LST_i - EST_i)} \right\rceil \tag{3.1}$$

3.4.5 Accounting for Analysis Accuracy and Ambient Temperature

The solutions produced by our techniques presented in section 3.4 are safe. By this we mean that:

1. It is guaranteed that deadlines are satisfied;
2. If, at run time, a certain frequency setting is selected for a task τ_i, it is guaranteed that the temperature during execution of τ_i will not exceed the limit allowed for the chip to run at the selected frequency.

There are two aspects which have to be discussed with respect to the second of the two statements above. First is the issue of ambient temperature. If a task τ_i is starting its execution at a certain temperature T, the temperature profile during task execution depends on the actual ambient temperature. Thus, a safe frequency selection has to also take into consideration the current ambient temperature. Two possible solutions can be considered:

1. Generate the voltage/frequency settings considering the highest ambient temperature under which the system is supposed to function. This is a safe but pessimistic solution with, potentially, smaller energy savings.
2. Generate alternative voltage/frequency settings for a set of ambient temperatures in the range assumed for the system to function. During run time, using sensors for the ambient temperature, the system will switch to those tables corresponding to that ambient temperature that is immediately higher than the actual measured one. This solution requires additional memory for storing a larger amount of tables but could lead to better energy efficiency.

The second aspect to be considered is the accuracy of the temperature analysis. The fact that a certain frequency setting is safe, with regard to the peak temperature reached during execution of a task, is based on the temperature analysis performed as part of the DVFS procedure. Thus, the results can be safe only to the extent to which this analysis provides safe temperatures. Of course, system-level thermal analysis

the number of entries along the temperature dimension (NT_i, see section 3.4.3) has been kept identical for all tasks in our experiments.

tools are not provably accurate. Nevertheless, relative precisions are reported for the various analysis tools and we are using this information in order to account for the inaccuracy of the thermal analysis. More precisely, given a certain relative precision of the temperature analysis tool that we use, we account for this precision in a conservative way when determining the peak temperatures used for frequency calculation.

In section 3.5, we will evaluate the impact of both ambient temperature and potential analysis inaccuracy on the energy optimization results.

3.5 Experimental Results

In this section, we perform experiments aiming at evaluating our DVFS approaches presented in Section 3.3 and Section 3.4.

3.5.1 Static DVFS Approach

Leakage/Temperature Dependency.
The first set of experiments is performed to compare the SDVFS approach with the consideration of leakage/temperature dependency (denoted T-SDVFS) as illustrated in Fig. 3.5, and a SDVFS approach that ignores the dependency (denoted as NT-SDVFS). NT-SDVFS is realized by running one single iteration of the process in Fig. 3.5. The assumed temperature is used for voltage selection which produces the voltage levels; temperature analysis gives the real temperature at which the chip will run using the obtained voltages, based on which the final energy consumption is computed.

We have generated 150 test applications which are running at temperatures in the range 40°C to 125°C. The number of tasks in each application is in the range [5, 100]. The WNC of the tasks is in the range [10^6, 10^7] cycles. 9 voltage levels in the interval [0.6V, 1.4V] were considered for dynamic supply voltage selection. The temperature model related coefficients are the same as in [19]. The chip working temperature is in the range $[T_a, T_{max}]$ where T_a and T_{max} are the ambient temperature and maximal allowable chip working temperature respectively. It is assumed that T_{max} = 125°C and $T_a = 40$°C. Leakage power is computed by Eq. (2.2) with parameters from [59]. The amount of leakage power (calculated at 70°C) is, on average, 40-60% of the total power. The frequency corresponding to a supply voltage is computed by Eq. (2.4) using parameters from [55].

For each generated test application we perform the following steps:

1. Apply our T-SDVFS algorithm illustrated in Fig. 3.5 and obtain the optimized, temperature-aware, energy consumption E_{ta}.

2. For the same application and setting, we run the NT-SDVFS approach resulting in energy consumption E_{nta}.

3. The energy reduction by using our T-SDVFS approach is computed as follows.

$$G = \frac{(E_{nta} - E_{ta})}{E_{nta}} \times 100\%$$

It is expected that the energy E_{ta} produced by the temperature-aware approach is smaller than E_{nta}. Obviously, E_{nta} depends on the assumed temperature provided by the designer. If the designer's guess is correct (equal to the temperature at which the chip functions with the selected voltages), a situation which is very unlikely, then E_{nta} = E_{ta}. As further away the designer's guess is, as larger E_{nta} is compared to E_{ta}. Fig. 3.9 shows the average energy reduction G as a function of how far the temperature guess is from the actual temperature at which the application runs. As

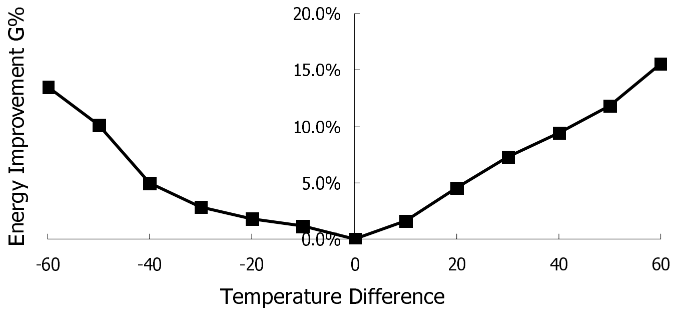


Figure 3.9: Energy Improvement with T-SDVFS Approach

can be seen, considerable amount of energy savings are achieved by our thermal-aware SDVFS approach. In the context in which it is practically impossible to predict at which temperature the circuit will function, since the actual voltages are not known before voltage selection, a thermal-aware approach is a safe solution if energy losses are to be avoided.

It is interesting to observe that, when temperatures are underestimated, the energy losses are slightly smaller. The explanation is the following: when temperatures are overestimated, the temperature-unaware approach assumes that leakage currents are very high (due to the high assumed temperature). Thus, the voltage selection algorithm will tend to select high supply voltages so that tasks are terminated early and slack time is used to put the circuit into low leakage modes. Since, in reality, the circuit will work at lower temperature and leakage currents will be considerably smaller (due to the exponential dependency of leakage on temperature, which at high temperature values leads to larger errors than at low temperatures), the temperature-aware approach will produce smaller supply voltages, which explains the energy differences at overestimated temperature. In the case of temperature underestimations, the temperature-unaware DVFS approach will produce lower voltages (which extend the execution time in the limits of available slack) and, by this, find solutions that are close to those produced by the temperature-aware approach.

We have also applied our temperature-aware voltage selection T-SDVFS on a real-life examples: an MPEG2 decoder which consists of 34 tasks and is described in more detail in [60]. Fig. 3.10 shows the energy reduction G as a function of how far the temperature guess is from the actual temperature at which the MPEG2 decoder runs. As can be observed, the trend is similar to the one produced for the generated applications.

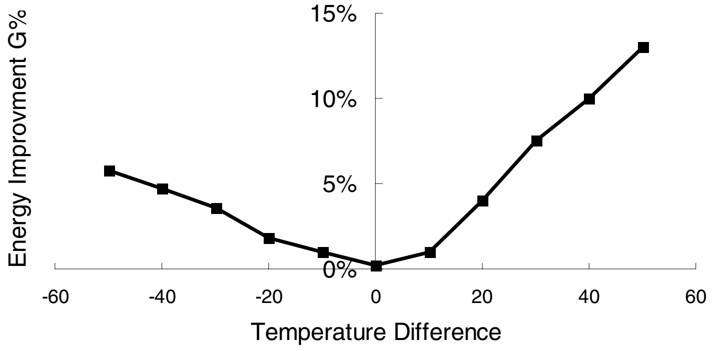


Figure 3.10: Energy Improvement for The MPEG2 Decoder

Frequency/Temperature Dependency.
A second set of experiments is performed to evaluate our temperature-aware DVFS approach SDVFS-LF, which takes both leakage/temperature and frequency/temperature dependency into consideration as described in Section 3.3.2.

150 test applications were generated and 9 discrete voltage levels in the range [1.0V, 1.8V] were used for dynamic voltage and frequency selection. The frequency for the SDVFS-LF approach is determined by Eq. (2.4) and Eq. (2.5). For Eq. (2.5), we use the coefficients μ = 1.19, ξ = 1.2, and $k = -1.0\text{V}/°\text{C}$ according to [19] and [61]. For each application, we perform both T-SDVFS and SDVFS-LF, and the corresponding energy consumptions are denoted by $E1_{ta}$ and $E2_{ta}$ respectively. We compute the energy reduction obtained from using SDVFS-LF compared to T-

SDVFS as follows:

$$\frac{E1_{ta} - E2_{ta}}{E1_{ta}} \times 100\%$$

As shown in Table 3.4, the energy consumption can be reduced by 22% on average over all test applications. For our real-life example: the MPEG2 decoder, the corresponding energy reduction is 21%.

Table 3.4: Energy Reduction from Using SDVFS-LF Comparing with T-SDVFS

	Energy Reduction
Generated Applications	22%
MPEG2 Decoder	21%

3.5.2 Dynamic DVFS Approach

In this section, we evaluate our on-line temperature-aware dynamic DVFS (DDVFS) approach described in Section 3.4. 150 applications were generated and each generated application consists of 2 to 50 tasks. The WNC of the tasks is in the range [10^6, 10^7] cycles. The test applications are executed on a processor which can run at 9 different supply voltage levels in the range [1.0V, 1.8V]. It is important to mention that in all our experiments, we have accounted for the time and energy overhead produced by the on-line component of our dynamic approach. Similarly, we have also taken into consideration the energy overhead due to the memory accesses. This overhead has been calculated based on the energy values given in [62] and [63].

Frequency/Temperature Dependency.

We have first performed experiments to explore the benefits of considering the frequency/temperature dependency with dynamic DVFS. Table 3.5 shows the energy reduction obtained by taking the frequency/temperature dependency into consideration with DDVFS compared to the similar approach but ignoring the dependency. The energy consumption is reduced by 17% on average, for our generated applications. In the case of our real life example, the MPEG2 decoder, the energy consumption reduction is 18.6%.

Table 3.5: Energy Improvement by DDVFS with Frequency/Temperature Dependency

	Energy Reduction
Generated Applications	17.3%
MPEG2 Decoder	18.6%

DDVFS vs. SDVFS.
This set of experiments is aimed at comparing the energy consumption between the static DVFS approach, presented in Section 3.3, and the dynamic one, presented in Section 3.4 (both considering the two pairs of dependencies: leakage/temperature and frequency/temperature). As the ratio BNC/WNC has a strong influence on the potential efficiency of a dynamic approach, we run the experiments considering three different ratios: 20%, 50%, and 70%. We also assume that the workload distribution of each task conforms to a normal distribution $N(ENC, \sigma^2)$, where ENC is the mean value, and σ is the standard deviation. For our energy evaluations we have generated actual numbers of executed clock cycles for each task considering standard deviations of $(WNC-BNC)/3$, $(WNC-BNC)/5$, $(WNC-BNC)/10$, and $(WNC-BNC)/100$.

Fig.3.11 shows the energy savings with the dynamic approach DDVFS relative to the static one SDVFS. As can be observed, the efficiency of the dynamic approach, compared to the static one, increases as the ratio between BNC and WNC becomes smaller. The energy savings are also larger, compared to the static approach, when the standard deviation σ is smaller (more of the actual executed number of clock cycles are clustering around the ENC). Remember that our DDVFS algorithm is targeted towards optimizing the energy consumption for the case when tasks execute the expected number of cycles ENC.

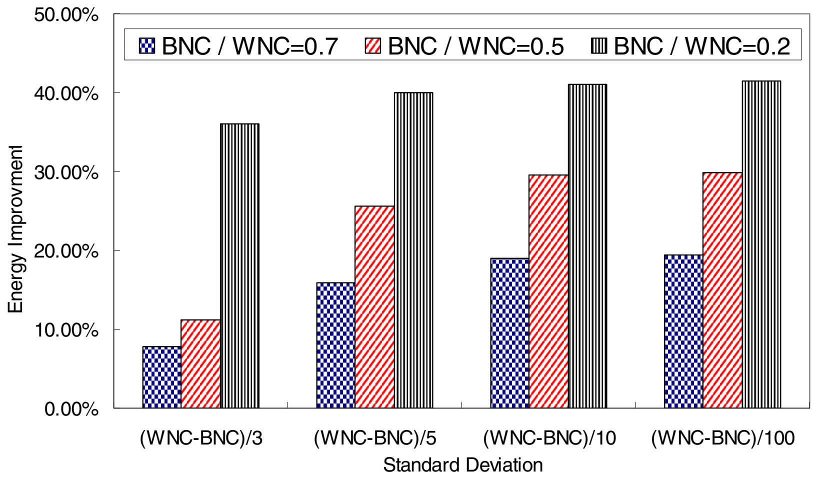


Figure 3.11: Dynamic vs. Static Approach

We also apply our dynamic approach to the MPEG2 decoder mentioned in Section 3.5.1. The energy consumption with the dynamic approach is 39% smaller than the one using the static DVFS approach.

Computation Time of the Off-line Phase of DDVFS.

For the above test applications, the computation time needed for LUT generation in the off-line phase is shown in Fig. 3.12. The computation time is illustrated as a function of the number of tasks in the application. For large size application, e.g. an application containing 50 tasks, the needed time for generating all the LUT tables off-line is around 400 minutes.

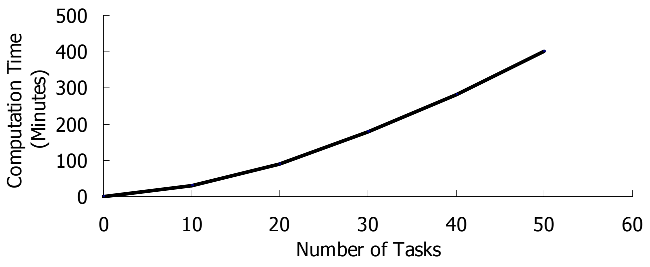


Figure 3.12: Computation Time: Off-line Phase

LUT Sizes.

The next set of experiments are aimed at exploring the impact of the LUT sizes on the efficiency of the DDVFS approach. In particular, we are interested in the impact of the number of entries along the temperature dimension. The number of entries along the time dimension has been kept constant for these experiments and is distributed according to the discussion in section 3.4.4. First we run, for all applications, our dynamic DVFS approach considering a granularity $\triangle T = 10°$. We evaluate the average energy reduction with the obtained LUTs, compared to the static approach. Then we impose a certain limitation on the number of entries along the temperature dimension and we construct the corresponding LUTs as discussed in Section 3.4.3. We again evaluate the energy consumption considering these reduced LUTs.

The diagram in Fig. 3.13 shows the average results for different number of entries under two different standard deviations of the actual number of clock cycles executed by tasks. Having one single temperature entry will produce energy reductions compared to the static case which are 37% smaller (for $\sigma = (WNC - BMC)/3$) than with an unreduced LUT. However, with 2 entries, the results are already very close to those obtained with an unreduced LUT and with 3 entries they are, in practice, identical. This is good news, since it shows that significant energy savings can be obtained with relatively small memory overhead. It should be mentioned that all other experiments presented in this section have been performed with 2 entries along the temperature dimension.

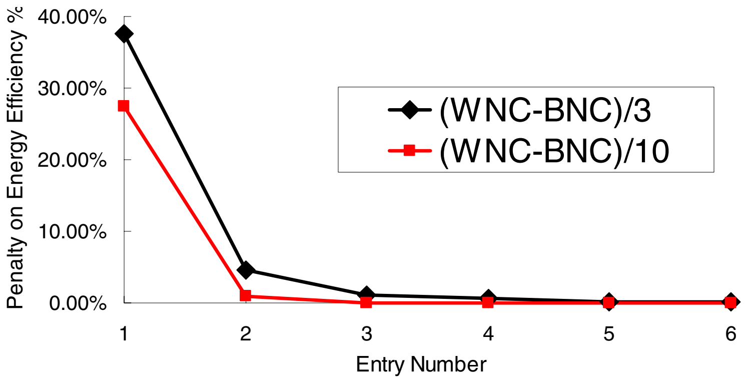


Figure 3.13: Impact of Temperature Line Number

Accounting for Ambient Temperature and Thermal Analysis Accuracy. We have also performed experiments to explore the impact of ambient temperature and temperature analysis accuracy. For all experiments above, we have assumed that T_a is 40°C and is known at design time. In order to evaluate the impact of the ambient temperature, we considered all the generated applications and constructed LUTs for values of T_a in the range [-10°C, 40°C]. For each (application, LUTs) pair corresponding to a certain T_a we have evaluated the energy consumption considering that the T_a is identical with the one assumed at LUT generation. Then we run the simulations for the same (application, LUTs) pair, but considering that T_a deviates with 10°, 20°, . . . , 50° from the value assumed at design time. The results are shown in Fig.3.14. We can see that if T_a is different by, for example, 20° from the one assumed at design time, the energy consumption increases by only 7% on average. This shows that, if the predicted range of ambient temperature is, for example, 40°, generating two sets of LUTs (granularity of 20°) will lead to energy losses, on average, less than 7%.

All the above experiments have been performed considering that the temperature modeling and analysis is accurate. We have repeated the experiments considering a relative accuracy of 85%. When calculating frequency settings we accounted, in a conservative way, for this degree of accuracy. In Table 3.6 we compare the energy efficiency achieved with simulation accuracy 85% and the one with 100%

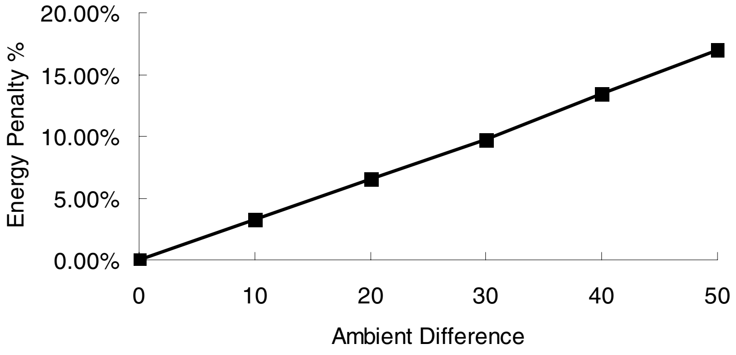


Figure 3.14: Impact of The Ambient Temperature

Table 3.6: Energy Improvement Degradation by Simulation Accuracy

	100%	85%	Degradation
Generated Applications	17.3%	16.7%	3.5%
MPEG2 Decoder	18.6%	18.1%	2.8%

simulation accuracy for both our generated examples and the MPEG2 decoder. As shown in Table 3.6, the degradation due to considering for the 15% inaccuracy is less than 3.5%.

