3-26-2013

Leakage Temperature Dependency Aware Real-Time Scheduling for Power and Thermal Optimization

Vivek Chaturvedi
Florida International University, vchaturv@fiu.edu

DOI: 10.25148/etd.FI13042337

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

LEAKAGE TEMPERATURE DEPENDENCY AWARE REAL-TIME SCHEDULING FOR POWER AND THERMAL OPTIMIZATION

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Vivek Chaturvedi

2013

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Vivek Chaturvedi, and entitled Leakage Temperature Dependency Aware Real-Time Scheduling for Power and Thermal Optimization, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Raju Rangaswami

Jean H. Andrian

Cheng-Xian Lin

Nezih Pala

Gang Quan, Major Professor

Date of Defense: March 26, 2013

The dissertation of Vivek Chaturvedi is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi N. Reddi
University Graduate School

Florida Internatinal University, 2013

DEDICATION

I would like to dedicate this Doctoral dissertation to my dearest mother Mrs. Sharda Chaturvedi and loving father Late Mr. J.P. Chaturvedi. Without their love, understanding, support, and encouragement, the completion of this endeavor would never have been possible.

ACKNOWLEDGMENTS

First and foremost, I would like to thank God Almighty for giving me the strength and patience to accomplish my dream of getting a Ph.D degree.

Next, I would like to express my deepest gratitude to my advisor Dr. Gang Quan. I feel fortunate and blessed to have an advisor like Dr. Quan, who gave me constant guidance, personal attention, suggestions and endless encouragement during the last four years of my doctoral study. I truly admire his perseverance, depth of knowledge and strong dedication to students and quality research. Dr. Quan's thoughts and suggestions have not only helped me to complete this dissertation successfully, but have also helped me to overcome several personal problems. I am thankful to him for his patience and support.

I also want to take this opportunity to thank Dr. Quan's family for being warm and welcoming.

I am very grateful to my Ph.D. committee members: Dr. Jean H. Andrian, Dr. Nezih Pala, Dr. Cheng-Xian Lin and Dr. Raju Rangaswami for their thoughtful insights and suggestions in improving my dissertation. I am extremely proud to have such a wonderful and knowledgeable people serving on my dissertation committee.

I want to give my heartfelt thanks to Dr. Adrian Nunez-Aldana, Dr. Nazanin Mansouri, Dr. Shangping Ren and Mr. Soroush Shakib for their endless encouragement and support in accomplishing my career objectives.

I am thankful to the staff of ECE department at FIU, specially to Mrs. Maria Benincasa, Mrs. Pat Brammer and Mrs. Ana Saenz for their great commitment to student services.

Next, I would like to thank my lab mates at ARCS lab for creating a wonderfully collaborative and enriching work environment filled with fun and laughter. I am confident that our friendship and cooperation will go a long way.

God blessed me with some wonderful friends, who are my solid support system. I am very thankful to them for providing me with unconditional love, support and care. I greatly value your friendship and I deeply appreciate your faith in me.

Finally, and above all, I want to thank my family for their unlimited love, faith, encouragement, blessings and prayers. My life-long gratitude go to my dearest mother Mrs. Sharda Chaturvedi and my loving father late Mr. J.P. Chaturvedi for all the love and affection they have showered upon their children. My parents are my true inspiration. I want to thank my brother Mr. Rajeev Chaturvedi and sister Mrs. Mamta Tripathi for being a great wall of support and motivation in my life. I am thankful to my brother-in-law Mr. Satish Tripathi and sister-in law Mrs. Megha Chaturvedi for their care and encouragement. Lastly, I want to express my deepest love for my nieces Vilina and Manavi for filling my life with their angelic presence.

ABSTRACT OF THE DISSERTATION

LEAKAGE TEMPERATURE DEPENDENCY AWARE REAL-TIME SCHEDULING FOR POWER AND THERMAL OPTIMIZATION

Vivek Chaturvedi

Florida International University, 2013

Miami, Florida

Professor Gang Quan, Major Professor

Catering to society's demand for high performance computing, billions of transistors are now integrated on IC chips to deliver unprecedented performances. With increasing transistor density, the power consumption/density is growing exponentially. The increasing power consumption directly translates to the high chip temperature, which not only raises the packaging/cooling costs, but also degrades the performance/reliability and life span of the computing systems. Moreover, high chip temperature also greatly increases the leakage power consumption, which is becoming more and more significant with the continuous scaling of the transistor size. As the semiconductor industry continues to evolve, power and thermal challenges have become the most critical challenges in the design of new generations of computing systems.

In this dissertation, we addressed the power/thermal issues from the system-level perspective. Specifically, we sought to employ real-time scheduling methods to optimize the power/thermal efficiency of the real-time computing systems, with leakage/temperature dependency taken into consideration. In our research, we first explored the fundamental principles on how to employ dynamic voltage scaling (DVS) techniques to reduce the peak operating temperature when running a real-time application on a single core platform. We further proposed a novel real-time scheduling method, "*M-Oscillations*" to reduce the peak temperature when scheduling a hard real-time periodic task set. We also developed three checking methods to guarantee the feasibility of a periodic real-time schedule under peak temperature constraint. We further extended our research from single core platform to multi-core platform. We investigated the energy estimation problem on the multi-core platforms and developed a light weight and accurate method to calculate the energy consumption for a given voltage schedule on a multi-core platform. Finally, we concluded the dissertation with elaborated discussions of future extensions of our research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

In merely past two decades, microprocessor performance has grown 1000-fold delivering unprecedented computing capabilities [24]. The advancement in microprocessor performance has largely be driven by the continuous scaling of the transistor feature size that facilitates exponential transistor integration capacity (doubling every 2 years, Moore's law). However, with increasing transistor density, the power density in the microprocessors has also grown exponentially, doubling every three years [118, 22]. Moreover, the exponentially increasing power consumption results in a dramatic increase in the chip operating temperature [118].

The exponential increase in the power consumption and the soaring chip temperature creates enormous challenges put hurdles not only to the design of the future computing systems but also to the associated economic (low cost computing) and green computing (environment protection) goals. New power/thermal aware methodologies and techniques to combat the power/thermal issues becomes essential. In this chapter, we first present the overview on the current technology trends, emphasizing on the motivation behind our research work. Next, we define our research problem and our contributions. Finally, we present the structure of this dissertation.

1.1 Semiconductor Industry and Moore's Law

The semiconductor industry has shown an evolutionary expansion in terms of sophisticated computing technologies and economic growth. Today, its a $2 trillion industry in US, impacting socioeconomic growth of human population around the world [88]. The remarkable growth of electronic systems has largely driven by so called "Moore's law" [91]. According to Moore's law, the number of transistors integrated on a single chip will double approximately every 18 months [91, 88, 46], delivering exponentially improved performance in new generations of computing systems. Moore's prediction of exponential increase in transistor density with every generation will continue to be so, atleast in the near future [88]. The attainment of integration density objective has been catalyzed by the foundation of transistor scaling laid by Dennard et al. in 1974 [43, 24]. For the past 50 years, with other technological advances (e.g. fabrication, architecture, micro-architecture, testing etc.), Moore's law coupled with transistor size scaling has resulted in consistent exponential performance gains [46].

Transistor scaling would not cause power consumption problem if the supply voltage could be scaled down accordingly. The idealistic theory of transistor size scaling rules that with every generation transistor dimensions should be reduced by 30% (0.7X) keeping electric fields constant, shrinking area by 50%, doubling transistor density and increasing performance by 40% (0.7X delay reduction, or 1.4X frequency increase). The electric field is kept constant by reducing supply voltage by 30% resulting in 50% power reduction per transistor [43, 24]. However, with the transistor design paradigm shifting to the deep sub-micron domain (DSM), the idealistic scaling theory ceases to hold valid anymore. Though the density of transistors' continues to increase exponentially for performance gains, the slow scaling of the supply voltage due to stringent threshold voltage constraint has resulted in the exponential increase in the power density of microprocessor systems.

As an example, in Figure 1.2, we can see the projected exponential scaling of the feature size from 45nm to 8nm during the course of 8 years. The frequency and supply voltage scaling (V_{dd}) on the other hand is expected to scale with slower pace in the future, particularly V_{dd} scaling getting flat after 2014 (node $<$ 20nm) [24, 46]. As a result, the power consumption has been increasingly exponentially with transistor density as illustrated in Figure 1.1.

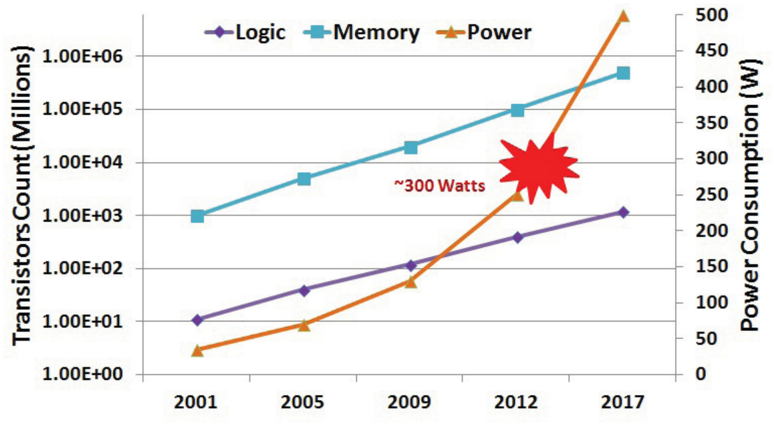


Figure 1.1: Trend Predicting Transistor Counts and Uncontrolled Power Consumption [24]

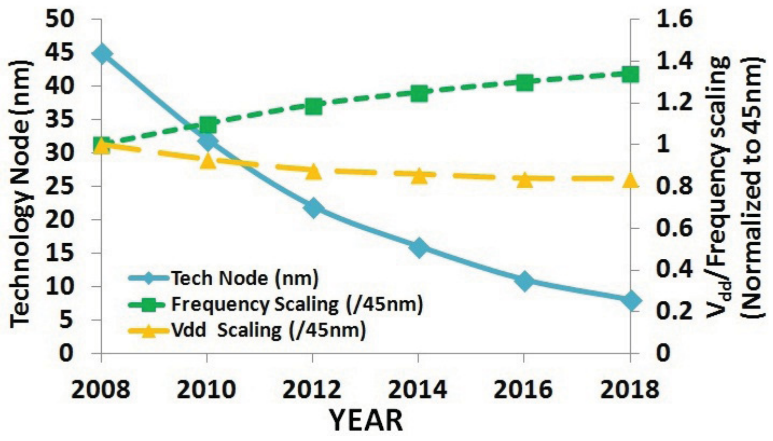


Figure 1.2: Technology Scaling Trends [24]

1.2 The Rising Power Consumption and Its Challenges

The tremendous increase in the transistor integration density has contributed to an exponential increase in power demands. As depicted in Figure 1.1, on a $300mm^2$ die, more than 100 billion transistors (logics + memory) are integrated today, seeking tremenodus power consumption demands (300 watts) in the near future [23]. The exponential increase of the power consumption brings two significant challenges in front of the designers of the electronic systems: (1) how to provide the enough supply, and (2) how to deal with the heat dissipated by the systems. These issues are critical for both battery-operated portable devices and high performance power-rich systems.

For the first challenge, computation devices with limited power sources have stringent constraints on power consumption. Recently times, there has been a tremendous shift in the market for personal computing, with a rapid and widespread demand for highly sophisticated portable/mobile devices like laptops, mobile phones, music players etc. For example, every four in ten Americans owns a portable MP3 player [116, 1] and mobile phones are the fastest growing electronic product ever [116, 45]. Portability in these devices put essential restrictions on the size, weight and power. Power is particulary important, as these portable devices largely depend upon the battery-life to deliver high performance. As the computation complexity is growing with a rapid rate demanding higher power/energy supply, the much slower growth in battery capacity (3 - 7%/year) poses a critical limitation in front of the designers [107, 118]. As the mobile devices are growing pervasive, energy efficiency is a critical design metric for these these energy-constrained architectures [116, 141].

Power consumption has also become increasingly critical for power-rich platforms such as data centers. Data centers and server systems play an important role in today's cyber infrastructure. Government organizations, hospitals, share markets, IT companies etc., all depend upon data centers for their daily business activities.

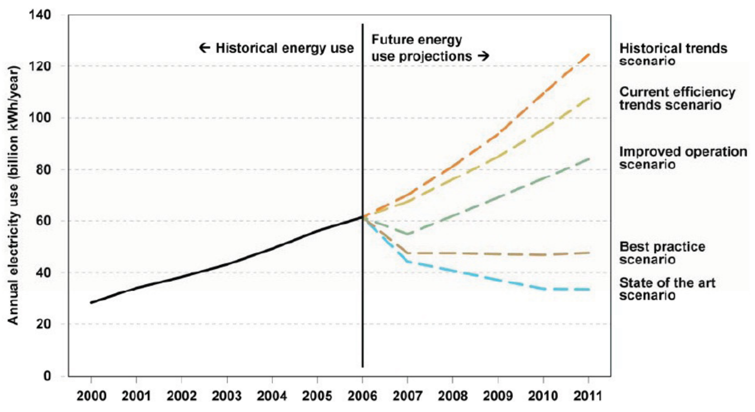


Figure 1.3: Projected Electricity in Data centers [7]

Data center power consumption accounts for 1.5 - 2% of the total electricity usage in the USA, costing nearly $4.5 billion [7]. The U.S. Environmental Protection Agency (EPA) predicts that energy consumption in data centers will exceed 100 billion kWh in 2011 [7](depicted in Figure 1.3), causing the federal governments electricity cost for servers and data centers to be nearly $740 million annually by 2011. Evidently, the need for power/energy efficient methods are of critical importance in contemporary and futuristic computing environment.