Chapter 4

Temperature-Aware Idle Time Distribution

4.1 Motivational Example

4.1.1 Static Idle Time Distribution

Let us consider a periodic application consisting of 7 tasks which share a global deadline of 96.85ms. The worst case workload, WNC (in clock cycles), and average switched capacitance, $Ceff$, for each task are shown in Table 4.1. The tasks run on a processor with a fixed supply voltage and frequency level of 0.6V and 132MHZ respectively. The corresponding execution times te^W are also shown in Table 4.1.

Table 4.1: Motivational Example: Application Parameters

	WNC	$Ceff$(f)	te^W(ms)
τ_1	8.26e+06	5.0e-10	6.22
τ_2	1.20e+07	5.0e-10	9.07
τ_3	2.32e+07	9.0e-8	18.76
τ_4	2.25e+07	1.7e-7	17.46
τ_5	1.46e+07	1.8e-7	16.94
τ_6	2.15e+07	1.9e-7	16.18
τ_7	8.26e+06	5.0e-10	6.22

Based on the performance of this processor, there exists 6ms static slack, ts, in each execution period of this application. Fig. 4.1 shows two different ways of distributing ts. The first distribution (1st ITD), as shown in Fig. 4.1a, places the

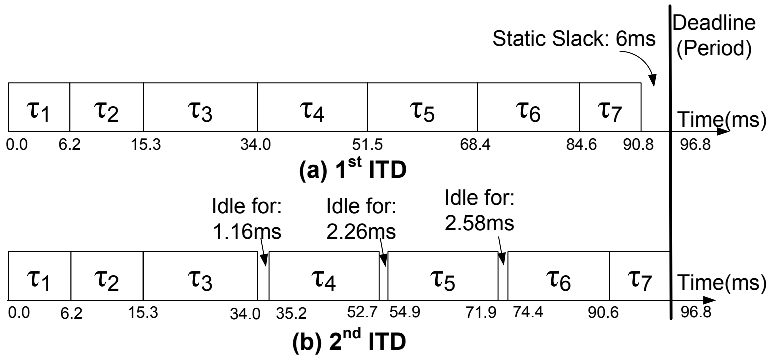


Figure 4.1: Motivational Example: Static Idle Time Distribution

whole ts after the last task, while the second distribution (2nd ITD), in Fig. 4.1b, divides the static slack ts into 3 segments of idle slots and places the 3 idle slots after execution of task τ_3, τ_4 and τ_5, respectively.

For simplicity, in this example, we ignore both energy and time overhead due to switching between the active and idle mode. The two different ITDs will lead to different temperature and leakage power profiles. The average working temperature Tw of each task, as well as the leakage energy consumption, are shown in Table 4.2, where E^{leak} is the leakage consumption of each task. E^{leak}_{tot} is the total leakage energy consumption of the whole application. Comparing E^{leak}_{tot} for the 1st and 2nd ITD, we can observe that around 10% reduction of leakage energy consumption can be achieved.

Table 4.2: Static ITD: Leakage Energy Comparison

	1st ITD		2nd ITD	
	Tw(°C)	E^{leak}(J)	Tw(°C)	E^{leak}(J)
τ_1	101	0.81	110	0.96
τ_2	102	1.20	107	1.30
τ_3	108	2.73	108	2.73
τ_4	119	3.08	113	2.78
τ_5	125	3.32	115	2.79
τ_6	129	3.39	117	2.68
τ_7	122	1.24	117	1.05
E^{leak}_{tot}		15.77		14.29

The leakage energy reduction is due to the modified working temperature of the chip which has a strong impact on the leakage power. It is also important to mention that the table reflects the steady state (not the start-up mode), for which energy minimization is targeted. This means that the starting temperature for τ_1 is identical to the temperature at the end of the previous period.

4.1.2 Dynamic Idle Time Distribution

The ITD approach outlined in the previous section is an off-line static one which assumes that tasks execute their WNC and, thus, it only distributes the static slack. However, in reality, most of the time, there are huge variations in the number of cycles executed by a task, from one activation to the other, which leads to a large amount of dynamic slack.

For the task set introduced in the previous section, let us imagine the activation scenario shown in Table 4.3, where the columns ANC and te^A contain the actual executed workload (in clock cycles) and the corresponding actual execution time of each task, respectively. td^i represents the dynamic slack generated due to the actual number of cycles executed by task τ_i (it is the difference between the te^W and te^A of the task). For this activation scenario, tasks τ_3, τ_4, τ_5 and τ_6 execute

Table 4.3: Motivational Example: An Activation Scenario

	ANC	te^W (ms)	te^A(ms)	td^i(ms)
τ_1	5.95e+05	6.22	0.45	5.77
τ_2	5.20e+05	9.07	0.40	8.67
τ_3	2.49e+07	18.76	18.76	0.0
τ_4	2.32e+07	17.46	17.46	0.0
τ_5	2.25e+07	16.94	16.94	0.0
τ_6	2.15e+07	16.18	16.18	0.0
τ_7	2.60e+06	6.22	1.96	4.26

their worst case workload, while τ_1, τ_2 and τ_7 execute less than their worst case workload and, thus, generate dynamic slack. The total amount of dynamic slack is $td = \sum_{i=1}^{7} td^i = 18.7$ms.

Fig. 4.2a illustrates the distribution of idle time slots during the above on-line activation scenario if we use the off-line ITD approach which distributes static slack as illustrated in Fig. 4.1b. In this case, the dynamic slack td^i is placed where it is generated (td^i is placed after τ_i terminates). Table 4.4 shows the corresponding working temperature and leakage energy consumption of each task as well as the total leakage energy consumption, which is 7.98J. However, leakage energy can be reduced by distributing the dynamic slack more wisely. For example, at

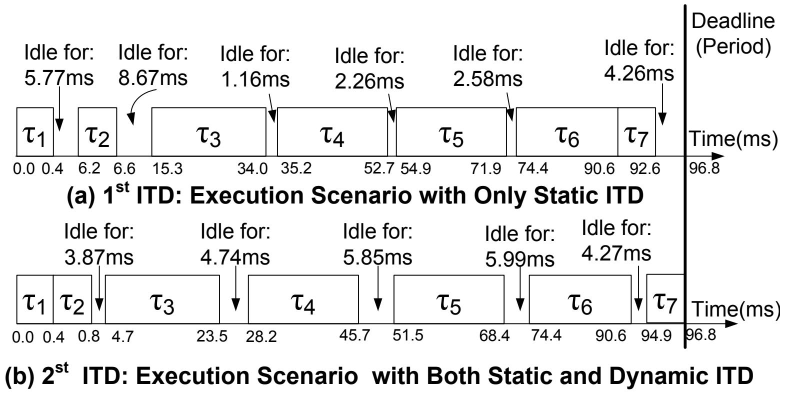


Figure 4.2: Motivational Example: Idle Time Distribution

run-time, whenever a task terminates, the idle time slot length following this task is calculated by taking into consideration the current time and the current chip temperature. Fig. 4.2b shows the ITD determined in this way. The corresponding total leakage energy consumed, as shown in Table 4.4, is 7.32J which means a leakage energy reduction of 8%. This leakage reduction is due to the further lowered working temperature of the energy hungry tasks τ_4, τ_5 and τ_6, which is achieved by ITD considering both static and dynamic slack.

Table 4.4: Dynamic ITD: Leakage Energy Comparison

	1st ITD		2nd ITD	
	Tw(°C)	E^{leak}(J)	Tw(°C)	E^{leak}(J)
τ_1	89	0.05	83	0.04
τ_2	78	0.03	83	0.04
τ_3	79	1.67	84	1.80
τ_4	91	1.92	87	1.78
τ_5	97	2.04	90	1.80
τ_6	99	2.02	91	1.73
τ_7	102	0.25	84	0.13
E^{leak}_{tot}		7.98		7.32

The above examples have demonstrated that leakage energy can be reduced through both static and dynamic idle time distribution.

4.2 Problem Formulation

We consider a set of periodic tasks $(\tau_1, \tau_2, \ldots, \tau_n)$ executed in the order τ_1, $\tau_2, \ldots,$ τ_n. For each task τ_i, as described in Chapter 2.2, the five-tuple:

$$< WNC_i, BNC, ENC, dl_i, Ceff_i >$$

is given. The supply voltage V_i at which the task τ_i is executed is fixed, e.g. by our DVFS technique proposed in Chapter 3. Corresponding to the supply voltage V_i that task τ_i is executed at, the worst case execution time te_i^W, best case execution time te_i^B, and expected execution time te_i^E can be directly calculated [1].

For each iteration of the application, the total static slack ts is constant and computed by Eq. (4.1):

$$ts = dl_n - \sum_{i=1}^{n} te_i^W \tag{4.1}$$

where dl_n represents the deadline of the last task τ_n in the execution order, and $\sum_{i=1}^{n} te_i^W$ is the sum of the worst case execution time of all tasks. The total dynamic slack for each execution iteration is varying due to execution time variation of tasks. For one iteration, td is calculated as follows:

$$td = \sum_{i=1}^{n} te_i^W - \sum_{i=1}^{n} te_i^A$$

where te_i^A represents the actual execution time of task τ_i in this iteration. te_i^A conforms to a distribution with expected execution time te_i^E as the arithmetic mean value of the probability density function $Pb(te_i^A)$ as follows:

$$te_i^E = \int_{te_i^B}^{te_i^W} Pb(te_i^A) \cdot te_i^A \, \mathrm{d}(te_i^A)$$

The total available slack t_{tot} for one iteration is equal to the sum of the static slack ts and dynamic slack td as shown in Eq. (4.2).

$$t_{tot} = ts + td \tag{4.2}$$

During t_{tot} the processor can be switched to idle mode consuming the power P_{idle}. The time and energy overhead for switching the processor to and from the idle state

[1]The frequency level corresponding to the supply voltage V_i is computed at the highest allowable working temperature of the processor T_{max}.

are t_o and E_o respectively. Idle slots can be placed after the execution of any task. The length of an idle slot i after task τ_i is denoted as t_i, and the sum of all idle slots $\sum_{j=1}^{n} t_i$ should be equal to the total available idle time t_{tot}. Note that the time overhead t_o is included in the slot length t_i.

We will, formulate the following two ITD problems.

- ITD with only static slack: static idle time distribution (SITD)
- ITD with both static and dynamic slack: static and dynamic idle time distribution (DITD)

4.2.1 ITD with Only Static Slack: SITD

Let us consider the scenario in which each task τ_i is always executed with the worst case workload: $te_i^A = te_i^W$. In this scenario, for each iteration, the available slack is constant and known: $t_{tot} = ts$ where ts is computed by Eq. (4.1).

For one iteration, the total energy consumption of the task set can be expressed as follows:

$$E^{tot} = \sum_{i=1}^{n} E_i^{dyn} + \sum_{i=1}^{n} E_i^{leak} + \sum_{i=1}^{n} (E_o \cdot x_i) + E^I$$

where $\sum_{i=1}^{n} E_i^{dyn}$ and $\sum_{i=1}^{n} E_i^{leak}$ are the total dynamic and leakage energy consumption of all tasks. $\sum_{i=1}^{n}(E_o \cdot x_i)$ is the total energy overhead when the processor is switched to/from idle state, where x_i is a binary variable indicating whether task τ_i is followed ($x_i = 1$) or not ($x_i = 0$) by an idle slot. E^I is the total energy consumption during the idle time t_{tot}.

The dynamic energy consumption of each task E_i^{dyn}, is further computed as:

$$E_i^{dyn} = P_i^{dyn}(V_i) \cdot te_i^W$$

where V_i is the supply voltage the task τ_i is executed at, and $P_i^{dyn}(V_i)$ computes the dynamic power according to Eq. (2.1). te_i^W represents the worst case execution time of task τ_i. As the supply voltage V_i and te_i^W are constants, the total dynamic energy $\sum_{i=1}^{n} E_i^{dyn}$ is hence constant and independent from the distribution of idle time.

The total energy consumption during idle time E^I is computed as:

$$E^I = P_{idle} \cdot t_{tot}.$$

where P_{idle} is the power consumption of the processor in the low power mode and $t_{tot} = ts$. Similar to E_i^{dyn}, E^I is also fixed and independent from ITD, as ts is constant with given supply voltages.

The leakage energy consumption of each task E_i^{leak} is a function of both temperature and supply voltage as expressed in Eq. (4.3):

$$E_i^{leak} = \int_0^{te_i^W} P_i^{leak}(V_i, T_i(t))\,\mathrm{d}t \tag{4.3}$$

where $T_i(t)$ describes the temperature of the processor during execution of task τ_i. $P_i^{leak}(V_i, T_i(t))$ calculates leakage power according to Eq. (2.2). With given supply voltages V_i, $T_i(t)$ is influenced by the distribution of idle time slots, so the leakage energy consumption E_i^{leak} is determined by the distribution of idle time slots.

We need to distribute the static slack ts to minimize the total leakage energy consumption of the application and the energy overheads due to switching as follows.

$$E = \sum_{i=1}^{n} E_i^{leak} + \sum_{i=1}^{n} (E_o \cdot x_i)$$

With given supply voltages V_i, and a fixed distribution of idle time slots, the same power pattern is periodically executed on the processor. As the task set is executed for a large number of iterations, the processor temperature is, thus, able to converge to a steady state dynamic temperature curve (SSDTC). Once the processor has reached the steady state, the SSDTC will repeat periodically. Our

Problem Formulation 1 Static Idle Time Distribution

$$E = \sum_{i=1}^{n} Es_i^{leak} + \sum_{i=1}^{n} (E_o \cdot x_i) \tag{4.4}$$

$$\text{subject to :}$$

$$ts = \sum_{i=1}^{n} t_i \tag{4.5}$$

$$dl_i \geq \sum_{j=1}^{i-1} t_i + \sum_{j=1}^{i} te_j^W \; (\forall i, 1 \leq i \leq n) \tag{4.6}$$

SITD problem can be formulated by Eq. (4.4)–Eq. (4.6). Given is a set of periodic tasks $(\tau_1, \tau_2, \ldots, \tau_n)$ as defined earlier in this section. The tasks are mapped and scheduled on the platform as described in Chapter 2.3. The idle time slot length t_i following each task τ_i and, implicitly, x_i (the binary variable which represents whether task τ_i is followed by an idle time slot or not) are to be determined such that the objective function Eq. (4.4) is minimized with the two constraints Eq. (4.5) and Eq. (4.6) to be satisfied. Es_i^{leak} in Eq. (4.4) represents the steady state leakage energy consumption. The constraint in Eq. (4.5) requires that the sum of all idle slots lengths should be equal with the total available static slack ts where ts is calculated by Eq. (4.1). The constraint in Eq. (4.6) guarantees that the deadline of each task is satisfied.

4.2.2 ITD with Both Static and Dynamic Slack: DITD

The above problem formulation ignores the execution time variations of tasks at run-time and, implicitly, ignores the dynamic slack. To deal with execution time variation and perform dynamic slack distribution, the idle slot length t_i following the termination of a task τ_i should be determined, at run-time, based on the actual time and processor temperature.

Our problem formulation for DITD is shown by Eq. (4.7)–Eq. (4.9). Given is

Problem Formulation 2 Dynamic Idle Time Distribution

$$\text{Minimize :}$$

$$E = \sum_{j=i+1}^{n} E_j^{leak} + \sum_{j=i}^{n} (E_o \cdot x_j) \tag{4.7}$$

$$\text{subject to :}$$

$$\sum_{j=i}^{n} t_j = dl_n - t_i^f - \sum_{j=i+1}^{n} te_j^E \tag{4.8}$$

$$dl_j \geq t_i^f + \sum_{k=i}^{j-1} t_k + \sum_{k=i+1}^{j} te_k^W \tag{4.9}$$

$$(\forall j, i+1 \leq j \leq n)$$

a set of periodic tasks $(\tau_1, \tau_2, \ldots, \tau_n)$ as defined earlier in this section. The tasks are mapped and scheduled on the platform as described in Chapter 2.3. When task τ_i terminates at time t_i^f, the idle time slot t_i following task τ_i's termination is determined such that Eq. (4.7) is minimized, where $\sum_{j=i+1}^{n} E_j^{leak}$ is the total leakage energy consumption of the remaining tasks τ_j, $(i < j \leq n)$, to be executed within the current iteration. The leakage energy consumption E_j^{leak} of each remaining task τ_j is estimated corresponding to the case when the expected workload is executed. E_j^{leak} is calculated according to Eq. (4.3) with the difference that the expected execution time te_j^E is used instead of te_j^W as the upper limit for the integral. The constraint in Eq. (4.8) requires that the sum of all idle slots lengths should be equal with the total available slack where t_i^f is the time the current task τ_i terminates. The total available slack is computed with the assumption that all the future tasks τ_{i+1} to τ_n are executed with their expected workload te_j^E $(\forall j, i < j \leq n)$. The deadline of each task is guaranteed by the constraint in Eq. (4.9), where dl_j represents the deadline of task τ_j. Note that, the worst case execution time te_k^W is used in Eq. (4.9) in order to guarantee the deadline of each task in the worst case.

4.3 Temperature Analysis

To solve the problem of ITD, we need to perform dynamic temperature analysis. In the previous chapter, the thermal analysis are based on Hotspot. However, using Hotspot for dynamic temperature analysis is not efficient due to its extremely long execution time. In this section, we propose an analytical thermal analysis approach which is fast and accurate enough to be used in our ITD optimization loop.

4.3.1 Temperature Model

Thermal Circuit.
In order to analyze the thermal behavior, we build an equivalent RC thermal circuit based on the physical parameters of the die and the package [24]. Due to the fact that the application period, t_p, can safely be considered significantly smaller than the RC time of the heat sink, which, usually, is in the order of minutes [64], the heat sink temperature stays constant after the state corresponding to the SSDTC is reached. For SSDTC estimation, we, hence, can ignore the thermal capacitance (not the thermal resistance!) of the heat sink and build the 2-RC thermal circuit shown in Fig. 4.3a. B_1 and B_2 represent the temperature node for the die and the heat spreader respectively. $P(t)$ stands for the processor power consumption as a function of time. We obtain the values of R_1, R_2, C_1 and C_2 from an RC network similar to the one constructed in Hotspot [25]. R_1 is calculated as the sum of the thermal resistance of the die and the thermal interface material (TIM), and C_1 as the sum of the thermal capacitance of the die and the TIM. R_2 is the equivalent thermal resistance from the heat spreader to the ground through the heat sink, and C_2 is the equivalent thermal capacitance of the heat spreader layer.

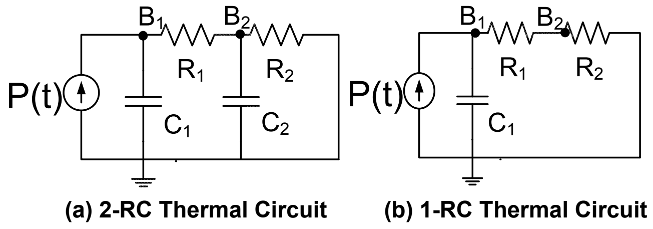


Figure 4.3: Thermal Circuit

When the application period t_p is significantly smaller than the RC time of the *heat spreader* in the 2-RC thermal circuit, the *heat spreader* temperature stays constant after SSDTC is reached. In this case, we can simplify the 2-RC to an 1-RC thermal circuit (Fig. 4.3b).

Temperature Equations.

For the 2-RC thermal circuit in Fig. 4.3a, we can describe the temperatures of B_1 and B_2 as follows:

$$C_1 \cdot \frac{dT^{die}}{dt} + \frac{T^{die} - T^{sp}}{R_1} = P(t) \tag{4.10}$$

$$C_2 \cdot \frac{dT^{sp}}{dt} + \frac{T^{sp}}{R_2} = \frac{T^{die} - T^{sp}}{R_1} \tag{4.11}$$

where T^{die} and T^{sp} represent the temperatures at B_1 and B_2 respectively. The power consumption, $P(t)$, is the sum of the dynamic and leakage power, which are dependent on the supply voltage V and T^{die}.

If, within a time interval, the power consumption P stays constant, the temperature at the beginning and end of the time interval can be expressed as follows, by solving Eq. (4.10) and Eq. (4.11):

$$T_e^{die} = a_1 \cdot T_b^{die} + b_1 \cdot T_b^{sp} + c_1 \tag{4.12}$$

$$T_e^{sp} = a_2 \cdot T_b^{die} + b_2 \cdot T_b^{sp} + c_2 \tag{4.13}$$

T_b^{die} and T_b^{sp} are the temperatures of B_1 and B_2 at the beginning of the time interval, while T_e^{die} and T_e^{sp} are the temperatures at the end of the time interval. a_1, a_2, b_1, b_2, c_1 and c_2 are constant coefficients determined by R_1, R_2, C_1, C_2 and P.

4.3.2 SSDTC Estimation

As an input to the SSDTC calculation we have the voltage levels, calculated by a DVFS algorithm, and a given idle time distribution, as illustrated in Fig. 4.4a.

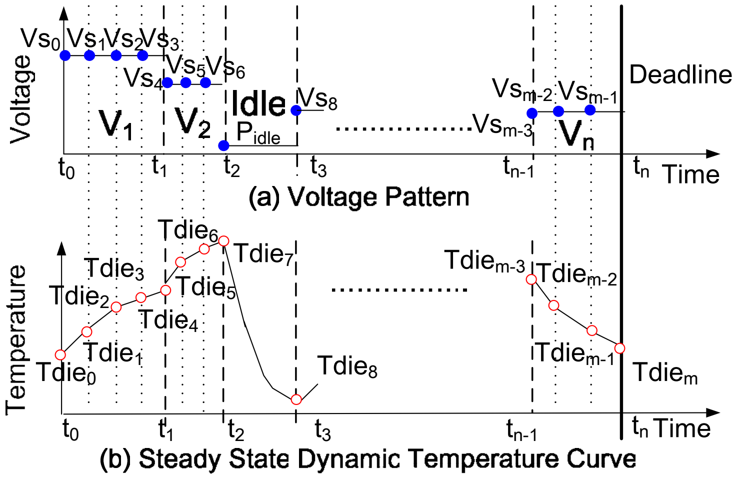


Figure 4.4: Temperature Analysis

When the processor is working in the active state, the leakage power consumption varies with the working temperature of the processor. In Fig. 4.4a, we divide the execution interval of each active state step into several sub-intervals. The total number of sub-intervals is denoted as m. Each sub-interval is short enough such that the temperature variation is small and the leakage power can be treated as constant inside the sub-interval.

P_i is the power consumption for each sub-interval i $(1 \leq i \leq m)$. When the processor is in the active state during the i^{th} sub-interval, P_i is computed by Eq. (4.14), where Vs_{i-1} and T_{i-1}^{die} are the supply voltage and processor temperature at the start of the i^{th} sub-interval. $P^{dyn}(Vs_{i-1})$ represents the dynamic power consumption while $P^{leak}(T_{i-1}^{die}, Vs_{i-1})$ represents the leakage power consumption based on the piece-wise linear leakage model discussed in Section 2.1.

$$P_i \quad = \quad P_i^{dyn}(Vs_{i-1}) + P_i^{leak}(T_{i-1}^{die}, Vs_{i-1}) \tag{4.14}$$

When the processor is in idle state during the i^{th} sub-interval, the power consumption $P_i = P_{idle}$.

As shown in Fig. 4.4b, we construct the SSDTC by calculating the temperature values T_0^{die} to T_m^{die}. The relationship between the start and end temperature of each sub-interval can be described by applying Eq. (4.12) and Eq. (4.13) to all sub-intervals. Thus, we can establish a linear system with $2m$ equations as shown in Eq. (4.15)–Eq. (4.18), where T_i^{die} and T_i^{sp} are the temperature of the processor and heat spreader at the beginning of the $i+1^{th}$ sub-interval. Due to periodicity, when dynamic steady state is reached, the processor and heat spreader temperature at the beginning of the period should be equal to the temperature values at the end of the previous period (Eq. (4.19)). Solving the linear system Eq. (4.15)–Eq. (4.19), we get the values for T_0^{die} to T_m^{die} and, hence, obtain the corresponding SSDTC. As this system is a tridiagonal linear system, it can be solved efficiently, e.g. through LU decomposition with only $O(m)$ operations [65]. It should be mentioned that, in fact, two SSDTCs can be obtained, one reflecting the temperature of the chip, and the other based on that of the heat spreader.

$$T_1^{die} = a_{11} \cdot T_0^{die} + b_{11} \cdot T_0^{sp} + c_{11} \tag{4.15}$$

$$T_1^{sp} = a_{12} \cdot T_0^{die} + b_{12} \cdot T_0^{sp} + c_{12} \tag{4.16}$$

$$\cdots\cdots\cdots$$

$$T_m^{die} = a_{m1} \cdot T_{m-1}^{die} + b_{m1} \cdot T_{m-1}^{sp} + c_{m1} \tag{4.17}$$

$$T_m^{sp} = a_{m2} \cdot T_{m-1}^{die} + b_{m2} \cdot T_{m-1}^{sp} + c_{m2} \tag{4.18}$$

$$T_0^{die} = T_m^{die};\ T_0^{sp} = T_m^{sp} \tag{4.19}$$

4.3.3 Transient Temperature Curve (TTC) Estimation

The temperature (SSDTC) calculated in the previous section corresponds to the dynamic steady state reached after a sufficient number of iterations have been executed. The same technique can be used to calculate any transient temperature curve (TTC), corresponding to an arbitrary time interval, as long as the length of the time interval is significantly smaller than the RC time of the heat sink (which is in the order of minutes). Under this assumption, as discussed earlier in this section, the thermal model in Fig. 4.3 can be used. The only difference relative to the SSDTC calculation is that Eq. (4.19) is no longer valid:

$$T_0^{die} \neq T_m^{die};\ T_0^{sp} \neq T_m^{sp}$$

To estimate the transient temperature curve (TTC), the temperature of T_0^{die} and T_0^{sp} are given as input. The temperature values: T_1^{die},T_2^{die}, $\ldots$, T_m^{die} and T_1^{sp}, T_2^{sp}, $\ldots$, T_m^{sp} are calculated by solving equations Eq. (4.15)–Eq. (4.18).

4.4 ITD with Only Static Slack (SITD)

In this section, we discuss our solutions to the SITD problem, as formulated in Section 4.2.1, which only considers static slack. We first introduce our approach ignoring the overheads E_o and t_o in Section 4.4.1. This approach will, then, be used in Section 4.4.2 where a general SITD technique with overheads consideration is presented.

4.4.1 SITD without Overhead (SITDNOH)

Since, in this section, we ignore the overheads ($E_o = t_o = 0$), it results from Eq. (4.4) that the cost to be minimized is $\sum_{i=1}^{n} Es_i^{leak}$, which is the total leakage energy consumed during task execution.

Assuming that the execution interval of task τ_i is divided into $q_i - 1$ sub-intervals, the leakage energy consumption of τ_i is the sum of the leakage energy of all sub-intervals:

$$Es_i^{leak} = \sum_{j=1}^{q_i-1} (P_{ij}^{leak}(V_{ij}, \frac{T_{ij}^{die} + T_{i(j+1)}^{die}}{2}) \cdot t_{ij}^{sub}) \tag{4.20}$$

where T_{ij}^{die}, $T_{i(j+1)}^{die}$ and t_{ij}^{sub} represent the processor SSDTC temperatures at the beginning and end of the j^{th} sub-interval and the length of this sub-interval, respectively. The model in Eq. (2.2) is used to compute the leakage power, P_{ij}^{leak}, in each sub-interval.

Let us first assume that the chip (as well as the heat spreader) temperature at the termination of each task is known and is independent of the starting temperature of the task. Under this assumption, we can formulate our SITDNOH problem as shown in Eq. (4.21)–Eq. (4.33), where the objective function to be minimized is the total leakage energy for all tasks $\sum_{i=1}^{n} Es_i^{leak}$. The optimization variables to be calculated are the idle slot lengths t_i, $(\forall i, 1 \le i \le n)$. Eq. (4.23) requires the sum of all idle slots lengths to be equal to the total available idle time: $dl_n - \sum_{i=1}^{n} te_i^W$. Eq. (4.24) guarantees that the deadline of each task is satisfied. The processor and heat spreader temperatures at the end of task τ_i, $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$, are considered known and assigned by Eq. (4.25) and Eq. (4.26), respectively, where Tg_i^{die} and Tg_i^{sp} are given constants. T_{ij}^{die} and $T_{i(j+1)}^{die}$ are the processor temperature at the beginning and end of j^{th} sub-interval in the execution of task τ_i, and are given by Eq. (4.27) similar to Eq. (4.15) and Eq. (4.17) in Section 4.3.2. Eq. (4.28) describes the same relationship for the heat spreader temperature. $T_{(i+1)1}^{die}$ and $T_{(i+1)1}^{sp}$ are the processor and heat spreader temperatures at the start of task τ_{i+1}, and are dependent

Formulation 1 SITD with No Overheads Consideration

Minimize:

$$\sum_{i=1}^{n} Es_i^{leak} = \sum_{i=1}^{n}\Big(\sum_{j=1}^{q_i-1}\big(t_{ij}^{sub} \cdot P_{ij}^{leak}(V_{ij}, \frac{T_{ij}^{die} + T_{i(j+1)}^{die}}{2})\big)\Big) \quad (4.21)$$

Subject to:

$$t_i \geq 0 \; (1 \leq i \leq n) \quad (4.22)$$

$$\sum_{i=1}^{n} t_i = dl_n - \sum_{i=1}^{n} te_i^W \quad (4.23)$$

$$dl_i \geq \sum_{j=1}^{i-1} t_j + \sum_{j=1}^{i} te_j^W (1 \leq i \leq n) \quad (4.24)$$

$$T_{iq_i}^{die} = Tg_i^{die}(1 \leq i \leq n) \quad (4.25)$$

$$T_{iq_i}^{sp} = Tg_i^{sp}(1 \leq i \leq n) \quad (4.26)$$

$$T_{i(j+1)}^{die} = a1_{ij} \cdot T_{ij}^{die} + b1_{ij} \cdot T_{ij}^{sp} + c1_{ij} \quad (4.27)$$

$$T_{i(j+1)}^{sp} = a2_{ij} \cdot T_{ij}^{die} + b2_{ij} \cdot T_{ij}^{sp} + c2_{ij} \quad (4.28)$$

$$(1 \leq i \leq n; 1 \leq j \leq q_i - 2)$$

$$T_{(i+1)1}^{die} \geq TIs + (T_{iq_i}^{die} - TIs) \cdot e^{(\frac{-t_i}{R_g \cdot C_1})} \; (1 \leq i \leq n-1) \quad (4.29)$$

$$T_{(i+1)1}^{sp} = T_{iq_i}^{sp} \; (1 \leq i \leq n-1) \quad (4.30)$$

$$T_{11}^{die} \geq TIs + (T_{nq_n}^{die} - TIs) \cdot e^{(\frac{-t_n}{R_g \cdot C_1})} \quad (4.31)$$

$$T_{11}^{sp} = T_{nq_n}^{sp} \quad (4.32)$$

$$TIs = P_{idle} \cdot R_g \quad (4.33)$$

on the finishing temperature of the previous task τ_i and the idle slot t_i placed after τ_i. If we assume that idle slots t_i are significantly shorter than the RC time of the heat spreader, then we can describe the processor temperature behavior during the idle slot i by Eq. (4.29) and Eq. (4.31), based on the 1-RC thermal circuit described in Section 4.3.1. TIs is the steady state temperature that the processor would reach if P_{idle} would be consumed for a sufficiently long time and is calculated according to Eq. (4.33). R_g is the sum of the two thermal resistances R_1 and R_2 in Fig. 4.3b. Under the same assumption as above, the heat spreader temperature stays constant during the idle slot as shown in Eq. (4.30) and Eq. (4.32)[2]. Eq. (4.29) and Eq. (4.30) calculate the processor and heat spreader temperature at the end of the idle slot following task τ_i and, implicitly, the starting temperature of τ_{i+1}. Eq. (4.31) and Eq. (4.32) compute the temperature at the start of task τ_1, taking into consideration that this task starts after the idle period following task τ_n (the task set is executed periodically). The presented formulation is a convex non-linear problem, and can be solved efficiently in polynomial time [66].

SITDNOH Approach.

The above formulation is based on the particular assumption that the temperature at the end of a task τ_i is known and fixed. However, in reality, this is not the case, and the temperature $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$ (Eq. (4.25) and Eq. (4.26)) at the termination of a task depend on the starting temperature of the task and, implicitly, on the distribution of the idle time. This makes the above formulation become a non-convex programming problem which is very time consuming to solve. In order to solve the problem efficiently we have developed an iterative heuristic outlined in Fig. 4.5.

The heuristic starts with an arbitrary initial ITD, for example, that the entire idle time t_{tot} is placed after the last task τ_n. Assuming this ITD and the given voltage levels, steady state dynamic temperature analysis is performed, as described in Section 4.3.2. Given the obtained SSDTC, the total leakage energy consumption $\sum_{i=1}^{n} E_i^{leak}$ corresponding to the assumed ITD is calculated. From the SSDTC we can also extract the final temperature $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$ for each task τ_i. Assuming this $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$ as the final temperature in Eq. (4.25) and Eq. (4.26), we can calculate the idle time t_i using the convex optimization formulated from Eq. (4.21) to Eq. (4.33).

From the new ITD resulted after the optimization, we calculate a new SSDTC which provides new temperatures $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$ at the end of each task τ_i. The new total leakage energy consumption $\sum_{i=1}^{n} E_i^{leak}$, corresponding to the updated ITD, is also calculated. The process is continued assuming the new end temperatures in Eq. (4.25) and Eq. (4.26) and the convex optimization produces a new ITD.

[2]Idle periods are supposed to be short. If, exceptionally, they are not significantly shorter than the heat spreader RC time, we use the 2-RC circuit to model the temperature during the idle period in Eq. (4.29) and Eq. (4.32). This will not affect the convexity of the formulation.