1.3 The Temperature Issues

A significant part of the power consumption consumed by the system is converted to heat. Therefore, the exponentially increasing power consumption results in the dramatic increase in chip temperature. Managing temperature in advanced microprocessors has become a severe challenge for computing system architects and designers.

The escalating heat has directly led to high cooling/packaging costs in microprocessors (1-3 dollar/watt [119]). With more compact devices and non-uniform heat distribution (hotspots) on the chips' surface, traditional air-cooling methods have become inadequate [98]. New cooling solutions like liquid cooling, micro-channel cooling etc [120, 110] are studied for cost-effective heat removal. The severity of thermal problems is highlighted by Intel's acknowledgement of hitting a "thermal wall" [90]. When studying the temperature management in data centers, the numbers are striking. As shown in Figure 1.5, the cooling cost in data centers has grown 400% in the last 10 years, and this is expected to rise with the same rate in the future. Moreover, it is also reported that for 1 watt of computing, half to one watt has to be consumed just for cooling [18, 25]. The expensive cooling methods in data centers are now a serious threat to economic problems, leading to the "economic meltdown of Moore's Law" [25].

High chip temperature not only increases the cooling/packaging costs, but also adversely affects the life-span and reliability of a device. The increase in chip temperature increases the rate of life-time fault processes, like electromigration, time-dependent dielectric breakdown, stress migration and thermal cycling [21, 20, 142]. Though there are several formulations of different type of fault processes, in general reliability can be modeled by using the Arrhenius equation [61, 136, 21], i.e. $MTF = MTF_0 e^{\frac{E_a}{K_b T}}$, where MTF is the mean-time-to-failure of a system, and T is the operating temperature. From this equation we can see that a device's mean-time-to-failure decreases exponentially with increasing operating temperature. Based on this, Yeh et.al [135] observed that a 10^oC rise in the temperature can result in 50% reduction of the system life span. Moreover, rising temperature has a negative impact on system reliability. For instance each 15^oC rise in the temperature can add approximately 10 -15% circuit delay [111]. This causes various timing anomalies and spurious transitions in digital systems.

High temperature also increases leakage power consumption substantially. With continuous scaling of semiconductor technology, the transistor device size has already entered the deep sub-micron era (feature size $<$ 90nm). As transistor size becomes smaller and smaller, the temperature leakage dependency becomes stronger and stronger. High temperature increases the leakage power consumption, which further contributes to the overall power consumption. This positive feedback loop between the overall power consumption and the temperature is a serious threat to design of modern computing systems. This is particularly true as the leakage power consumption has now become a significant contributor to the overall power consumption of the system. As shown in Figure 1.4, the leakage power consumption will increase dramatically in the near future. It is catching up and even surpassing the dynamic power consumption [65]. It is shown by Liao et al. [77] that when changing the temperature from 65^oC to 110^oC, the leakage power can increase as much as 38% for processors using the 65nm technology. Moreover, a 10^oC rise in temperature above 35^oC can result in an increase of 126% in the leakage current [111].

Furthermore, driven by the performance demands and energy constraints, processor architectures are evolving from 2D integration to 3D stacked integrated chips. In 3D designs multiple 2D chips are stacked vertically to achieve higher performance and energy efficiency due to reduced interconnects' lengths [19, 86]. However, in 3D stacked chips, the impact of increased temperature grows many folds due to strong thermal relation between vertically aligned neighbors. This strong thermal correlation elevates the thermal issues by creating more hot spots [147]. Particularly, as increasing temperature has strong relation with leakage power and reliability, it is shown that 30% variation in process parameters can result up to 20X increased leakage power [28].

From the above discussion, it is evident that there is an urgent need to develop new and efficient methodologies to address power/thermal issues in computing systems. In the next section, we present the problem which we address in this dissertation, followed by a brief description of our contributions.

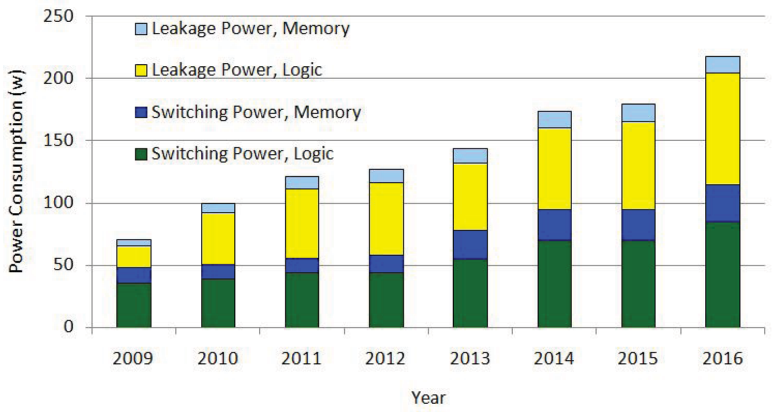


Figure 1.4: Power Consumption Trend [65]

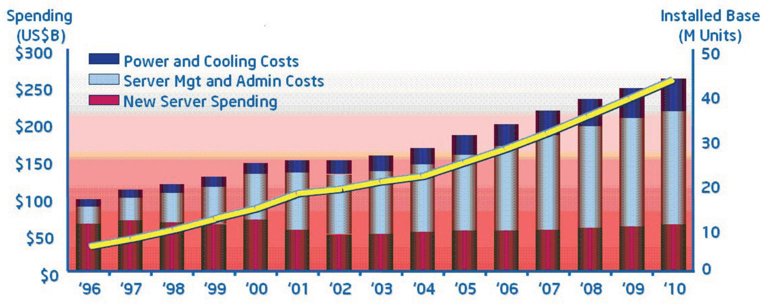


Figure 1.5: Projected Electricity in Data centers [7]

1.4 Research Goal and Contributions

The power/thermal challenges have grown critical and imposed severe threats to the realization of new generations of computing systems. The severity of the challenges requires development of new power/thermal aware design solutions at every design abstraction level, i.e. architecture, system, logic, circuit, device etc. In this dissertation, we address these issues from the system-level perspective. We want to study the real-time scheduling techniques coupled with resource management capabilities available in modern processors to optimize the power and the thermal efficiency of the real-time computing systems.

Through our research, we have developed several solutions to address the power/thermal issues, which are summarized as following:

1. We first explored fundamental principles on how to employ dynamic voltage scaling (DVS) to reduce the peak operating temperature. We find that, for a specific interval, a real-time schedule using the lowest constant speed is not necessarily the optimal choice any more in minimizing the peak temperature. We identify the scenarios when a schedule using two different speeds can outperform the one using the lowest constant speed. In addition, we find that, when scheduling a periodic task set, the constant speed schedule is still the optimal solution for minimizing the peak temperature when the temperature is at its *stable status*. We formulate our conclusions into several theorems with formal proofs.