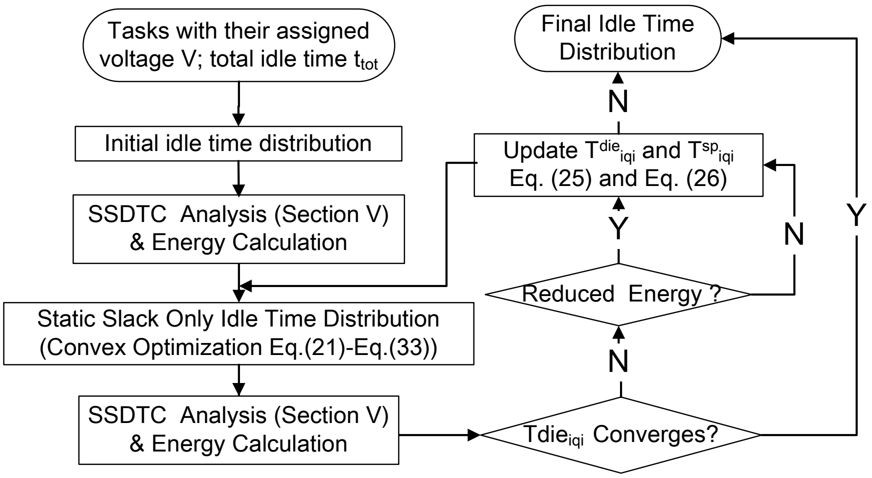

Tasks with their assigned voltage V; total idle time t_tot
Initial idle time distribution
SSDTC Analysis (Section V) & Energy Calculation
Static Slack Only Idle Time Distribution (Convex Optimization Eq.(21)-Eq.(33))
SSDTC Analysis (Section V) & Energy Calculation
Tdie_iqi Converges?
N
Reduced Energy ?
Y
N
Update T^die_iqi and T^sp_iqi Eq. (25) and Eq. (26)
N
Final Idle Time Distribution
Y

Figure 4.5: SITDNOH Heuristic

The iterations outlined above stop when the temperature $T_{iq_i}^{die}$ converges (i.e. $|T_{iq_i}^{die^{new}} - T_{iq_i}^{die^{old}}| < \varepsilon$, $1 \leq i \leq n$). However, it can happen that, after a certain point, additional iterations do not significantly improve the ITD. Therefore, even if convergence has not yet been reached, the optimization is stopped if no significant energy reduction has been achieved: $(E_{tot}^{old} - E_{tot}^{new})/E_{tot}^{old} < \varepsilon'$. Our experiments have shown that maximum 5 iterations are needed with $\varepsilon = 0.5°$ and $\varepsilon' = 0.001$.

4.4.2 SITD with Overhead (SITDOH)

The approach presented in Section 4.4.1 is based on the assumption that time and energy overheads for switching the processor to and from the idle state, t_o and E_o, are zero, which is not the case in reality. If we consider the hypothetical case that the end temperature of each task is known, the problem can be formulated similar to Eq. (4.21)–Eq. (4.33), with the main difference that the total energy to be minimized is given in Eq. (4.4). Based on this formulation, we could solve the SITDOH problem for the real case, when the end temperatures are not supposed to be known, similarly with the approach described in Fig. 4.5. However, the formulation with the objective function Eq. (4.4), due to the binary variable x_i, is a mixed integer convex programing problem which is very time consuming to solve. We, hence, propose an SITDOH heuristic based on the SITDNOH approach presented in Section 4.4.1.

Our SITDOH heuristic comprises two steps. In the first step an optimization of the idle time distribution is performed by eliminating idle intervals whose lengths are smaller than a certain threshold limit. In the second step, the ITD is further refined in order to improve energy efficiency.

A lower bound t^{min} on the length t_i of an idle slot can be determined by considering the following two bounds:

- No idle slot is allowed to be shorter than t_o, the total time needed to switch to/from the idle state.
- The energy overhead due to switching should be compensated by the gain due to putting the processor into the idle state. The energy gain for an idle interval t_i is computed as:

$$E_g = \int_0^{t_i} P_i^{leak}(V_i, T_i(t))\,\mathrm{d}t - P_{idle} \cdot t_i \tag{4.34}$$

  where $T_i(t)$ is the processor temperature as a function of time during the idle time interval [0, t_i]. V_i is the supply voltage for task τ_i. $P_i^{leak}(V_i, T_i(t))$ is the leakage power consumption in the active state during the idle time interval of [0, t_i]. Thus, in order for the overhead to be compensated, we need $E_o < E_g$. As P_i^{leak} depends on the temperature, the threshold length of an idle slot is not a given constant. Nevertheless, this length will be always larger than $E_o/(P_{max_i}^{leak} - P_{idle})$, where $P_{max_i}^{leak} = P_i^{leak}(V_i, T_{max})$ is the leakage power at the maximum temperature T_{max} at which the processor is allowed to run.

In conclusion, for the first step of the SITDOH heuristic, illustrated in Fig. 4.6a, we consider: $t^{min} = max(t_o, E_o/(P_{max_i}^{leak} - P_{idle}))$.

The basic idea of the first step is that no idle slot is allowed to be shorter than t^{min}. Thus, after running SITDNOH, the obtained ITD is checked slot by slot. If a slot length t_i is smaller than t^{min}, this slot will be removed. In order to achieve this, the particular constraint in Eq. (4.22), corresponding to slot i, is changed from $t_i \geq 0$ to $t_i = 0$. After all slots have been visited and Eq. (4.22) updated, SITDNOH is performed again. The obtained ITD is such that all slots which in the previous iteration have been found shorter than t^{min} have disappeared and the corresponding idle time has been redistributed among other tasks. The process is repeated until no slot shorter than t^{min} has been identified.

After step1, we still can be left with slots that are too short to be energy efficient. There are two reasons for this:

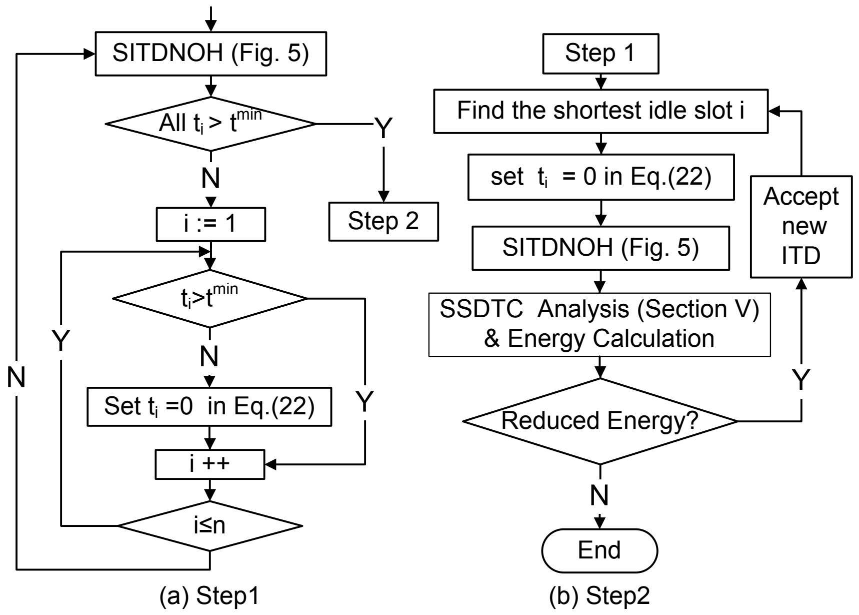


Figure 4.6: SITDOH Heuristic

1. Due to the fact that the processor is running at a temperature lower than the maximum allowed T_{max}, it can happen that the real t^{min} is smaller than the one considered in step1.

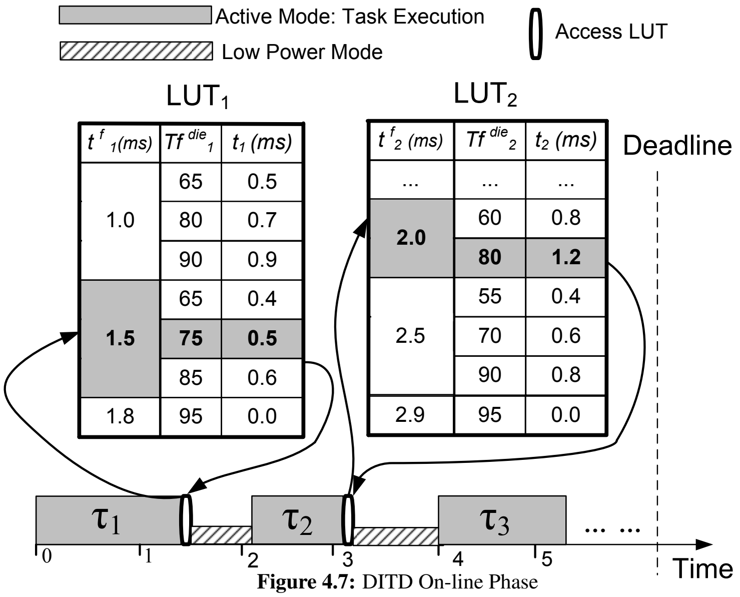
2. Even if $E_o < E_g$, which means that an energy reduction due to the idle slot is obtained, energy efficiency can, possibly, be improved by eliminating the slot and distributing the corresponding idle time among other slots.

In the second step (Fig. 4.6b), we start from the shortest idle slot and consider to eliminate it (by setting the corresponding constraint $t_i = 0$ in Eq. (4.22)). If the ITD obtained after applying SITDNOH is more energy efficient, the new ITD is accepted. The process is continued as long as, by eliminating a slot, the total energy consumption is reduced.

4.5 ITD with Dynamic and Static Slack (DITD)

The above SITD approach determines idle time settings assuming that tasks always execute their WNC. However, due to execution time variations, large amounts of dynamic slack are created at run-time. In order to exploit the dynamic slack, the slot length t_i has to be determined at run-time based on the values of the current time and temperature after termination of task τ_i. In principle, calculating the appropriate t_i implies the execution of a temperature-aware ITD algorithm similar to the one described in Section 4.4.2 (with the optimization objective function and constraints shown by Eq. (4.7) to Eq. (4.9)). Running this algorithm on-line, after execution of each task, implies a time and energy overhead which is not acceptable.

To overcome the above problem, we have divided our DITD approach into an off-line and an on-line phase. In the off-line phase, idle time settings for all tasks are pre-computed, based on possible finishing times and finishing temperatures of the task. The results are stored in look-up tables (LUTs), one for each task. In Fig. 4.7, we show two such tables. They contain idle time settings for combinations of possible termination times t_i^f and finishing temperatures Tf_i^{die} of a task τ_i.



t^f_1(ms)	Tf^{die}_1	t_1 (ms)
1.0	65	0.5
	80	0.7
	90	0.9
1.5	65	0.4
	75	**0.5**
	85	0.6
1.8	95	0.0

t^f_2 (ms)	Tf^{die}_2	t_2 (ms)
...	...	...
2.0	60	0.8
	80	**1.2**
2.5	55	0.4
	70	0.6
	90	0.8
2.9	95	0.0

Figure 4.7: DITD On-line Phase

4.5.1 On-line Phase

The on-line phase is illustrated in Fig. 4.7. Each time a task τ_i terminates, the length of the idle time slot t_i following the termination of τ_i has to be fixed; the on-line scheme chooses the appropriate setting from the lookup table LUT$_i$, depending on the actual time and temperature sensor reading. If there is no exact entry in

LUT$_i$, corresponding to the actual time/temperature, the entry corresponding to the immediately higher time and closest temperature value is selected. For example, in Fig. 4.7, τ_1 finishes at time 1.35ms with a temperature 78°C. To determine the appropriate idle time slot length t_1, LUT$_1$ is accessed. As there is no exact entry with $t_1^f = 1.35$ms and $Tf_1^{die} = 78$°C, the entry corresponding to termination time 1.5ms (1.5ms is immediately higher than 1.35ms) and temperature 70°C (as it is the closest one to $Tf^{die} = 78$°C) is chosen. Hence, the processor will be switched to the idle state for 0.5ms before the next task, τ_2, starts. This on-line phase is of very low, constant time complexity O(1) and, thus, very efficient.

We should notice that, according to our temperature model presented in Section 4.3, the state of the system is defined by both the die and the heat spreader temperatures. In our LUTs, however, we only consider the die temperature for taking the decision on the idle slack. This is due to the following reasons:

1. It is both impractical and potentially expensive to obtain, at run-time, temperature readings from the heat spreader.

2. The variations of the heat spreader temperature are small compared to those of the chip. This is due to the fact that the heat capacitance of the heat spreader is much larger than that of the chip.

3. Considering also the heat spreader temperature as an additional dimension in the LUTs would dramatically increase the size of the tables without significant contribution to energy efficiency.

Thus, when generating the LUTs, we will consider that, at the termination of a task τ_i, the heat spreader has a certain expected temperature Tf_i^{sp}. In Section 4.5.5 we will show how Tf_i^{sp} is calculated.

4.5.2 Off-line Phase

In the off-line phase, one LUT table is generated for each task. The LUT table generation algorithm is illustrated in Fig. 4.8. The outermost loop iterates over the set of tasks and successively constructs the table LUT$_i$ for each task τ_i. The next loop generates LUT$_i$ entries corresponding to the various possible finishing temperatures Tf_i^{die} of τ_i. Finally, the innermost loop iterates, for each possible finishing temperature, over all considered termination times t_i^f of task τ_i.

The algorithm starts by computing the earliest EFT_i and latest possible finishing times LFT_i, as well as the lowest Tf_i^l and highest possible finishing temperature Tf_i^h for each task τ_i. With a given finishing time t_i^f and finishing temperature Tf_i^{die} of task τ_i, the innermost loop performs the slack distribution step DITDOH,

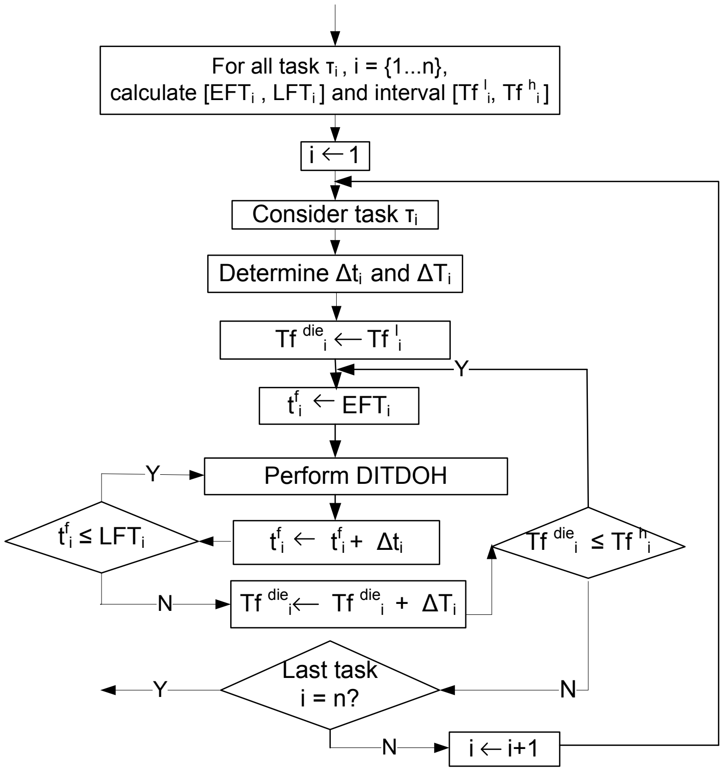

For all task τ_i, i = {1...n}, calculate [EFT_i, LFT_i] and interval [Tf^l_i, Tf^h_i]
$i \leftarrow 1$
Consider task τ_i
Determine Δt_i and ΔT_i
$Tf^{die}_i \leftarrow Tf^l_i$
$t^f_i \leftarrow EFT_i$
Perform DITDOH
$t^f_i \leftarrow t^f_i + \Delta t_i$
$t^f_i \leq LFT_i$
Y
N
$Tf^{die}_i \leftarrow Tf^{die}_i + \Delta T_i$
$Tf^{die}_i \leq Tf^h_i$
Y
N
Last task i = n?
Y
N
$i \leftarrow i+1$

Figure 4.8: DITD Off-line Phase

iteratively. We describe the DITDOH algorithm in detail in Section 4.5.3. For successive iterations, the finishing temperature Tf_i^{die} and time t_i^f will be increased with the time and temperature quanta $\triangle t_i$ and $\triangle T_i$, respectively. The calculation of the parameters EFT_i, LFT_i, Tf_i^l and Tf_i^h as well as the determination of the granularities and number of entries along the time and temperature dimensions are presented in Section 4.5.4 and Section 4.5.5, respectively.

4.5.3 DITDOH Algorithm

When calculating the actual LUT entries for a task τ_i, the ITD algorithm DITDOH is performed to determine the idle slot length t_i following the termination of τ_i, with the given termination time and temperature, based on the problem formulation described in Section 4.2.2. DITDOH is similar to SITDOH outlined in Section 4.4.2. However, unlike the formulation used in SITDOH (Eq. (4.21)–Eq. (4.33)) which is based on SSDTC estimation, the formulation used for DITDOH is based on the estimation of a transient temperature curve (TTC) described in Section 4.3.3. Since we do not rely on the fact that successive iterations of the application are identical and that tasks execute always with their worst case number of cycles, we do not calculate an SSDTC corresponding to the dynamic steady state. But, instead, we estimate a TTC.

The formulation used for DITDOH is shown in Eq. 4.35–Eq. (4.49). As mentioned in Section 4.2.2, the energy is optimized for the case that the future tasks τ_{i+1} to τ_n execute their expected time te^E which, in reality, happens with a much higher probability than, e.g., the te^W (nevertheless, idle time slots are distributed such that, even in the worst case, deadlines are satisfied). The objective function, Eq. (4.35), to be minimized is the total leakage energy of further tasks to be executed in the current iteration: τ_k, $(\forall k, i < k \leq n)$. Eq. (4.35) is similar to Eq. 4.21 with two differences:

1. It refers only to the remaining tasks $\tau_{i+1}, \ldots, \tau_n$.
2. The execution interval of a task τ_k, which is divided into $q_k - 1$ subintervals, is not corresponding to the worst case te_k^W, but to the expected case te_k^E.

The optimization variables to be calculated are the idle slot lengths t_k, $(\forall k, i \leq k \leq n)$. Eq. (4.37) requires that the sum of all idle slot lengths should be equal to the total available idle time, where t_i^f is the current task's finishing time. The total available idle time is calculated based on the assumption that all future tasks are executed with their expected workload.

Eq. (4.38) guarantees the deadline of task τ_{i+1}—the next task to be executed after the termination of the current task τ_i in the worst case (task τ_{i+1} executed

Formulation 2 DITD with No Overheads Consideration

Minimize:

$$\sum_{k=i+1}^{n} E_k^{leak} = \sum_{k=i+1}^{n} (\sum_{j=1}^{q_k-1} (t_{kj}^{sub} \cdot P_{kj}^{leak}(V_{kj}, \frac{T_{kj}^{die} + T_{k(j+1)}^{die}}{2}))) \quad (4.35)$$

Subject to:

$$t_j \geq 0 \; (i \leq j \leq n) \quad (4.36)$$

$$\sum_{k=i}^{n} t_k = dl_n - t_i^f - \sum_{k=i+1}^{n} te_j^E \quad (4.37)$$

$$dl_{i+1} \geq t_i^f + t_i + te_{i+1}^W \quad (4.38)$$

$$LFT_{i+1} \geq t_i^f + t_i + te_{i+1}^W \quad (4.39)$$

$$dl_j \geq t_i^f + \sum_{k=i}^{j-1} t_k + te_{i+1}^W + \sum_{k=i+2}^{j} te_k^E \quad (4.40)$$

$$(i+2 \leq j \leq n)$$

$$T_{kq_k}^{die} = Tg_k^{die} \; (i+1 \leq k \leq n) \quad (4.41)$$

$$T_{kq_k}^{sp} = Tg_k^{sp} \; (i+1 \leq k \leq n) \quad (4.42)$$

$$T_{(i+1)1}^{die} \geq TIs + (Tf_i^{die} - TIs) \cdot e^{(\frac{-t_i}{R_g \cdot C_1})} \quad (4.43)$$

$$T_{(i+1)1}^{sp} = Tf_i^{sp} \quad (4.44)$$

$$T_{k(j+1)}^{die} = a1_{kj} \cdot T_{kj}^{die} + b1_{kj} \cdot T_{kj}^{sp} + c1_{kj} \quad (4.45)$$

$$T_{k(j+1)}^{sp} = a2_{kj} \cdot T_{kj}^{die} + b2_{kj} \cdot T_{kj}^{sp} + c2_{kj} \quad (4.46)$$

$$(i+1 \leq k \leq n; 1 \leq j \leq q_k - 2)$$

$$T_{(k+1)1}^{die} \geq TIs + (T_{kq_k}^{die} - TIs) \cdot e^{(\frac{-t_k}{R_g \cdot C_1})} \quad (4.47)$$

$$(i+1 \leq k \leq n-1)$$

$$T_{(k+1)1}^{sp} = T_{kq_i}^{sp} \; (i \leq k \leq n-1) \quad (4.48)$$

$$TIs = P_{idle} \cdot R_g \quad (4.49)$$

with te^W_{i+1}). In order to guarantee that all future tasks meet their deadlines in the worst case, Eq. (4.39) requires that τ_{i+1} finishes before LFT_{i+1}, in the worst case. The latest finishing time LFT_{i+1} (see Section 4.5.4) is the latest termination time of task τ_{i+1} that still allows future tasks, following τ_{i+1}, to satisfy their deadline even if their worst case workloads are executed. Thus Eq. (4.38) and Eq. (4.39) guarantee not only that the deadline of τ_{i+1} is satisfied in the worst case but also that τ_{i+1} finishes in time for all the remaining tasks to be able to meet their deadline in the worst case. Eq. (4.40) enforces the deadline of the remaining tasks τ_j, $(\forall j, i+2 \leq j \leq n)$, considering that they execute their expected workload. This means that the idle time t_i following task τ_i is determined such that it guarantees deadlines to be satisfied in the worst case but is optimized for the situation that tasks execute their expected workloads.

Similar to Eq. (4.25) and Eq. (4.26), Eq. (4.41) and Eq. (4.42) specify the processor and heat spreader temperatures at the finishing of task τ_k: $T^{die}_{kq_k}$ and $T^{sp}_{kq_k}$. Eq. (4.43) computes the processor temperature at the beginning of task τ_{i+1} similar to Eq. (4.31), where Tf^{die}_i is the chip temperature at the termination of the current task τ_i. Similarly, Eq. (4.44) computes the heat spreader temperature at the beginning of task τ_{i+1}, where Tf^{sp}_i is, as described in Section 4.5.1, the expected heat spreader temperature at the termination of task τ_i. Tf^{sp}_i is pre-calculated as will be explained in Section 4.5.5. Eq. (4.45)–Eq. (4.48) compute the TTC of processor/heat spreader based on our TTC estimation method described in Section 4.3.3, where T^{die}_{kj} and $T^{die}_{k(j+1)}$ are the processor temperature at the beginning and end of the j^{th} sub-interval during the execution of task τ_k. The above formulation is a convex non-linear problem and can be solved efficiently in polynomial time [66].

Coming back to the DITD off-line phase in Fig. 4.8, the DITDOH algorithm is invoked for each line in the LUT$_i$ corresponding to a task τ_i. This invocation will result in the calculation of the slack length t_i corresponding to the current value of termination time t^f_i and temperature Tf^{die}_i. DITDOH performs exactly like SITDOH (Fig. 4.6), with the exception that for solving SITDNOH (Fig. 4.5), instead of Eq. (4.21)–Eq. (4.33), the formulation in Eq. (4.35)–Eq. (4.49) is used.

4.5.4 Time Bounds and Granularity

In the first step of the algorithm in Fig. 4.8, the EFT_i and LFT_i for each task are calculated. The earliest finishing time EFT_i is calculated based on the situation that all tasks execute their best case execution time te^B_i. The latest finishing time LFT_i is calculated as the latest termination time of τ_i that still allows all tasks $\tau_j, j > i$ to satisfy their deadlines when they execute their worst case execution time te^W_i. With the time interval $[EFT_i, LFT_i]$ for task τ_i, a straightforward approach to determine the number of entries along the time dimension would be to allocate the same number of entries for each task. However, the time interval sizes $LFT_i - EFT_i$ can differ very much among tasks, which should be taken into consideration when deciding on the number of time entries Nt_i. Therefore, given a total number of entries along the time dimension NL_t, we determine the number of time entries in each LUT$_i$, as follows:

$$Nt_i = \left\lceil NL_t \cdot \frac{(LFT_i - EFT_i)}{\sum\limits_{i=1}^{n} (LFT_i - EFT_i)} \right\rceil$$

4.5.5 Temperature Bounds and Granularity

The granularity $\triangle T_i$ along the temperature dimension is the same for all task τ_i and has been determined experimentally. Our experiments have shown that values around 15° are appropriate, in the sense that finer granularities will only marginally improve energy efficiency.

To determine the number of entries along the temperature dimension, we need to calculate the temperature interval $[Tf_i^l, Tf_i^h]$ at the termination of each task. In fact, it is not needed to determine the bounds of the temperature interval exactly[3]. A good estimation, such that, at run-time, temperature readings outside the determined interval will happen rarely, is sufficient. If the temperature readings exceed the upper/lower bound of the interval, the idle time setting corresponding to the highest/lowest temperature value available in the LUT will be used. One alternative would be to simply assume that all tasks have a finishing temperature interval $[T_a, T_{max}]$, where T_a is the ambient temperature and T_{max} is the maximum temperature at which the chip is allowed to work. This would lead to huge amounts of wasted memory space (for storing LUT tables) as well as wasted computation time in the off-line phase. We have developed an estimation technique for the temperature interval $[Tf_i^l, Tf_i^h]$, which balances computation complexity and accuracy of the results.

In order to estimate the temperature bounds Tf_i^l and Tf_i^h, we define two run-time scenarios:

- *Worst case execution scenario*: in which the actual execution time of each task τ_i is always equal to its worst case execution time: $te_i^A = te_i^W$.

[3]This is different from the situation in Chapter 3 (Section 3.4.3). The LUTs generated there have to guarantee that frequencies are assigned in a safe way.

- *Best case execution scenario*: in which the actual execution time of each task τ_i is always equal to its best case execution time: $te_i^A = te_i^B$.

In both scenarios, the processor will execute the corresponding periodic power pattern repeatedly and the processor temperature will eventually reach the corresponding steady state dynamic temperature curve (denoted as $SSDTC^w$ for the worst case scenario and $SSDTC^b$ for the best case scenario, respectively). From the corresponding SSDTC, we can obtain, for each task τ_i, its finishing temperature. We use the finishing temperature of task τ_i corresponding to the *worst case execution scenario*, Tf_i^w, as the upper bound of the finishing temperature of task τ_i: $Tf_i^h = Tf_i^w$; the finishing temperature of task τ_i corresponding to the *best case execution scenario*, Tf_i^b, will be used as the lower bound: $Tf_i^l = Tf_i^b$.

In order to obtain the $SSDTC^w$ we first perform the SITDOH heuristic (Fig. 4.6). Then, the temperature analysis (Section 4.3) produces the temperature curve for the worst case scenario with the corresponding idle time distribution generated by SITDOH. The $SSDTC^b$ curve is obtained in a similar way, by replacing te_i^W with te_i^B in the constraint in Eq. (4.23).

With the upper and lower bounds Tf_i^l and Tf_i^h obtained for each task, the number of the entries along the temperature dimension, for task τ_i, is:

$$NT_i = \left\lceil \frac{Tf_i^h - Tf_i^l}{\triangle T_i} \right\rceil$$

where $\triangle T_i$ is the granularity along the temperature dimension. The considered temperature values in LUT$_i$ are then determined as follows:

$$Tf_i^{die} = k_i \cdot \triangle T_i + Tf_i^l$$

where the integer k_i is in the interval: $0 \leq k_i \leq NT_i$.

As mentioned in Section 4.5.1, when generating the LUTs, we consider that, at the termination of a task τ_i, the heat spreader has a certain expected temperature Tf_i^{sp}. In order to obtain these temperatures, we perform the same procedure as outlined above but, in this case, considering the expected execution time of each task: $te_i^A = Te_i^E$. We obtain the temperature curve $SSDTC_{sp}^{exp}$ corresponding to the heat spreader (see Section 4.3.2), from which we extract the expected temperature of the heat spreader, Tf_i^{sp}, at the termination of each task τ_i.

4.6 Experimental Results

4.6.1 Evaluation of The Thermal Model

Experimental Setup.
We have evaluated our thermal model considering platforms with parameter settings based on values from [67], [68] and [69]. We consider die areas of 6×6, 8×8 and $10 \times 10\text{mm}^2$. The heat spreader area is five times the die area, and the heat sink area is between 1.3 and 1.4 times the area of the heat spreader. The thickness of the die and heat spreader are 0.5mm and 2mm respectively. The thickness of the heat sink is between 10mm and 20mm. The coefficients corresponding to the power model (see Chapter 2) are based on [55] and [59]. For the temperature calculation we have considered a piecewise linear leakage model with 3 segments (see Chapter 2), as recommended in [21].
Accuracy.
We first performed a set of experiments to evaluate the accuracy of our temperature analysis approach proposed in Section 4.3. We randomly generated 500 periodic voltage patterns corresponding to applications with periods in the range between 5ms and 100ms. For each application, considering the coefficients and platform parameters outlined above, we have computed the SSDTC using the approach proposed in Section 4.3.2 and by using Hotspot simulation. For each pair of temperature curves obtained, we calculated the maximum deviation as the largest temperature difference between any corresponding pairs of points (in absolute value), as well as the average deviation. Fig. 4.9 illustrates the results for different application periods. For applications with a period of 50ms, for example, there is no single case with a maximum deviation larger than 2.1°C, and the average deviation is 0.8°C. Over all 500 applications, the average and maximum deviation are 0.8°C and 3.8°C respectively. We can observe that the deviation increases with the increasing period of the application. This is due to the fact that, with larger periods, accuracy can be slightly affected by neglecting the thermal capacitance of the heat sink (see Section 4.3).
Computation Time.
We have compared the corresponding computation time of our SSDTC generation approach with the time needed by Hotspot. Fig. 4.9 illustrates the average speedup as the ratio of the two execution times. The speedup is between 3000 for periods of 5ms and 20 for 100ms periods. An increasing period leads to a larger linear system that has to be solved for SSDTC estimation (Section 4.3.2), which explains the shape of the speedup curve in Fig. 4.9.

The accuracy and speedup of our approach are also dependent on the length of the sub-interval considered for the temperature analysis (Section 4.3.2 and Fig. 4.4). For the experiments throughout Section 4.6, the length of the sub-interval is 2ms.

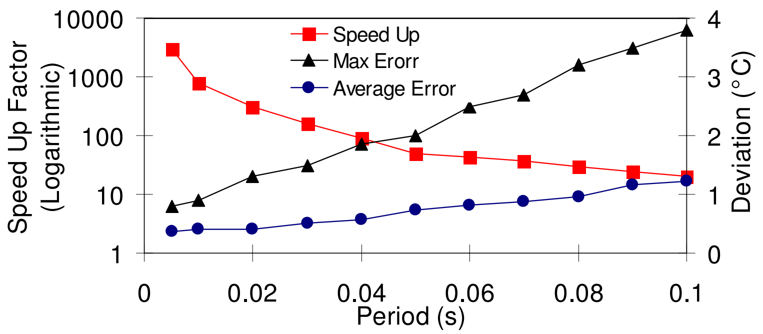


Figure 4.9: SSDTC Estimation with Our Approach vs. Hotspot

This is based on the observation that reducing the length beyond this limit does not improve the accuracy significantly.

4.6.2 Evaluation of ITD Heuristics

We have used both generated test applications as well as a real life example in our experiments to evaluate our DITD approach presented in Section 4.5. This, implicitly, also evaluates the SITD approach (Section 4.4) since:

- For small dynamic slack ratio, the dynamic approach converges towards the static one.
- The DITD approach is based on the SITD for calculation of each entry in the LUTs.

Experimental Setup.
We have randomly generated 100 test applications consisting of 30 to 100 tasks. The workload in the worst case (WNC) for each task is generated randomly in the range $[10^6, 5.0{\times}10^6]$ clock cycles, while the workload in the best case is generated in the range $[10^5, 5.0{\times}10^5]$ clock cycles. To generate the expected workload ENC_i of each task, the following steps are performed:

1. The value of the expected total dynamic idle time, t_d^E, is given as an input: t_d^E is the total dynamic slack when all tasks execute their workload in the expected case: $t_d^E = \sum_{i=1}^{n}(te_i^W - te_i^E)$.
2. t_d^E is divided into a number n_{sub} of sub-intervals with equal length (t_{sub}).

3. The n_{sub} sub-intervals are allocated among all tasks based on a uniform distribution; as result, each task is allocated a number p of sub-intervals.

4. The expected workload ENC_i of task τ_i is, thus, determined as: $ENC_i = WNC_i - p \cdot t_{sub} \cdot f_i$, where f_i is the processor frequency when task τ_i is executed.

In order to evaluate our DITD technique, we have considered a straightforward approach (SFA) for comparison. This SFA scenario corresponds to the natural execution procedure for the case when no idle time distribution is performed. Following this approach, tasks are executed according to a static schedule generated based on the worst case execution time. According to this schedule, the static slack is placed at the end of the application, after the last task. At run-time, when the tasks execute less than their WNC and the generated dynamic slack is large enough, the processor is put in idle mode. More exactly, the SFA works as follows:

1. The start time of each task t_i^{st} is determined off-line by: $t_i^{st} = te_{i-1}^{W} + t_{i-1}^{st}$.

2. At runtime, whenever a task τ_i terminates, we compute the gap $t_g = t_{i+1}^{st} - t_i^f$, where t_i^f is the termination time of the current task.

3. If $t_g = 0$, the next task τ_{i+1} starts immediately after the termination of task τ_i. When $t_g > 0$, if the following two conditions are both satisfied, the processor will be switched to idle state during t_g (otherwise the processor will stay in the same active state with the voltage level at which task τ_i is executed): (a) $t_g > t_o$, where t_o is the time overhead due to power state switching; (b) the energy gain E_g is positive: $E_g = E^a - (P_{idle} \cdot t_g + E_o) > 0$, where E^a is the leakage energy consumption of the processor during t_g if the processor stays in the active state. E^a is estimated as $P^{leak} \cdot t_g$, where P^{leak} is the leakage power consumption calculated at the temperature when task τ_i terminates. $P_{idle} \cdot t_g + E_o$ is the energy consumption if the processor is switched to idle state during t_g, where E_o is the energy overhead due to switching, and P_{idle} is the processor power consumption in idle state.

We have applied both the DITD and SFA approaches on the same test applications. We assume, for each task τ_i, that the actual number of executed workload at run-time conforms to the beta distribution [65]. When we simulate the execution of the test applications, the actual number of executed clock cycles of a task is generated using a random number generator according to the beta distribution Beta($\alpha 1_i, \alpha 2_i$). The parameters $\alpha 1_i$ and $\alpha 2_i$ are determined based on (1) the expected workload ENC_i and (2) a given standard deviation σ_i of the executed clock cycles of task τ_i. The Hotspot system [25] is used to simulate the sensor readings which track the temperature behaviour of the platform during the execution of a test application.

In our experiments, the granularity along the time and temperature dimensions for the LUT tables is set to 1.5–2.0ms and 15°–20°, respectively. It is important to mention that in all our experiments we have accounted for the time and energy overhead imposed by the on-line phase of our DITD. Similarly, we have also taken into consideration the energy overhead due to the memory access. This overhead has been calculated based on the energy values given in [62] and [63]. The energy and time overheads due to power state switching are set to $E_o = 0.5$mJ and $t_o = 0.4$ms, respectively, according to [20].

After applying both the DITD and SFA approaches on a test application, we compute the corresponding leakage energy reduction due to our DITD approach compared to the SFA: $I = (E^{SFA} - E^{DITD})/E^{SFA} \times 100\%$, where E^{SFA} and E^{DITD} are the consumed leakage energy corresponding to the SFA and DITD approach, respectively.

Leakage Energy Reduction vs. Slack Time Ratios.
We first performed experiments considering different combinations of static (r_s) and dynamic idle time ratio (r_d). The static idle time ratio is computed as: $r_s = (dl_n - \sum_{i=1}^{n} te_i^W)/dl_n$, where dl_n is the deadline of the last task in execution order. The dynamic idle time ratio is calculated as: $r_d = t_d^E/dl$, where t_d^E is the total dynamic slack when all tasks execute their workload in the expected case, as described earlier in this section. Fig. 4.10 shows the averaged leakage energy reduction I over all test applications. The energy reduction achieved by DITD grows with the available amount of static and dynamic slack. With $r_s = 0.2$ and $r_d = 0.2$, for example, leakage energy can be reduced with 20% by applying our $DITD$ approach.