2. Next, we studied the problem on how to minimize the peak temperature of a processor when executing a periodic task set. In our research, we developed a novel real-time scheduling method, "*M-Oscillations*" that can reduce the peak temperature when scheduling a hard real-time periodic tasks set. We formally proved the correctness of the proposed algorithm based on a processor model

that can effectively account for the leakage/temperature relationship. The proposed *M-Oscillations* scheduling method can reduce peak-temperature of the system up to 14^oC, improving feasibility of given tasks set by maximum 20%.

3. We also studied feasibility checking problem for real-time periodic task sets under the peak temperature constraint. We showed that the traditional scheduling approach, i.e. to repeat the schedule that is feasible through the range of one hyper-period, does not apply any more. We then developed new necessary and sufficient conditions to check the feasibility of real-time schedules.

4. We further extended our research from single-core processor to a multi-core platform. Different from the traditional numerical approach, we developed an analytical method to calculate the overall energy consumption rapidly and accurately. Our experiments show that the proposed method can achieve a speedup of two orders of magnitude compared with the numerical method, with a relative error of no more than 0.1%. Based on our light weight energy estimation method, we presented new energy minimization techniques based on different task allocation methods and compared their efficiency in minimizing the overall energy consumption of a multi-core system.

1.5 Structure of the Dissertation

This dissertation is organized as follows: We start by presenting pertinent background of our research work and related work in Chapter 2. Chapter 3 presents several fundamental theorems and properties in reducing peak temperature of a processor system. A novel real-time scheduling method, "*M-Oscillations*" that can reduce the peak temperature when scheduling a hard real-time periodic tasks set is introduced in Chapter 4. In Chapter 5, we present a set of feasibility checking methods for a hard real-time periodic tasks set to guarantee real-time constraints under maximum temperature constraints. In Chapter 6, we present a novel method to calculate the energy consumption efficiently and effectively for a given voltage schedule on a multi-core platform, with the leakage/temperature dependency taken into consideration. Finally, in Chapter 7, we conclude this dissertation and discuss the possible future work of this research.

CHAPTER 2

Background and Related Work

In this chapter, we present pertinent background of our research and related work. We begin our discussion with a general introduction to the basic concepts related to the real-time systems and computing. Next, we discuss the power consumption sources in the CMOS circuits, followed by a survey on power and thermal management techniques at different design abstraction levels. Then, we present a literature review on the existing power/thermal aware scheduling methods that are closely related to this dissertation.

2.1 Real-Time Systems and Computing

Real-time systems are pervasive. They are adapted and implemented in several domains of computing, like defense and space systems, communication systems, automotive systems, multimedia etc. Mostly hidden, real-time systems work in the heart of various computing platforms and perform important services for humankind. They reside in our cars controlling engine and brakes, in our TVs/games maintaining undisrupted entertainment, in health monitoring devices like blood pressure machine evaluating our fitness level, etc. With real-time systems finding utility in almost every electronic device around us and performing some very crucial tasks, the reliability and correctness of real-time systems becomes very important. It will be no exaggeration in identifying real-time computing and associated issues as the most important research area in the development of computing systems. In this section, we discuss some of the key concepts of real-time systems. We further discuss different problem areas and published solutions related to real-time scheduling.

2.1.1 Real-Time Systems

In a real-time system the correctness of the system behavior depends not only on the logical result of the computations, but also on the physical instant at which these results are produced. A real-time system has to respond to an externally generated stimuli within a specified time [81, 114]. In simple words, we can describe a real-time system as a system that has a deadline. The violation of timing constraints in real-time systems degrade the quality of service and in some cases may also result in catastrophical accidents [81, 114]. Real-time systems can be broadly categorized into hard real-time systems with hard timing constraint and soft real-time systems that have some tolerance for timing lateness and may respond with decreased service quality.

In general, a unit of work that is scheduled and completed by a computing system is called a job, and a set of related jobs jointly executing some specific system function is called as a task [81]. The job/task models used in real-time systems are very important in characterizing the type of real-time computing system. A task can be periodic or aperiodic.

Periodic vs Aperiodic: A periodic task is a sequence of jobs, with a minimum length intervals between release times of two consecutive jobs. The tasks are invoked at regular intervals following a deterministic pattern of time intervals. For example, in air traffic control(ATC) system, the status of each aircraft is monitored using active radars. These radars check the status periodically and update the ATC controller [81]. On the other hand, an aperiodic or sporadic task is a sequence of jobs with unknown release times. These tasks are invoked in irregular pattern and the inter-arrival times between consecutive jobs in such a task may vary widely. For instance, in a setting of radar surveillance system, the system should be responsive to operators command but not on the expense of task with hard deadline.

2.1.2 Real-Time Scheduling

In a real-time system the execution of every task requires computational and data resources. The real-time scheduling is concerned with the allocation and management of the resources to complete the assigned workload within the timing constraints. Given a task set with necessary timing information, available system resource and design constraints, the real-time scheduling answer the questions of when and where the task should be executed in order to satisfy timing constraints, at the same time other design metrics are optimized and never violated, e.g. energy, peak temperature etc.

Over the past several decades real-time scheduling has been studied extensively to address issues related to feasibility of task sets, performance optimization etc. for a span of combinations of processor architectures and task models. Based on different characteristics of applications, resource availability and system requirements, real-time scheduling methods can be classified in several ways:

1. Priority-driven vs Non-Priority : In priority driven real-time scheduling methods at any scheduling decision time, the jobs with the highest priorities are scheduled and executed on the available processors. Other commonly used names for this approach are greedy scheduling, list scheduling and work-conserving scheduling [81]. Some examples of priority-driven scheduling includes earliest deadline first (EDF) scheduling, rate monotonic scheduling(RMS) etc [79]. On the other hand in non-priority driven methods some policies like round-robin are used to determine if the task should start executing or not [121].

   Moreover, priority-driven algorithms can be further divided into fixed/static priority algorithms and dynamic priority algorithms. In fixed priority algorithms, priorities of tasks are assigned during compile time and they remain unchanged throughout the execution, e.g. RMS. A fixed-priority algorithm assigns the same priority to all the jobs in each task. In contrast, a dynamic-priority algorithm assigns different priorities to the individual jobs in each task. Hence the priority of the task with respect to that of the other tasks changes as jobs are released and completed [81], e.g. EDF algorithms.

2. Preemptive vs Non-Preemptive : If the execution of lower priority task is stopped or preempt for a higher priority task then the scheduling scheme is called as preemptive scheduling and non-preemptive otherwise. Preemptive or non-preemptive scheduling of tasks is possible with static and dynamic scheduling [10].

3. On-line vs Off-line : On-line algorithms makes scheduling decisions during runtime. The parameters of each job become known to the on-line scheduler only after the job is released. All priority-driven algorithms are considered on-line scheduling techniques. In off-line scheduling, scheduling decisions are made statically during compile time. This assumes parameters of all the tasks is known a priori and builds a static schedule based on this [79].

Furthermore, real-time scheduling methods can be classified based on type of underlying architecture, i.e. uniprocessor scheduling and multiprocessor scheduling. When addressing scheduling problems on uniprocessors, a scheduler needs to deal with when the task should be executed such that the entire workload is feasible. On uniprocessor scheduling, work done by Liu and Layland [79] is of special interest and great importance. In [79], Liu and layland proposed two priority-driven scheduling algorithms i.e. a dynamic-priority scheduling named earliest deadline first(EDF) and a fixed priority algorithm called rate monotonic scheduling (RMS). Both EDF and RMS have been used extensively in the research domain as the underlying scheduling policy for other design metrics optimization like energy minimization ([101, 149, 103]), schedulability/feasibility analysis([5, 100]) etc.

In multiprocessor scheduling, the problem is not only to determine when a given task executes but also where it executes. There are also issues related to availability of necessary resources at the processor at which a task is scheduled to execute, contention for communication across a network, etc. These issues make the problem substantially harder to solve. There have been extensive researches published on real-time scheduling for homogeneous/heterogenous multi-core systems [9, 8, 69, 74, 51, 47]. These scheduling algorithms can be largely categorized into three classes: the partitioned approach (e.g. [9]), the global (or non-partitioned) approach (e.g. [8]) and the semi-partitioned approach (e.g. [69, 74, 51, 47].

In the partitioned scheduling, each real-time task is assigned to a dedicated processor. All instances from the same task will be executed solely on that particular processor. In global scheduling, all jobs first enter a global queue, and thus each task can be potentially executed on any processor. The semi-partitioned algorithms are combination of previous two approaches, i.e. some tasks are assigned to a dedicated processor, while rest can migrate among available resources.

2.2 Power Minimization

The CMOS technology has served as the leading solution and will continue to be the primary solution for the foreseeable future in the design/fabrication of integrated circuits. As the goal of this dissertation is to address power/thermal issues in CMOS ICs, it is important to understand the basics of power dissipation in CMOS circuits.

2.2.1 Power Dissipation in CMOS ICs

There are mainly two sources of power consumption in a CMOS circuit, i.e. dynamic power consumption and static/leakage power consumption.

Dynamic Power: The dynamic power dissipation is associated with the switch-

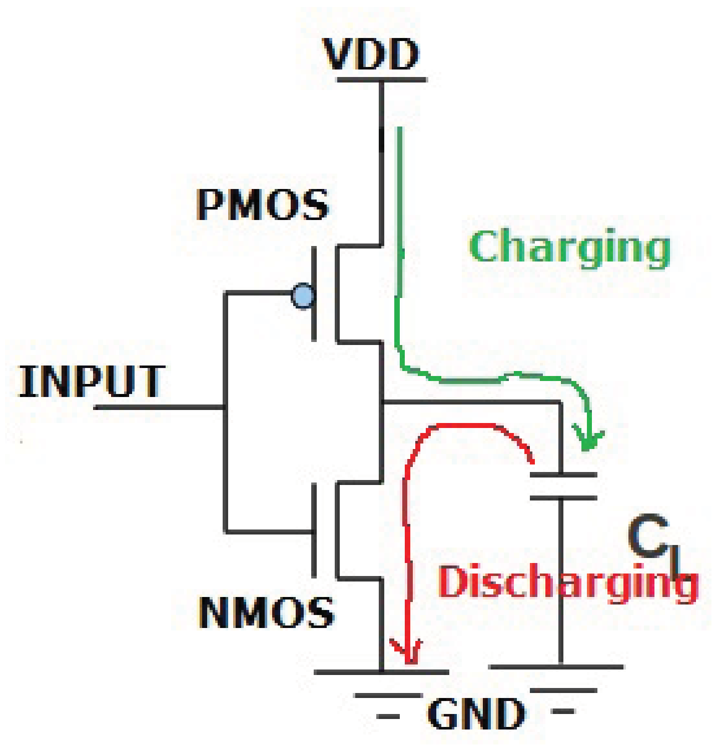


Figure 2.1: A CMOS Inverter Circuit [95]

ing of logic values in the circuit. This power component is essential to performing useful logic operations and occurs during the charging/discharging of the load capacitor in MOS transistors, as in Figure 2.1 [95, 107]. It can be formulated as [107]:

$$P_{dyn} = \alpha_{0->1} * C_L * V_{dd}^2 * f \tag{2.1}$$

Where $\alpha_{0->1}$ is the switching factor, V_{dd} represents the supply voltage, C_L is the load capacitance and f represents the clock frequency. Equation (2.1), implies the methods and underlying fundamental principles in dynamic power reduction, such as reducing activity ($\alpha_{0->1}$), reducing the circuit complexity (C_L), scaling the supply voltage(V_{dd}) and/or adjusting the circuit speed (f). We can also notice that the impact of scaling supply voltage to dynamic power can be significant as they share a quadratic relation.

Leakage power: The static power is associated with maintaining the logic values of internal circuit nodes between the switching events. This type of power consumption mostly comes from leakage power which occurs when the leakage current flows through diodes or transistors. Such a power dissipation does not contribute to any useful computation. Leakage power can be formulated as [107]:

$$P_{leak} = I_{leak} * V_{dd} \tag{2.2}$$

Where I_{leak} is the leakage current and V_{dd} is the supply voltage. In MOS transistors (PMOS/NMOS), there are three main sources of leakage current, as shown in Figure 2.2

1. Source/Drain Junction Leakage Current (I_D) : The junction leakage occurs from the source or drain to the substrate through the reverse-biased diodes when a transistor is OFF. The magnitude of the junction leakage current depends on the area of the drain diffusion and the leakage current density, which is in turn determined by the process technology [95, 107].

2. Gate Direct Tunneling Leakage Current (I_G) : The gate direct tunneling leakage flows from the gate through the leaky oxide insulation to the substrate. Its magnitude increases exponentially with the gate oxide thickness and supply voltage [95, 107].