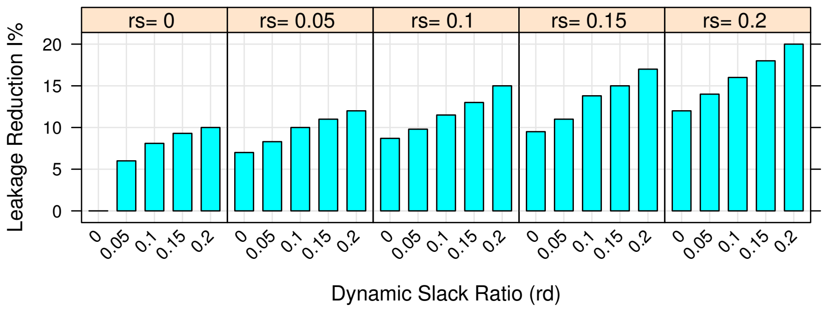


Figure 4.10: Leakage Energy Reduction with Low Switching Overheads

In order to explore the influence of the energy and time overheads on the potential leakage reduction, we have repeated the previous experiments in a context where energy and time overheads are set to higher values: $E_o = 1.0\text{mJ}$ and $t_o = 0.8\text{ms}$. Fig. 4.11 shows the corresponding averaged leakage energy reduction I. The results show a similar trend as in Fig. 4.10. Comparing the results in Fig. 4.10 and Fig. 4.11, we can observe that the leakage reduction achieved with the higher overhead settings is larger. The leakage reduction approaches 40% with $r_s = 0.2$ and $r_d = 0.2$. The reason is the following: with large switching overhead, it is

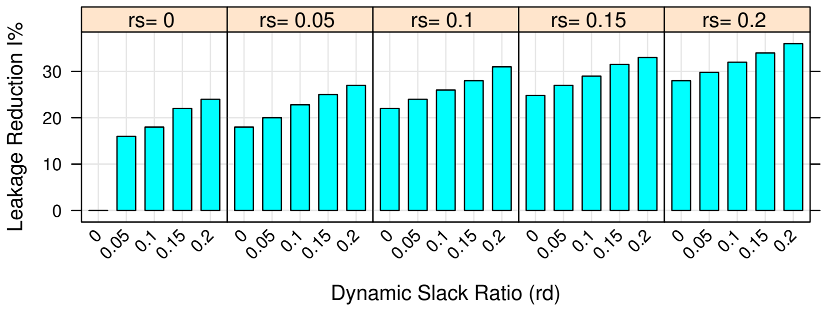


Figure 4.11: Leakage Energy Reduction with High Switching Overheads

more likely that the generated slots are too small for switching power state to be energy efficient. Thus, using the SFA approach, the processor will keep in the active power state. With the DITD approach, however, the slack time will be redistributed such that large slack slots are generated and, even with large overhead, power state switches can be performed.

The DITD approach proposed in Section 4.5 achieves leakage energy reduction due to two main features: (1) it is temperature aware, which means that idle time is distributed such that the temperature is controlled in order to minimize leakage; (2) it redistributes slack such that the number of idle slots which are too short to switch power state, is minimized. A comparison between Fig. 4.10 and Fig. 4.11 illustrates the second feature of our ITD technique. However, the following question still remains open: How much does the temperature awareness of our approach contribute to the energy reduction? In order to answer this question we have repeated the above experiments considering a hypothetical scenario with zero switching overhead: $E_o = 0\text{mJ}$ and $t_o = 0\text{ms}$. The results are shown in Fig. 4.12. Under such a scenario, the processor can be switched to the low power state for the duration of the total idle time (regardless the length of the individual idle slots). Thus, the

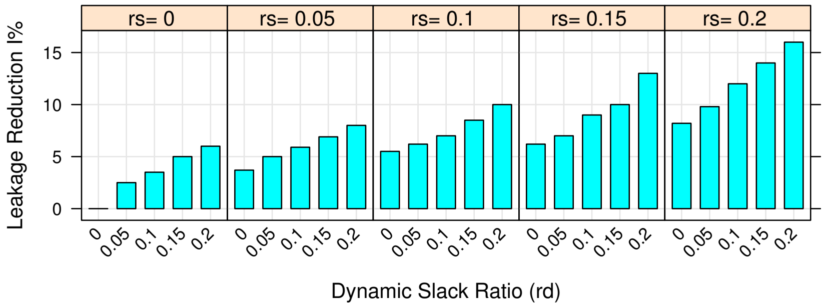


Figure 4.12: Leakage Energy Reduction with No Switching Overheads

energy gains obtained with DITD compared to SFA, as illustrated in Fig. 4.12, are exclusively due to the temperature awareness of the DITD approach.

From Fig. 4.10, Fig. 4.11, and Fig. 4.12 one can also observe the efficiency of the ITD approach with only static slack (SITD, Section 4.4). The cases where $rd = 0$ (no dynamic slack) are, in fact, corresponding to those situations when only static slack is distributed. Obviously, in the cases that both $rs = 0$ and $rd = 0$, there is no slack to distribute and, thus, the energy reduction is zero.

Leakage Energy Reduction vs. Standard Deviation.
As mentioned that, for our experiments we have generated workloads for each task τ_i according to a beta distribution Beta($\alpha 1_i$, $\alpha 2_i$), where $\alpha 1_i$ and $\alpha 2_i$ are determined based on the expected workload ENC_i and standard deviation σ_i of the executed workload. For the above experiments, the standard deviation

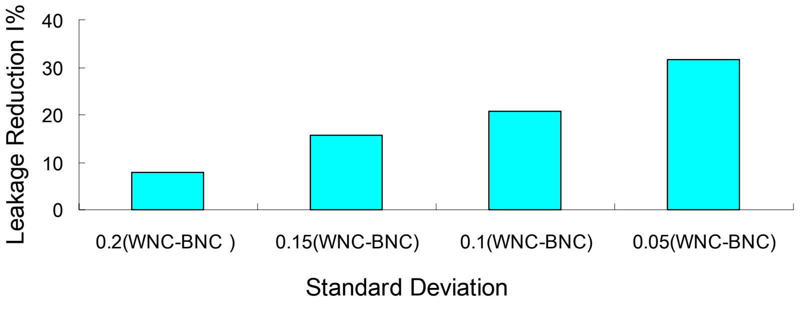


Figure 4.13: Leakage Energy Reduction with Different Standard Deviations

σ_i for each task is considered to be: $\sigma_i = 0.1 \times (WNC_i - BNC_i)$. As the standard deviation has an influence on the potential leakage reduction, we have repeated the above experiments, considering three different settings of σ_i, namely, $0.2 \times (WNC_i - BNC_i)$, $0.15 \times (WNC_i - BNC_i)$, and $0.05 \times (WNC_i - BNC_i)$. Fig. 4.13 shows the leakage reduction $I\%$ by applying our DITD approach relative to the SFA, with different standard deviation settings. We have considered test applications having static and dynamic ratios of: $r_s = 0.2$ and $r_d = 0.2$. As can be observed, the efficiency of the DITD approach increases as the standard deviation decreases. This is due to the fact that our DITD algorithm is targeted towards optimizing the energy consumption for the case that tasks execute the expected number of cycles ENC. When the standard deviation is smaller, more of the actual executed number of clock cycles are clustering around the ENC, and, therefore, our DITD approach can achieve better leakage reduction.

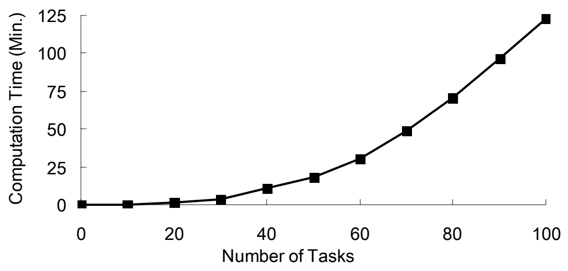


Figure 4.14: Computation Time

Computation Time.
We have also evaluated the computation time for the off-line phase of our DITD approach. The results are shown in Fig. 4.14.
MPEG2 Decoder.
We have applied our DITD approach to a real-life application, namely an MPEG2 decoder, which consists of 34 tasks. Details regarding the application are described in [60]. We have considered a platform with the size of the chip, heat spreader, and heat sink of $8 \times 8\text{mm}^2$, $18 \times 18\text{mm}^2$, and $22 \times 22\text{mm}^2$, respectively. The thickness of the chip, heat spreader, and the heat sink is 0.5mm, 2mm, and 15mm, respectively. The execution time distribution of the tasks has been obtained from simulations on the MPARM platform [70]. We considered the following two overheads settings: (1) $E_o = 0.5$mJ, $t_o = 0.4$ms, (2) and $E_o = 1.0$mJ, $t_o = 0.8$ms. The leakage energy reduction by applying our DITD approach relative to the SFA approach is 32.5% and 40.8%, respectively.

Chapter 5

Conclusions

In this thesis, we have addressed the two following problems: (1) Energy optimization via temperature-aware dynamic voltage/frequency scaling (DVFS), (2) Energy optimization through temperature-aware idle time distribution (ITD).

In Chapter 3 we have proposed temperature-aware DVFS techniques where both leakage/temperature and frequency/temperature dependencies are taken into consideration. We first proposed an off-line temperature-aware DVFS approach which only considers static slack. Based on this off-line DVFS approach, we proposed an on-line temperature-aware DVFS technique which is able to exploit both static and dynamic slack. The on-line DVFS approach consists of two parts: an off-line temperature-aware optimization step and an on-line voltage/frequency setting based on temperature sensor readings. Experiments have demonstrated that significant energy improvements can be achieved using the proposed temperature-aware DVFS techniques.

In Chapter 4, we have addressed the problem of temperature-aware ITD. We started with proposing a static temperature-aware ITD approach for leakage energy optimization where only static slack is considered. We then proposed a dynamic approach considering both static and dynamic idle time, which consists of an off-line and an on-line step. The experiments have demonstrated that considerable energy reduction can be achieved by our temperature-aware ITD approaches. In order to efficiently perform temperature analysis inside our optimization loop, we have also proposed a fast and accurate system-level temperature analysis approach. Experiments show that our temperature analysis method achieves good accuracy with fast speed.

References

[1] P. Marwedel. *Embedded System Design*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] J. Catsoulis. *Designing Embedded Hardware, 2nd Edition*. O'Reilly Media, May. 2005.

[3] W. Wolf and J. Staunstrup. *Hardware/Software CO-Design: Principles and Practice*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[4] M. T. Schmitz, Bashir M. Al-Hashimi, and P. Eles. *System-Level Design Techniques for Energy-Efficient Embedded Systems*. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[5] J. Rabaey. *Low Power Design Essentials*. Springer Publishing Company, Incorporated, 2009.

[6] J. J. Chen and C. F. Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms. In *Proc. 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 28 –38, Aug. 2007.

[7] A. Andrei, P. Eles, and Z. Peng. Energy optimization of multiprocessor systems on chip by voltage selection. *IEEE Transactions on Very Large Scale Integration Systems*, 15:262–275, Mar. 2007.

[8] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proc. International Symposium on Low Power Electronics and Design*, pages 197–202, Aug. 1998.

[9] W. C. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. *ACM Transactions on Embedded Computing Systems*, 4(1):211–230, 2005.

[10] H. Aydi, P. Mejía-Alvarez, D. Mossé, and R. Melhem. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proc. the 22nd IEEE Real-Time Systems Symposium*, pages 95–105, Washington, DC, USA, Dec. 2001. IEEE Computer Society.

[11] A. Andrei, P. Eles, Z. Peng, M. Schmitz, and B. M. Al-Hashimi. Quasi-static voltage scaling for energy minimization with time constraints. In *Proc. Design Automation and Test in Europe*, pages 514–519, Mar. 2005.

[12] C. Xian, Y. H. Lu, and Z.Y. Li. Dynamic voltage scaling for multitasking real-time systems with uncertain execution time. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(8):1467–1478, Aug. 2008.

[13] J. J. Chen, C. Y. Yang, and T. W. Kuo. Slack reclamation for real-time task scheduling over dynamic voltage scaling multiprocessors. In *Proc. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, volume 1, pages 1–8, Jun. 2006.

[14] A. Vassighi and M. Sachdev. *Thermal and Power Management of Integrated Circuits (Series on Integrated Circuits and Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[15] D. Brooks, R. P. Dick, R. Joseph, and L. Shang. Power, thermal, and reliability modeling in nanometer-scale microprocessors. *IEEE Micro*, 27:49–62, 2007.

[16] R. Cobbold. Temperature effects on mos transistors. *Electronic Letters*, 2:190–191, 1966.

[17] J. C. Ku and Y. Ismail. On the scaling of temperature-dependent effects. *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems*, 26(10):1882–1888, Oct. 2007.

[18] International technology roadmap for semiconductors. `http://public.itrs.net`.

[19] W. P. Liao, L. He, and K. M. Lepak. Temperature and supply voltage aware performance and power modeling at micro-architecture level. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(No.7):1042–1053, Jul. 2005.

[20] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for realtime embedded systems. In *Proc. Design automation Conference*, pages 275–280, Jun. 2004.

[21] Y. P. Liu, R. P. Dick, L. Shang, and H. Z. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *Proc. Design Automation Test in Europe Conference*, pages 1–6, apr. 2007.

[22] C. Y. Yang, J. J. Chen, L. Thiele, and T. W. Kuo. Energy-efficient real-time task scheduling with temperature-dependent leakage. In *Proc. Design Automation and Test in Europe*, pages 9–14, Mar. 2010.

[23] H. Huang, G. Quan, and J. Fan. Leakage temperature dependency modeling in system level analysis. In *Proc. 11th International Symposium on Quality Electronic Design*, pages 447–452, Mar. 2010.

[24] F. Kreith. *The CRC Handbook of Thermal Engineering*. CRC Press, Boca Raton, 2000.

[25] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotspot: A compact thermal modeling methodology for early-stage vlsi design. *IEEE Transactions on VLSI Systems*, 14(5):501–513, May 2006.

[26] Y. Yang, Z. P. Gu, R. P. Dick, and L. Shang. Isac: Integrated space and time adaptive chip-package thermal analysis. *IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems*, 26(1):86–99, Jan. 2007.

[27] S. Wang and R. Bettatin. Delay analysis in temperature-constrained hard real-time systems with general task arrivals. In *Proc. Real-Time Systems Symposium*, pages 323–334, 2006.

[28] R. Jayaseelan and T. Mitra. Temperature aware task sequencing and voltage scaling. In *Proc. International Conference on Computer Aided Design*, pages 618–623, 2008.

[29] S. Zhang and K. S. Chatha. System-level thermal aware design of applications with uncertain execution time. In *Proc. International Conference on Computer-Aided Design*, pages 242–249, Nov. 2008.

[30] R. Rao and S. Vrudhula. Fast and accurate prediction of the steady-state throughput of multicore processors under thermal constraints. *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems*, 28(10):1559–1572, Oct. 2009.

[31] M. Sasaki, M. Ikeda, and K. Asada. -1/+0.8°c error, accurate temperature sensor using 90nm 1v cmos for on-line thermal monitoring of vlsi circuits. *IEEE Transactions on Semiconductor Manufacturing*, 21:201–208, 2008.

[32] A. K. Coskun, T. S. Rosing, and K. Whisnant. Temperature aware task scheduling in mpsocs. In *Proc. Design Automation Test in Europe*, pages 1–6, apr. 2007.

[33] M. Rajarshi and M. S. Ogrenci. Systematic temperature sensor allocation and placement for microprocessors. In *Proc. Design Automation Conference*, pages 542–547, Jul. 2006.

[34] A. N. Nowroz, R.Cochran, and S. Reda. Thermal monitoring of real processors: Techniques for sensor allocation and full characterization. In *Proc. Design Automation Conference*, pages 56–61, Jun. 2010.

[35] R. Cochran and S. Reda. Spectral techniques for high-resolution thermal characterization with limited sensor data. In *Proc. Design Automation Conference*, pages 478–483, Jul. 2009.

[36] S. Sharifi, C. C. Liu, and T. S. Rosing. Accurate temperature estimation for efficient thermal management. In *Proc. International Symposium on Quality Electronic Design*, pages 137–142, Mar. 2008.

[37] Y. F. Zhang and A. Srivastava. Adaptive and autonomous thermal tracking for high performance computing systems. In *Proc. Design Automation Conference*, pages 68–73, Jun. 2010.

[38] B. Nikhil, K. Tracy, and P. Kirk. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1):1–39, 2007.

[39] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. In *Proc. Design Automation and Test in Europe*, pages 288–293, Mar. 2008.

[40] Y. Ge, P. Malani, and Q. Qiu. Distributed task migration for thermal management in many-core systems. In *Proc. Design Automation Conference*, pages 579–584, Jun. 2010.

[41] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *Proc. International Conference on Computer Aided Design*, pages 281–288, Nov. 2007.

[42] S. Zhang and K. S. Chatha. Thermal aware task sequencing on embedded processors. In *Proc. Design Automation Conference*, pages 585–590, Jun. 2010.

[43] R. Rao and S. Vrudhula. Efficient online computation of core speeds to maximize the throughput of thermally constrained multi-core processors. In *Proc. International Conference on Computer-Aided Design*, pages 537–542, Nov. 2008.

[44] Lin Yuan and Gang Qu. Alt-dvs: Dynamic voltage scaling with awareness of leakage and temperature for real-time systems. In *Proc. Conference on Adaptive Hardware and Systems, NASA/ESA*, pages 660–670, 2007.

[45] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. De Micheli. Temperature control of high-performance multi-core platforms using convex optimization. In *Proc. Design automation and test in Europe*, pages 110–115, New York, NY, USA, 2008. ACM.

[46] L. Yuan, S. Leventhal, and G. Qu. Temperature-aware leakage minimization technique for real-time systems. In *Proc. International Conference on Computer Aided Design*, pages 761–764, 2006.

[47] M. Bao, A. Andrei, P. Eles, and Z. Peng. On-line temperature-aware idle time distribution for leakage energy optimization. In *Proc. The 6th International Symposium on Electronic Design, Test and Applications*, Jan 2011.