3. Sub-Threshold Leakage Current (I_{SUB}) : The sub-threshold current is the drain-source current of an OFF transistor. This is due to the diffusion current of the minority carriers in the channel for a MOS device operating in the weak inversion mode (i.e., the sub-threshold region). The magnitude of the sub-threshold current is a function of the temperature, supply voltage, device size, and the process parameters out of which the threshold voltage plays a dominant

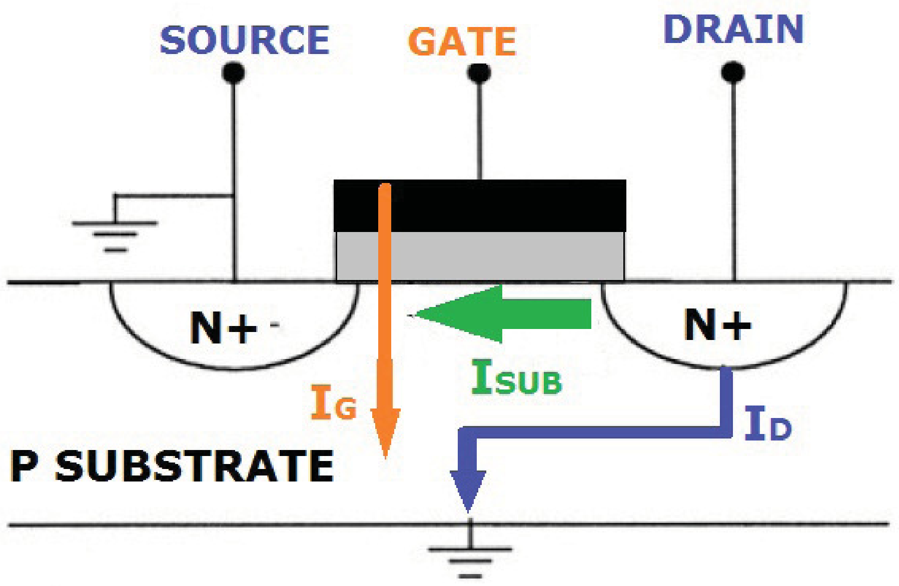


Figure 2.2: Types of Leakage Current in MOS [95]

role [95, 107].

It has been studied that the leakage current due to subthreshold conduction (I_{SUB} and gate leakage current (I_G) (see Figure 2.2) are the dominating component among different types of leakage current [107, 95]. Following what, Liao et al. established a complex relationship between the leakage current and the temperature and formulated the leakage current as: [77],

$$I_{leak} = I_s \cdot (\mathcal{A} \cdot T^2 \cdot e^{((\alpha \cdot V_{dd} + \beta)/T)} + \mathcal{B} \cdot e^{(\gamma \cdot V_{dd} + \delta)}), \tag{2.3}$$

where I_s is the leakage current at certain reference temperature and supply voltage, T is the operating temperature, V_{dd} is the supply voltage, $\mathcal{A}, \mathcal{B}, \alpha, \beta, \gamma, \delta$ are empirically determined technology constants. This model is used extensively in several research work in developing theoretical framework for system level analysis [35, 30, 66, 140, 106, 100]. This micro-architecture level modeling is very complex and unwieldily to be implemented for system level analysis. Following which, Liu et al. [84] demonstrated that within a temperature range using a piece-wise linear model can accurately estimate leakage power consumption (within 1% relative error). Moreover, for a tighter temperature range linear models can be fairly accurate. Based on this observation, a number of researches (such as [35, 48, 30]) adopted a simple temperature/leakage dependency model that assumes the leakage current changes linearly *only* with temperature. Quan et al. [104] introduced a more accurate and practical linear model that is sensitive to both temperature and supply voltage (with 5% relative error). A detailed comparative study of different leakage models is presented in [59].

2.2.2 Power Minimization Methodologies

As power consumption becoming more and more critical, research has been conducted at various design abstraction levels, i.e. architecture level, system level, logic level, circuit level and device level, targeting both dynamic power and leakage power consumptions.

Dynamic Power Minimization: To reduce the dynamic power, techniques that can help to reduce the factors formulated in Equation 2.1, namely capacitance, supply voltage, frequency, and switching activity, have been developed at different design abstraction levels:

- Dynamic Voltage and Frequency Scaling (DVFS/DVS): In general real-time systems are usually designed under the assumption of worst case execution time (WCET). Since tasks rarely execute up to their WCETs, there are good opportunities for power and energy savings. DVFS (or DVS) is a method to provide variable amount of energy for a task by scaling the operating voltage/frequency at run-time (V_{dd} and f in Equation 2.1). With dynamic power having convex relationship with supply voltage, this technique is one of the most effective dynamic power reduction methodology [125]. Most modern processors today are augmented with hardware blocks that allow changing the supply voltage dynamically [134, 107, 95, 101].

- Clock Gating : Clock-gating is a well known High level (architecture/ RTL) technique for reducing dynamic power consumption of the synchronous designs. Clock power is a major component of total power consumption (about 40%), which makes clock-gating a very important power saving technique. Several work on low power design using different variation of clock gating are available in literature [129, 89, 44]

- Logic Gate Restructuring : This is a logic-level strategy to attack the switching activity factor ($\alpha_{0->1}$ in Equation 2.1) by improving the topology of the logic network. The topology of a logic network can severely affect the power dissipation. For example, in Figure 2.3, both chain and tree structure give same output. However, the switching activity factor of chain structure is observed to be less than the tree structure for random inputs [107]. Besides logic gate restructuring, there are some other techniques like input reordering, path balancing methods etc. which are used for reducing the switching activity by either removing spurious transitions due to glitching/jitter or reducing delay in intermediate nodes through reordering the input signals [107].

- Transistor Sizing : Transistor sizing is a circuit level technique that target load capacitance (C_L in Equation 2.1) of CMOS gates to minimize dynamic power consumption. The rationale behind transistor sizing is that not every transistor is required to be large (fast) for performance. This is a complex method that involves caution and is implemented based on peformance/power tradeoff factors [107]. Several research work on developing algorithms for better

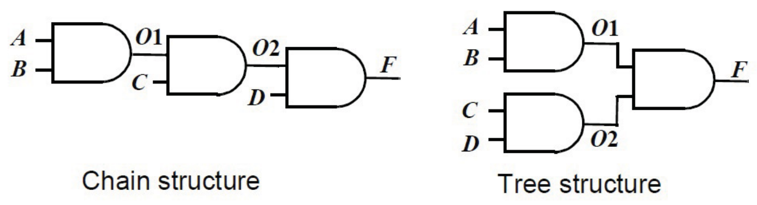


Figure 2.3: A Logic Restructuring Example [95]

transistor sizing decisions are available in the literature [108, 37].

Leakage Power Minimization: Leakage power dissipation is becoming substantial in the modern CMOS VLSI circuits. It severely impact the design of the future computing systems and is of vital importance. Due to the continuous scaling of device sizes (Moore's law), the voltage supply levels and threshold voltage of MOS transistors are also scaling. Based on the leakage model discussed above, we now present some of the effective leakage power minimization techniques

- Body Biasing: The leakage current can be reduced by reverse body biasing that increases the threshold voltage of transistors in STANDBY modes [112, 95]. Due to reverse biasing, a transistor increases its threshold voltage. This results in a decrease in the leakage current of the transistor. Because the threshold voltage changes with the square root of the reverse bias voltage [107], a large voltage may be necessary to get a small increase in the threshold voltage. As a result, this method becomes less effective as the supply voltage is scaled down.