[48] M. Bao, A. Andrei, P. Eles, and Z. Peng. Temperature-aware idle time distribution for energy optimization with dynamic voltage scaling. In *Proc. The 10th Swedish System-on-Chip Conference*, Mar 2010.

[49] M. Bao, A. Andrei, P. Eles, and Z. Peng. Temperature-aware idle time distribution for energy optimization with dynamic voltage scaling. In *Proc. Design Automation and Test in Europe*, pages 21–26, Mar. 2010.

[50] M. Bao, A. Andrei, P. Eles, and Z. Peng. On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration. In *Proc. Design Automation Conference*, pages 490–495, Jul. 2009.

[51] M. Bao, A. Andrei, P. Eles, and Z. Peng. Temperature-aware voltage selection for energy optimization. In *Proc. The 9th Swedish System-on-Chip Conference*, May 2009.

[52] M. Bao, A. Andrei, P. Eles, and Z. Peng. An energy efficient technique for temperature-aware voltage selection. Technical Report 4, Linkping UniversityLinkping University, ESLAB - Embedded Systems Laboratory, The Institute of Technology, 2009.

[53] M. Bao, A. Andrei, P. Eles, and Z. Peng. Temperature-aware task mapping for energy optimization with dynamic voltage scaling. In *Proc. the 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, pages 1–6, Apr. 2008.

[54] M. Bao, A. Andrei, P. Eles, and Z. Peng. Temperature-aware voltage selection for energy optimization. In *Proc. Design Automation Test in Europe*, pages 1083–1086, Apr. 2008.

[55] S. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proc. International Conference on Computer Aided Design*, pages 721–725, Nov. 2002.

[56] J. Choi, A. Bansal, M. Meterelliyoz, J. Murthy, and K. Roy. Leakage power dependent temperature estimation to predict thermal runaway in finfet circuits. In *Proc. International Conference on Computer-Aided Design*, pages 583–586, 2006.

[57] H. Su, F. Liu, A. Acar, and S. Nassif. Full chip leakage estimation considering power supply and temperature variations. In *Proc. International Symposium on Low Power Electronics and Design*, 2003.

[58] Y. Tsai, A. Ankadi, and N. Vijaykrishnan. Chippower: An architecture-level leakage simulator. In *Proc. International SOC Conference*, 2004.

[59] Y. Liu, H. Yang, R. P. Dick, H. Wang, and L. Shang. Thermal vs energy optimization for dvfs-enabled processors in embedded systems. In *Proc. International Symposium on Quality Electronic Design*, pages 204–209, Mar. 2007.

[60] `http://ffmpeg.mplayerhq.hu/`.

[61] B. Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill Science Engineering, Erewhon, NC, Aug. 2000.

[62] S. Hsu, A. Alvandpour, S. Mathew, S. L. Lu, R. K. Krishnamurthy, and S. Borkar. A 4.5-ghz 130-nm 32-kb l0 cache with a leakage-tolerant self reverse-bias bitline scheme. *IEEE Journal of Solid-State Circuits*, 38:755–761, May 2003.

[63] A. Macii, E. Macii, and M. Poncino. Improving the efficiency of memory partitioning by address clustering. In *Proc. of the Design Automation and Test in Europe*, pages 18–23, 2003.

[64] R. Rao and S. Vrudhula. Performance optimal processor throttling under thermal constraints. In *Proc. International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, pages 257–266, Nov. 2007.

[65] W. H. Pressa, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, The Edinburgh Building, Cambridge, UK, 2007.

[66] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. Society for Industrial Mathematics, 1987.

[67] Application note: Powerpc 970mp thermal considerations, Jul. 2006.

[68] Intel core 2 duo mobile processors on 65-nm process for embedded applications: Thermal design guide, Aug. 2007.

[69] Intel core 2 duo mobile processors on 45-nm process for embedded applications: Thermal design guide, Jun. 2008.

[70] Luca Benini, Davide Bertozzi, Davide Bruni, Nicola Drago, Franco Fummi, and Massimo Poncino. Systemc cosimulation and emulation of multiprocessor soc designs. *Computer*, 36:53–59, 2003.

LINKÖPINGS UNIVERSITET

Avdelning, institution
Division, department

Institutionen för datavetenskap

Department of Computer and Information Science

Datum
Date

2010-12-16

Språk Language	
[ ] Svenska/Swedish	
[X] Engelska/English	
[ ]	

Rapporttyp Report category	
[X] Licentiatavhandling	
[ ] Examensarbete	
[ ] C-uppsats	
[ ] D-uppsats	
[ ] Övrig rapport	

ISBN 978-91-7393-264-6

ISRN LiU-Tek-Lic-2010:30

Serietitel och serienummer Title of series, numbering **ISSN** 0280-7971

Linköping Studies in Science and Technology

Thesis No. 1459

URL för elektronisk version

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-60855

Titel
Title

System-Level Techniques for Temperature-Aware Energy Optimization

Författare
Author

Min Bao

Sammanfattning
Abstract

Energy consumption has become one of the main design constraints in today's integrated circuits. Techniques for energy optimization, from circuit-level up to system-level, have been intensively researched.

The advent of large-scale integration with deep sub-micron technologies has led to both high power densities and high chip working temperatures. At the same time, leakage power is becoming the dominant power consumption source of circuits, due to continuously lowered threshold voltages, as technology scales. In this context, temperature is an important parameter. One aspect, of particular interest for this thesis, is the strong inter-dependency between leakage and temperature. Apart from leakage power, temperature also has an important impact on circuit delay and, implicitly, on the frequency, mainly through its influence on carrier mobility and threshold voltage. For power-aware design techniques, temperature has become a major factor to be considered. In this thesis, we address the issue of system-level energy optimization for real-time embedded systems taking temperature aspects into consideration.

We have investigated two problems in this thesis: (1) Energy optimization via temperature-aware dynamic voltage/frequency scaling (DVFS). (2) Energy optimization through temperature-aware idle time (or slack) distribution (ITD). For the above two problems, we have proposed off-line techniques where only static slack is considered. To further improve energy efficiency, we have also proposed on-line techniques, which make use of both static and dynamic slack. Experimental results have demonstrated that consider-able improvement of the energy efficiency can be achieved by applying our temperature-aware optimization techniques. Another contribution of this thesis is an analytical temperature analysis approach which is both accurate and sufficiently fast to be used inside an energy optimization loop.

Nyckelord
Keywords

Temperature-Aware Design, Energy Optimization, System-Level Design, Embedded Systems

Department of Computer and Information Science
Linköpings universitet

Linköpings Studies in Science and Technology
Faculty of Arts and Sciences – Licentiate Theses

No 17 **Vojin Plavsic:** Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. (Available at: FOA, Box 1165, S-581 11 Linköping, Sweden. FOA Report B30062E)
No 28 **Arne Jönsson, Mikael Patel:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms, 1984.
No 29 **Johnny Eckerland:** Retargeting of an Incremental Code Generator, 1984.
No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching, 1985.
No 52 **Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems, 1985.
No 60 **Johan Fagerström:** Simulation and Evaluation of Architecture based on Asynchronous Processes, 1985.
No 71 **Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System, 1987.
No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems, 1986.
No 73 **Ola Strömfors:** A Structure Editor for Documents and Programs, 1986.
No 74 **Christos Levcopoulos:** New Results about the Approximation Behavior of the Greedy Triangulation, 1986.
No 104 **Shamsul I. Chowdhury: S**tatistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology, 1987.
No 108 **Rober Bilos:** Incremental Scanning and Token-Based Editing, 1987.
No 111 **Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments, 1987.
No 113 **Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge, 1987.
No 118 **Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs, 1987.
No 126 **Dan Strömberg:** Transfer and Distribution of Application Programs, 1987.
No 127 **Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems, 1987.
No 139 **Christer Bäckström:** Reasoning about Interdependent Actions, 1988.
No 140 **Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing, 1988.
No 146 **Johan Hultman:** A Software System for Defining and Controlling Actions in a Mechanical System, 1988.
No 150 **Tim Hansen:** Diagnosing Faults using Knowledge about Malfunctioning Behavior, 1988.
No 165 **Jonas Löwgren:** Supporting Design and Management of Expert System User Interfaces, 1989.
No 166 **Ola Petersson:** On Adaptive Sorting in Sequential and Parallel Models, 1989.
No 174 **Yngve Larsson:** Dynamic Configuration in a Distributed Environment, 1989.
No 177 **Peter Åberg:** Design of a Multiple View Presentation and Interaction Manager, 1989.
No 181 **Henrik Eriksson:** A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.
No 184 **Ivan Rankin:** The Deep Generation of Text in Expert Critiquing Systems, 1989.
No 187 **Simin Nadjm-Tehrani:** Contributions to the Declarative Approach to Debugging Prolog Programs, 1989.
No 189 **Magnus Merkel:** Temporal Information in Natural Language, 1989.
No 196 **Ulf Nilsson:** A Systematic Approach to Abstract Interpretation of Logic Programs, 1989.
No 197 **Staffan Bonnier:** Horn Clause Logic with External Procedures: Towards a Theoretical Framework, 1989.
No 203 **Christer Hansson:** A Prototype System for Logical Reasoning about Time and Action, 1990.
No 212 **Björn Fjellborg:** An Approach to Extraction of Pipeline Structures for VLSI High-Level Synthesis, 1990.
No 230 **Patrick Doherty:** A Three-Valued Approach to Non-Monotonic Reasoning, 1990.
No 237 **Tomas Sokolnicki:** Coaching Partial Plans: An Approach to Knowledge-Based Tutoring, 1990.
No 250 **Lars Strömberg:** Postmortem Debugging of Distributed Systems, 1990.
No 253 **Torbjörn Näslund:** SLDFA-Resolution - Computing Answers for Negative Queries, 1990.
No 260 **Peter D. Holmes:** Using Connectivity Graphs to Support Map-Related Reasoning, 1991.
No 283 **Olof Johansson:** Improving Implementation of Graphical User Interfaces for Object-Oriented Knowledge- Bases, 1991.
No 298 **Rolf G Larsson:** Aktivitetsbaserad kalkylering i ett nytt ekonomisystem, 1991.
No 318 **Lena Srömbäck:** Studies in Extended Unification-Based Formalism for Linguistic Description: An Algorithm for Feature Structures with Disjunction and a Proposal for Flexible Systems, 1992.
No 319 **Mikael Pettersson:** DML-A Language and System for the Generation of Efficient Compilers from Denotational Specification, 1992.
No 326 **Andreas Kågedal:** Logic Programming with External Procedures: an Implementation, 1992.
No 328 **Patrick Lambrix:** Aspects of Version Management of Composite Objects, 1992.
No 333 **Xinli Gu:** Testability Analysis and Improvement in High-Level Synthesis Systems, 1992.
No 335 **Torbjörn Näslund:** On the Role of Evaluations in Iterative Development of Managerial Support Systems, 1992.
No 348 **Ulf Cederling:** Industrial Software Development - a Case Study, 1992.
No 352 **Magnus Morin:** Predictable Cyclic Computations in Autonomous Systems: A Computational Model and Implementation, 1992.
No 371 **Mehran Noghabai:** Evaluation of Strategic Investments in Information Technology, 1993.
No 378 **Mats Larsson:** A Transformational Approach to Formal Digital System Design, 1993.
No 380 **Johan Ringström:** Compiler Generation for Parallel Languages from Denotational Specifications, 1993.

No 381	**Michael Jansson:** Propagation of Change in an Intelligent Information System, 1993.
No 383	**Jonni Harrius:** An Architecture and a Knowledge Representation Model for Expert Critiquing Systems, 1993.
No 386	**Per Österling:** Symbolic Modelling of the Dynamic Environments of Autonomous Agents, 1993.
No 398	**Johan Boye:** Dependency-based Groudness Analysis of Functional Logic Programs, 1993.
No 402	**Lars Degerstedt:** Tabulated Resolution for Well Founded Semantics, 1993.
No 406	**Anna Moberg:** Satellitkontor - en studie av kommunikationsmönster vid arbete på distans, 1993.
No 414	**Peter Carlsson:** Separation av företagsledning och finansiering - fallstudier av företagsledarutköp ur ett agent-teoretiskt perspektiv, 1994.
No 417	**Camilla Sjöström:** Revision och lagreglering - ett historiskt perspektiv, 1994.
No 436	**Cecilia Sjöberg:** Voices in Design: Argumentation in Participatory Development, 1994.
No 437	**Lars Viklund:** Contributions to a High-level Programming Environment for a Scientific Computing, 1994.
No 440	**Peter Loborg:** Error Recovery Support in Manufacturing Control Systems, 1994.
FHS 3/94	**Owen Eriksson:** Informationssystem med verksamhetskvalitet - utvärdering baserat på ett verksamhetsinriktat och samskapande perspektiv, 1994.
FHS 4/94	**Karin Pettersson:** Informationssystemstrukturering, ansvarsfördelning och användarinflytande - En komparativ studie med utgångspunkt i två informationssystemstrategier, 1994.
No 441	**Lars Poignant:** Informationsteknologi och företagsetablering - Effekter på produktivitet och region, 1994.
No 446	**Gustav Fahl:** Object Views of Relational Data in Multidatabase Systems, 1994.
No 450	**Henrik Nilsson:** A Declarative Approach to Debugging for Lazy Functional Languages, 1994.
No 451	**Jonas Lind:** Creditor - Firm Relations: an Interdisciplinary Analysis, 1994.
No 452	**Martin Sköld:** Active Rules based on Object Relational Queries - Efficient Change Monitoring Techniques, 1994.
No 455	**Pär Carlshamre:** A Collaborative Approach to Usability Engineering: Technical Communicators and System Developers in Usability-Oriented Systems Development, 1994.
FHS 5/94	**Stefan Cronholm:** Varför CASE-verktyg i systemutveckling? - En motiv- och konsekvensstudie avseende arbetssätt och arbetsformer, 1994.
No 462	**Mikael Lindvall:** A Study of Traceability in Object-Oriented Systems Development, 1994.
No 463	**Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av Sandviks förvärv av Bahco Verktyg, 1994.
No 464	**Hans Olsén:** Collage Induction: Proving Properties of Logic Programs by Program Synthesis, 1994.
No 469	**Lars Karlsson:** Specification and Synthesis of Plans Using the Features and Fluents Framework, 1995.
No 473	**Ulf Söderman:** On Conceptual Modelling of Mode Switching Systems, 1995.
No 475	**Choong-ho Yi:** Reasoning about Concurrent Actions in the Trajectory Semantics, 1995.
No 476	**Bo Lagerström:** Successiv resultatavräkning av pågående arbeten. - Fallstudier i tre byggföretag, 1995.
No 478	**Peter Jonsson:** Complexity of State-Variable Planning under Structural Restrictions, 1995.
FHS 7/95	**Anders Avdic:** Arbetsintegrerad systemutveckling med kalkylprogram, 1995.
No 482	**Eva L Ragnemalm:** Towards Student Modelling through Collaborative Dialogue with a Learning Companion, 1995.
No 488	**Eva Toller:** Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995.
No 489	**Erik Stoy:** A Petri Net Based Unified Representation for Hardware/Software Co-Design, 1995.
No 497	**Johan Herber:** Environment Support for Building Structured Mathematical Models, 1995.
No 498	**Stefan Svenberg:** Structure-Driven Derivation of Inter-Lingual Functor-Argument Trees for Multi-Lingual Generation, 1995.
No 503	**Hee-Cheol Kim:** Prediction and Postdiction under Uncertainty, 1995.
FHS 8/95	**Dan Fristedt:** Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995.
FHS 9/95	**Malin Bergvall:** Systemförvaltning i praktiken - en kvalitativ studie avseende centrala begrepp, aktiviteter och ansvarsroller, 1995.
No 513	**Joachim Karlsson:** Towards a Strategy for Software Requirements Selection, 1995.
No 517	**Jakob Axelsson:** Schedulability-Driven Partitioning of Heterogeneous Real-Time Systems, 1995.
No 518	**Göran Forslund:** Toward Cooperative Advice-Giving Systems: The Expert Systems Experience, 1995.
No 522	**Jörgen Andersson:** Bilder av småföretagares ekonomistyrning, 1995.
No 538	**Staffan Flodin:** Efficient Management of Object-Oriented Queries with Late Binding, 1996.
No 545	**Vadim Engelson:** An Approach to Automatic Construction of Graphical User Interfaces for Applications in Scientific Computing, 1996.
No 546	**Magnus Werner :** Multidatabase Integration using Polymorphic Queries and Views, 1996.
FiF-a 1/96	**Mikael Lind:** Affärsprocessinriktad förändringsanalys - utveckling och tillämpning av synsätt och metod, 1996.
No 549	**Jonas Hallberg:** High-Level Synthesis under Local Timing Constraints, 1996.
No 550	**Kristina Larsen:** Förutsättningar och begränsningar för arbete på distans - erfarenheter från fyra svenska företag. 1996.
No 557	**Mikael Johansson:** Quality Functions for Requirements Engineering Methods, 1996.
No 558	**Patrik Nordling:** The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
No 561	**Anders Ekman:** Exploration of Polygonal Environments, 1996.
No 563	**Niclas Andersson:** Compilation of Mathematical Models to Parallel Code, 1996.
No 567	**Johan Jenvald:** Simulation and Data Collection in Battle Training, 1996.

No 575 **Niclas Ohlsson:** Software Quality Engineering by Early Identification of Fault-Prone Modules, 1996.

No 576 **Mikael Ericsson:** Commenting Systems as Design Support—A Wizard-of-Oz Study, 1996.

No 587 **Jörgen Lindström:** Chefers användning av kommunikationsteknik, 1996.

No 589 **Esa Falkenroth:** Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.

No 591 **Niclas Wahllöf:** A Default Extension to Description Logics and its Applications, 1996.

No 595 **Annika Larsson:** Ekonomisk Styrning och Organisatorisk Passion - ett interaktivt perspektiv, 1997.

No 597 **Ling Lin:** A Value-based Indexing Technique for Time Sequences, 1997.

No 598 **Rego Granlund:** C^3Fire - A Microworld Supporting Emergency Management Training, 1997.

No 599 **Peter Ingels:** A Robust Text Processing Technique Applied to Lexical Error Recovery, 1997.

No 607 **Per-Arne Persson:** Toward a Grounded Theory for Support of Command and Control in Military Coalitions, 1997.

No 609 **Jonas S Karlsson:** A Scalable Data Structure for a Parallel Data Server, 1997.

FiF-a 4 **Carita Åbom:** Videomötesteknik i olika affärssituationer - möjligheter och hinder, 1997.

FiF-a 6 **Tommy Wedlund**: Att skapa en företagsanpassad systemutvecklingsmodell - genom rekonstruktion, värdering och vidareutveckling i T50-bolag inom ABB, 1997.

No 615 **Silvia Coradeschi**: A Decision-Mechanism for Reactive and Coordinated Agents, 1997.

No 623 **Jan Ollinen:** Det flexibla kontorets utveckling på Digital - Ett stöd för multiflex? 1997.

No 626 **David Byers:** Towards Estimating Software Testability Using Static Analysis, 1997.

No 627 **Fredrik Eklund:** Declarative Error Diagnosis of GAPLog Programs, 1997.

No 629 **Gunilla Ivefors:** Krigsspel och Informationsteknik inför en oförutsägbar framtid, 1997**.**

No 631 **Jens-Olof Lindh:** Analysing Traffic Safety from a Case-Based Reasoning Perspective, 1997

No 639 **Jukka Mäki-Turja:**. Smalltalk - a suitable Real-Time Language, 1997.

No 640 **Juha Takkinen**: CAFE: Towards a Conceptual Model for Information Management in Electronic Mail, 1997.

No 643 **Man Lin**: Formal Analysis of Reactive Rule-based Programs, 1997.

No 653 **Mats Gustafsson**: Bringing Role-Based Access Control to Distributed Systems, 1997.

FiF-a 13 **Boris Karlsson:** Metodanalys för förståelse och utveckling av systemutvecklingsverksamhet. Analys och värdering av systemutvecklingsmodeller och dess användning, 1997.

No 674 **Marcus Bjäreland:** Two Aspects of Automating Logics of Action and Change - Regression and Tractability, 1998.

No 676 **Jan Håkegård**: Hierarchical Test Architecture and Board-Level Test Controller Synthesis, 1998.

No 668 **Per-Ove Zetterlund**: Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.

No 675 **Jimmy Tjäder**: Projektledaren & planen - en studie av projektledning i tre installations- och systemutvecklingsprojekt, 1998.

FiF-a 14 **Ulf Melin**: Informationssystem vid ökad affärs- och processorientering - egenskaper, strategier och utveckling, 1998.

No 695 **Tim Heyer**: COMPASS: Introduction of Formal Methods in Code Development and Inspection, 1998.

No 700 **Patrik Hägglund:** Programming Languages for Computer Algebra, 1998.

FiF-a 16 **Marie-Therese Christiansson:** Inter-organisatorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationssystem, 1998.

No 712 **Christina Wennestam:** Information om immateriella resurser. Investeringar i forskning och utveckling samt i personal inom skogsindustrin, 1998.

No 719 **Joakim Gustafsson:** Extending Temporal Action Logic for Ramification and Concurrency, 1998.

No 723 **Henrik André-Jönsson:** Indexing time-series data using text indexing methods, 1999.

No 725 **Erik Larsson:** High-Level Testability Analysis and Enhancement Techniques, 1998.

No 730 **Carl-Johan Westin:** Informationsförsörjning: en fråga om ansvar - aktiviteter och uppdrag i fem stora svenska organisationers operativa informationsförsörjning, 1998.

No 731 **Åse Jansson:** Miljöhänsyn - en del i företags styrning, 1998.

No 733 **Thomas Padron-McCarthy:** Performance-Polymorphic Declarative Queries, 1998.

No 734 **Anders Bäckström:** Värdeskapande kreditgivning - Kreditriskhantering ur ett agentteoretiskt perspektiv, 1998.

FiF-a 21 **Ulf Seigerroth:** Integration av förändringsmetoder - en modell för välgrundad metodintegration, 1999.

FiF-a 22 **Fredrik Öberg:** Object-Oriented Frameworks - A New Strategy for Case Tool Development, 1998.

No 737 **Jonas Mellin:** Predictable Event Monitoring, 1998.

No 738 **Joakim Eriksson:** Specifying and Managing Rules in an Active Real-Time Database System, 1998.

FiF-a 25 **Bengt E W Andersson:** Samverkande informationssystem mellan aktörer i offentliga åtaganden - En teori om aktörsarenor i samverkan om utbyte av information, 1998.

No 742 **Pawel Pietrzak:** Static Incorrectness Diagnosis of CLP (FD), 1999.

No 748 **Tobias Ritzau:** Real-Time Reference Counting in RT-Java, 1999.

No 751 **Anders Ferntoft:** Elektronisk affärskommunikation - kontaktkostnader och kontaktprocesser mellan kunder och leverantörer på producentmarknader, 1999.

No 752 **Jo Skåmedal:** Arbete på distans och arbetsformens påverkan på resor och resmönster, 1999.

No 753 **Johan Alvehus:** Mötets metaforer. En studie av berättelser om möten, 1999.

No 754 **Magnus Lindahl:** Bankens villkor i låneavtal vid kreditgivning till högt belånade företagsförvärv: En studie ur ett agentteoretiskt perspektiv, 2000.

No 766 **Martin V. Howard:** Designing dynamic visualizations of temporal data, 1999.
No 769 **Jesper Andersson:** Towards Reactive Software Architectures, 1999.
No 775 **Anders Henriksson:** Unique kernel diagnosis, 1999.
FiF-a 30 **Pär J. Ågerfalk:** Pragmatization of Information Systems - A Theoretical and Methodological Outline, 1999.
No 787 **Charlotte Björkegren:** Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.
No 788 **Håkan Nilsson:** Informationsteknik som drivkraft i granskningsprocessen - En studie av fyra revisionsbyråer, 2000.
No 790 **Erik Berglund:** Use-Oriented Documentation in Software Development, 1999.
No 791 **Klas Gäre:** Verksamhetsförändringar i samband med IS-införande, 1999.
No 800 **Anders Subotic:** Software Quality Inspection, 1999.
No 807 **Svein Bergum**: Managerial communication in telework, 2000.
No 809 **Flavius Gruian:** Energy-Aware Design of Digital Systems, 2000.
FiF-a 32 **Karin Hedström:** Kunskapsanvändning och kunskapsutveckling hos verksamhetskonsulter - Erfarenheter från ett FOU-samarbete, 2000.
No 808 **Linda Askenäs:** Affärssystemet - En studie om teknikens aktiva och passiva roll i en organisation, 2000.
No 820 **Jean Paul Meynard:** Control of industrial robots through high-level task programming, 2000.
No 823 **Lars Hult:** Publika Gränsytor - ett designexempel, 2000.
No 832 **Paul Pop:** Scheduling and Communication Synthesis for Distributed Real-Time Systems, 2000.
FiF-a 34 **Göran Hultgren:** Nätverksinriktad Förändringsanalys - perspektiv och metoder som stöd för förståelse och utveckling av affärsrelationer och informationssystem, 2000.
No 842 **Magnus Kald:** The role of management control systems in strategic business units, 2000.
No 844 **Mikael Cäker:** Vad kostar kunden? Modeller för intern redovisning, 2000.
FiF-a 37 **Ewa Braf**: Organisationers kunskapsverksamheter - en kritisk studie av "knowledge management", 2000.
FiF-a 40 **Henrik Lindberg:** Webbaserade affärsprocesser - Möjligheter och begränsningar, 2000.
FiF-a 41 **Benneth Christiansson:** Att komponentbasera informationssystem - Vad säger teori och praktik?, 2000.
No. 854 **Ola Pettersson:** Deliberation in a Mobile Robot, 2000.
No 863 **Dan Lawesson**: Towards Behavioral Model Fault Isolation for Object Oriented Control Systems, 2000.
No 881 **Johan Moe:** Execution Tracing of Large Distributed Systems, 2001.
No 882 **Yuxiao Zhao:** XML-based Frameworks for Internet Commerce and an Implementation of B2B e-procurement, 2001.
No 890 **Annika Flycht-Eriksson:** Domain Knowledge Management in Information-providing Dialogue systems, 2001.
FiF-a 47 **Per-Arne Segerkvis**t: Webbaserade imaginära organisationers samverkansformer: Informationssystemarkitektur och aktörssamverkan som förutsättningar för affärsprocesser, 2001.
No 894 **Stefan Svarén:** Styrning av investeringar i divisionaliserade företag - Ett koncernperspektiv, 2001.
No 906 **Lin Han**: Secure and Scalable E-Service Software Delivery, 2001.
No 917 **Emma Hansson:** Optionsprogram för anställda - en studie av svenska börsföretag, 2001.
No 916 **Susanne Odar:** IT som stöd för strategiska beslut, en studie av datorimplementerade modeller av verksamhet som stöd för beslut om anskaffning av JAS 1982, 2002.
FiF-a-49 **Stefan Holgersson:** IT-system och filtrering av verksamhetskunskap - kvalitetsproblem vid analyser och beslutsfattande som bygger på uppgifter hämtade från polisens IT-system, 2001.
FiF-a-51 **Per Oscarsson:** Informationssäkerhet i verksamheter - begrepp och modeller som stöd för förståelse av informationssäkerhet och dess hantering, 2001.
No 919 **Luis Alejandro Cortes:** A Petri Net Based Modeling and Verification Technique for Real-Time Embedded Systems, 2001.
No 915 **Niklas Sandell:** Redovisning i skuggan av en bankkris - Värdering av fastigheter. 2001.
No 931 **Fredrik Elg:** Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konfidens i kontroll av mikrovärlden Moro, 2002.
No 933 **Peter Aronsson:** Automatic Parallelization of Simulation Code from Equation Based Simulation Languages, 2002.
No 938 **Bourhane Kadmiry**: Fuzzy Control of Unmanned Helicopter, 2002.
No 942 **Patrik Haslum**: Prediction as a Knowledge Representation Problem: A Case Study in Model Design, 2002.
No 956 **Robert Sevenius:** On the instruments of governance - A law & economics study of capital instruments in limited liability companies, 2002.
FiF-a 58 **Johan Petersson:** Lokala elektroniska marknadsplatser - informationssystem för platsbundna affärer, 2002.
No 964 **Peter Bunus:** Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.
No 973 **Gert Jervan:** High-Level Test Generation and Built-In Self-Test Techniques for Digital Systems, 2002.
No 958 **Fredrika Berglund:** Management Control and Strategy - a Case Study of Pharmaceutical Drug Development, 2002.
FiF-a 61 **Fredrik Karlsson:** Meta-Method for Method Configuration - A Rational Unified Process Case, 2002.
No 985 **Sorin Manolache:** Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times, 2002.
No 982 **Diana Szentiványi:** Performance and Availability Trade-offs in Fault-Tolerant Middleware, 2002.
No 989 **Iakov Nakhimovski:** Modeling and Simulation of Contacting Flexible Bodies in Multibody Systems, 2002.
No 990 **Levon Saldamli:** PDEModelica - Towards a High-Level Language for Modeling with Partial Differential Equations, 2002.
No 991 **Almut Herzog:** Secure Execution Environment for Java Electronic Services, 2002.
No 999 **Jon Edvardsson:** Contributions to Program- and Specification-based Test Data Generation, 2002.
No 1000 **Anders Arpteg:** Adaptive Semi-structured Information Extraction, 2002.

No 1192	**Nicklas Bergfeldt:** Towards Detached Communication for Robot Cooperation, 2005.
No 1194	**Dennis Maciuszek:** Towards Dependable Virtual Companions for Later Life, 2005.
No 1204	**Beatrice Alenljung:** Decision-making in the Requirements Engineering Process: A Human-centered Approach, 2005.
No 1206	**Anders Larsson:** System-on-Chip Test Scheduling and Test Infrastructure Design, 2005.
No 1207	**John Wilander:** Policy and Implementation Assurance for Software Security, 2005.
No 1209	**Andreas Käll:** Översättningar av en managementmodell - En studie av införandet av Balanced Scorecard i ett landsting, 2005.
No 1225	**He Tan:** Aligning and Merging Biomedical Ontologies, 2006.
No 1228	**Artur Wilk:** Descriptive Types for XML Query Language Xcerpt, 2006.
No 1229	**Per Olof Pettersson:** Sampling-based Path Planning for an Autonomous Helicopter, 2006.
No 1231	**Kalle Burbeck:** Adaptive Real-time Anomaly Detection for Safeguarding Critical Networks, 2006.
No 1233	**Daniela Mihailescu:** Implementation Methodology in Action: A Study of an Enterprise Systems Implementation Methodology, 2006.
No 1244	**Jörgen Skågeby:** Public and Non-public gifting on the Internet, 2006.
No 1248	**Karolina Eliasson:** The Use of Case-Based Reasoning in a Human-Robot Dialog System, 2006.
No 1263	**Misook Park-Westman:** Managing Competence Development Programs in a Cross-Cultural Organisation - What are the Barriers and Enablers, 2006.
FiF-a 90	**Amra Halilovic:** Ett praktikperspektiv på hantering av mjukvarukomponenter, 2006.
No 1272	**Raquel Flodström**: A Framework for the Strategic Management of Information Technology, 2006.
No 1277	**Viacheslav Izosimov:** Scheduling and Optimization of Fault-Tolerant Embedded Systems, 2006.
No 1283	**Håkan Hasewinkel:** A Blueprint for Using Commercial Games off the Shelf in Defence Training, Education and Research Simulations, 2006.
FiF-a 91	**Hanna Broberg:** Verksamhetsanpassade IT-stöd - Designteori och metod, 2006.
No 1286	**Robert Kaminski:** Towards an XML Document Restructuring Framework, 2006.
No 1293	**Jiri Trnka:** Prerequisites for data sharing in emergency management, 2007.
No 1302	**Björn Hägglund:** A Framework for Designing Constraint Stores, 2007.
No 1303	**Daniel Andreasson**: Slack-Time Aware Dynamic Routing Schemes for On-Chip Networks, 2007.
No 1305	**Magnus Ingmarsson:** Modelling User Tasks and Intentions for Service Discovery in Ubiquitous Computing, 2007.
No 1306	**Gustaf Svedjemo**: Ontology as Conceptual Schema when Modelling Historical Maps for Database Storage, 2007.
No 1307	**Gianpaolo Conte**: Navigation Functionalities for an Autonomous UAV Helicopter, 2007.
No 1309	**Ola Leifler**: User-Centric Critiquing in Command and Control: The DKExpert and ComPlan Approaches, 2007.
No 1312	**Henrik Svensson**: Embodied simulation as off-line representation, 2007.
No 1313	**Zhiyuan He:** System-on-Chip Test Scheduling with Defect-Probability and Temperature Considerations, 2007.
No 1317	**Jonas Elmqvist:** Components, Safety Interfaces and Compositional Analysis, 2007.
No 1320	**Håkan Sundblad:** Question Classification in Question Answering Systems, 2007.
No 1323	**Magnus Lundqvist**: Information Demand and Use: Improving Information Flow within Small-scale Business Contexts, 2007.
No 1329	**Martin Magnusson:** Deductive Planning and Composite Actions in Temporal Action Logic, 2007.
No 1331	**Mikael Asplund:** Restoring Consistency after Network Partitions, 2007.
No 1332	**Martin Fransson:** Towards Individualized Drug Dosage - General Methods and Case Studies, 2007.
No 1333	**Karin Camara:** A Visual Query Language Served by a Multi-sensor Environment, 2007.
No 1337	**David Broman:** Safety, Security, and Semantic Aspects of Equation-Based Object-Oriented Languages and Environments, 2007.
No 1339	**Mikhail Chalabine:** Invasive Interactive Parallelization, 2007.
No 1351	**Susanna Nilsson:** A Holistic Approach to Usability Evaluations of Mixed Reality Systems, 2008.
No 1353	**Shanai Ardi:** A Model and Implementation of a Security Plug-in for the Software Life Cycle, 2008.
No 1356	**Erik Kuiper:** Mobility and Routing in a Delay-tolerant Network of Unmanned Aerial Vehicles, 2008.
No 1359	**Jana Rambusch**: Situated Play, 2008.
No 1361	**Martin Karresand:** Completing the Picture - Fragments and Back Again, 2008.
No 1363	**Per Nyblom:** Dynamic Abstraction for Interleaved Task Planning and Execution, 2008.
No 1371	**Fredrik Lantz:**Terrain Object Recognition and Context Fusion for Decision Support, 2008.
No 1373	**Martin Östlund:** Assistance Plus: 3D-mediated Advice-giving on Pharmaceutical Products, 2008.
No 1381	**Håkan Lundvall:** Automatic Parallelization using Pipelining for Equation-Based Simulation Languages, 2008.
No 1386	**Mirko Thorstensson:** Using Observers for Model Based Data Collection in Distributed Tactical Operations, 2008.
No 1387	**Bahlol Rahimi:** Implementation of Health Information Systems, 2008.
No 1392	**Maria Holmqvist:** Word Alignment by Re-using Parallel Phrases, 2008.
No 1393	**Mattias Eriksson**: Integrated Software Pipelining, 2009.
No 1401	**Annika Öhgren:** Towards an Ontology Development Methodology for Small and Medium-sized Enterprises, 2009.
No 1410	**Rickard Holsmark:** Deadlock Free Routing in Mesh Networks on Chip with Regions, 2009.
No 1421	**Sara Stymne:** Compound Processing for Phrase-Based Statistical Machine Translation, 2009.
No 1427	**Tommy Ellqvist**: Supporting Scientific Collaboration through Workflows and Provenance, 2009.
No 1450	**Fabian Segelström:** Visualisations in Service Design, 2010.
No 1459	**Min Bao**: System Level Techniques for Temperature-Aware Energy Optimization, 2010.