- Power Gating : Power gating is an intuitive method to address leakage power dissipation during device off-mode. It is implemented using one PMOS transistor and one NMOS transistor in series with the transistors of each logic block to create a virtual ground and a virtual power supply as depicted in Figure 2.4. During the STANDBY state of the circuit, the extra transistor disconnect the gate from the ground, hence increasing threshold voltage. In

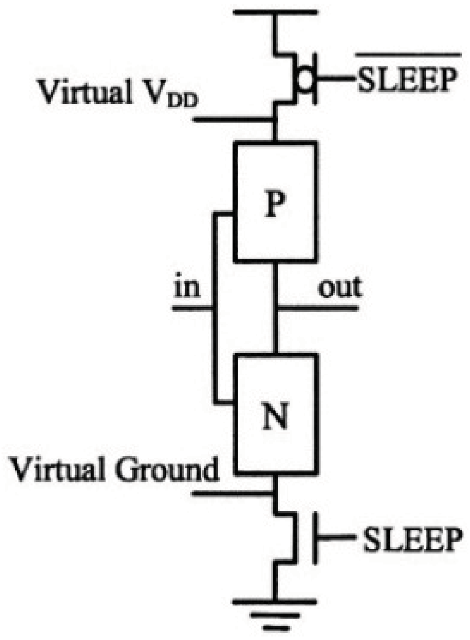


Figure 2.4: Power Gating Circuit [95]

practice, Dual CMOS or Multi-Threshold CMOS (MTCMOS) is used for power gating [95, 70, 12]. The implementation of this technique is non-trivial, specially for sequential circuits.

- Minimum Leakage Vector Method : The leakage current of a circuit is a strong function of its input values. In [6], Abdollahi et al. leveraged on this fact to reduce leakage current. They formulate the problem of finding the minimum leakage vector (MLV) using a series of Boolean Satisfiability problems. The solution vector was used to reduce leakage power by driving the circuit using calculated input vector during STANDBY mode [95, 49]. This method is inferior to power gating in terms of magnitude of leakage reduction, but is free from any implementation difficulties [95, 49].

As previously discussed, controlling chip temperature is also an effective method to reduce leakage power consumption.

2.3 Temperature Minimization

In managing thermal problems, thermal modeling is of critical importance to explore the large design space without expensive silicon prototypes [60]. Thermal models facilitate accurate characterization of thermal stress, temporal and spatial non-uniformities and application dependent behavior [119].

To study thermal-aware design techniques at the architectural-level, Skadron et al. [4, 119] proposed an accurate and fast architectural-level thermal model called "Hotspot". The thermal modeling is based on the well known duality between the heat transfer and the electrical current flow (as shown in Table 2.1),

Table 2.1: Duality Between Thermal and Electrical Quantities

Thermal Quantity	Electrical Quantity
Power consumption: P (W)	Current flow: I (A)
Temperature: T (^oC)	Voltage: V (V)
Thermal resistance: R $(^oC/W)$	Electrical resistance: R (Ω)
Thermal capacitance: C $(J/^oC)$	Electrical capacitance: C (F)

HotSpot models every power consuming smallest micro-architecture block of processor as a node in an RC network. In this way, each functional unit on the chip is represented by one or more nodes within the RC network 2.5. The HotSpot model includes active layer, thermal interface layer, heat spreader layer and heat sink layer, resulting in a three dimensional RC network. Based on this RC network and basic circuit laws, a system of differential equation is established.

HotSpot uses fourth order Runge-Kutta method to solve this set of differential equations to capture the thermal dynamics of microprocessors. The model has been validated using finite element simulation. HotSpot has a simple set of interfaces and hence can be integrated with most power-performance simulators [119]. The chief advantage of HotSpot is that it is compatible with the kinds of power/performance models used in the computer-architecture community, requiring no detailed design or synthesis description.

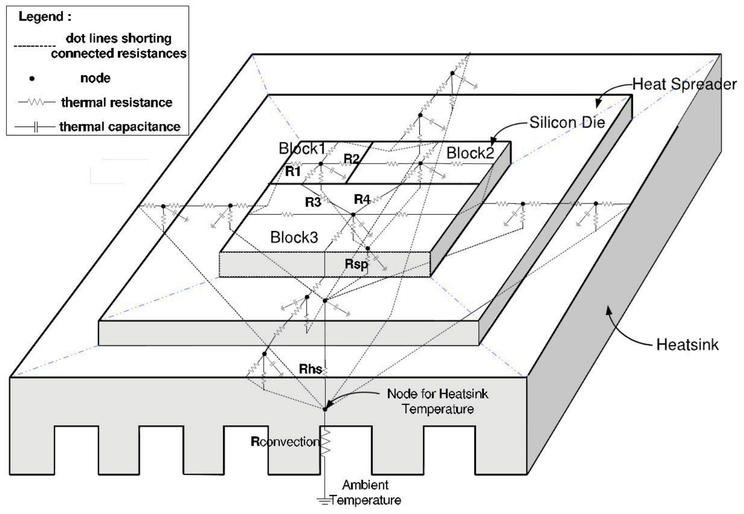


Figure 2.5: Using RC Network to Model a Processor's Heat Transfer [119]

Temperature Minimization Methodologies

Developing solutions for thermal related issues is not a straight forward effort, like what we see in the power minimization techniques. One interesting fundamental of the thermal issues is that even though technically temperature is a byproduct of power consumption, but thermal management problems are distinctly different from the power management issues [118]. In modern integrated circuits, heat dissipation is not uniform on the chip creating several hot spots or high power density points. This non-uniformity gets worse with increasing transistors density, non-uniform scaling of supply voltage and increasing leakage power consumption.

As a result, an effective low-power technique may have little or no effect on operating temperature of the chip [118, 77, 106]. In fact, sometimes low power techniques could result in the higher temperature due to smaller structures and limiting activity to a smaller area [60]. There is another one important difference between power management and thermal management solutions, i.e. power management techniques usually seek to reduce total chip power ignoring localized power densities, on the other hand temperature management methods control local hot spots [60].

The traditional way to protect ICs from worst case heat dissipation is to depend heavily on the rather expensive thermal package solutions (e.g. heat sink, fan, cold plates etc.). However, with rising packaging costs and meagre chances of worst case temperature to happen, this solution can be costly. Moreover, with processors evolving from single core to multi-core, conventional back-side heat removal strategies such as air-cooled heat sink etc. are insufficient to provide better cooling.

Therefore, better cooling solutions like inter-tier liquid cooling, micro-channel liquid cooling etc. are gaining lot of attentions for efficient heat removal. For instance, in [120], an energy efficient dynamic inter-tier liquid cooling method is proposed, that can achieve as much as 30% reduction in system level energy consumption. In addition, any applications that dissipate more heat than package capacity, should engage an alternative dynamic thermal management technique (DTM) [118].

Dynamic Thermal Management Approaches: In the recent years, dynamic thermal management(DTM) has emerged as an interesting and very effective thermal management scheme [119]. DTM allows an adaptive control on chip temperature by dynamically applying forced (e.g. clock throttling) heat reduction when the chip reaches a thermal emergency situation. It addresses the problem of higher thermal packaging costs by allowing designers to use thermal packaging for some lower temperature instead of high temperature due to peak power consumption.

A number of DTM approaches are proposed recently, such as task migration [78], global clock gating [52], fetch toggling [26] and decode throttling [119]. Out of which, task migration and clock gating are widely used in the temperature minimization.

2.4 Power/Thermal Aware Scheduling

The system-level scheduling techniques are very effective in addressing power/thermal issues in microprocessors. In past two decades power/thermal aware scheduling methods have been studied extensively for both single core (e.g. [134, 64, 97, 127] etc.) and multi-core architectures (e.g. [124, 36]etc).

2.4.1 Scheduling Solutions for Dynamic Power reduction

The early efforts in the real-time scheduling were focused on using DVS for reducing the then dominant dynamic power consumption (e.g. [134, 64, 97] etc.). For instance, addressing single processor architectures, Yao et al. [134] proposed an energy-optimal off-line scheduling method with continuous speed levels. Employing this greedy algorithm, Yao et al. proved that using lowest constant speed to complete the task results in the minimum energy consumption. Ishihara et al. [64] extended this work to discrete speed levels and proved that if this constant speed is not available then using the two neighboring speeds will be the next optimal solution. Pillai et.al [97] proposed a couple of on-line real-time DVS scheduling schemes that can guarantee deadlines of tasks saving significant energy consumption. Several other related approaches were proposed in [71, 101, 102, 115]. For example, Kim et al. [71] proposed an energy reduction technique for periodic tasks with EDF priority by improving slack time estimation. On the other hand, authors in [101, 102, 115], addressed fixed-priority scheduling exploiting slack time for power reduction using power down mode and DVS.

The energy-efficient scheduling in multi-core systems are often NP-hard, as a result several heuristics and approximation techniques are studied to minimize dynamic energy consumption on multi-core platforms [143, 33, 50, 148]. In [50] Gruian used simulated annealing algorithm for task to core allocation and developed two-stage energy minimization method when scheduling tasks with inter-dependencies. Zhang et al. [143] proposed a technique that finds the optimal task allocation for energy minimization by exhaustively checking all possible permutations of task allocation running on the minimum speed that could guarantee real-time constraints for each task. A heuristic algorithm with largest task first strategy is proposed in [33]. Moreover, exploiting slack time sharing on multiple processors, Zhu et al. [148] developed a power-aware scheduling method for energy minimization such that timeliness of the schedule is never violated.

2.4.2 Scheduling for Overall Power Reduction with Constant Leakage

Reducing dynamic power consumption is important in solving energy efficiency problems, however, ignoring leakage power consumption leaves the above mentioned scheduling techniques ineffective in contemporary and modern integrated circuits. Acknowledging the growing dominance of leakage power dissipation in overall energy consumption, several research works are published addressing overall energy minimization in both single core (e.g. [63, 67, 94, 103] etc.) and multi-core platforms(e.g. [131, 34, 42] etc). For example, Irani et al. [63] proposed an off-line approximation algorithm with a competitive ratio of 3 on a DVS processor with continuous speed levels. The approach is based on energy-optimal greedy algorithm in [134]. They also proposed an on-line algorithm with constant approximation factor.

In a separate approach, authors in [94] and [103] proposed efficient method that merge idle intervals due in a DVS schedule by delaying the execution of task instances, so that the processor shutdown overhead can be reduced and the overall energy performance can be improved. When addressing overall energy minimization on multi-core systems, Langen et al. [42] presented leakage-aware heuristics to determine tradeoff between DVS, shut down and optimal number of active processors. In a related work, Xu et al. [131] proposed an algorithm to estimate the system load and corresponding optimal number of processing nodes for energy efficiency in cluster systems. Unlike [42], processors in this work are assumed to have discrete speed levels. However, all these techniques for overall energy reduction modeled leakage power as a constant value and therefore their efficiency in energy reduction is strictly limited, abandoning energy-efficiency to continue as a grave challenge.

2.4.3 Thermal Aware Scheduling

In addition, along with technology scaling thermal challenges are growing aggressively, making temperature a critical constraint to achieve desirable performance. Moreover, as previously mentioned that power/energy aware solutions are incompetent to address thermal issues single handedly, an increasing number of researches have been published on thermal-aware scheduling for both single and multiple processor platforms(e.g. [14, 105, 139, 127, 109] etc). For example, in [14] and [32] authors try to identify the upper bound of the maximal temperature when executing real-time tasks on a single processor. These techniques cannot guarantee that real-time tasks can still meet deadlines when the maximal temperature is given. Some others (e.g. [14, 38, 29, 109]) intend to minimize the peak temperature or guarantee the given maximal temperature constraints when scheduling a job set or a single copy of a task graph. For example, Chantem et al. [29] proposed an MILP-based solution to minimize the peak temperature when executing a task graph.

Recently, significant amount of 3D thermal-conscious OS-level work has been published [40, 146, 76, 83, 87, 122]. In [146], an online thermal-aware task scheduling technique for high performance with reduced peak temperature is proposed. The methodology involves temperature-balance among 'super-cores' combined with DVFS to avoid thermal emergencies. An online rotation task scheduling policy to reduce peak temperature on the 3D multi-core system is proposed in [76]. Liu et al. [83] proposed a thermal-aware job allocation technique, which always assigns hot jobs to core near heat sink, for fast heat removal so that the peak temperature of the system can be reduced.

2.4.4 Power/Thermal Aware Scheduling with Temperature Sensitive Leakage Power

Several power/thermal aware scheduling methods have been proposed addressing various design metrics for both single core and multi-core systems incorporating leakage/temperature dependency into system models. For example, in [104, 100], authors formulated several feasibility checking methods when scheduling a periodic tasks set on a uniprocessor. In [31, 73, 139, 66, 29, 48] etc. several thermal-aware scheduling techniques were proposed to reduce peak temperature on both single core and multi-core processors. In [66] Jayaseelan et al. proposed different iterative job sequencing techniques to identify the peak temperature and to reduce the peak temperature of the system. Kumar et. al [73] proposed a stop-n-go approach to reduce the peak temperature for task with data dependencies. They distribute the slack time between jobs such that temperature can be minimized and there is no make-span violation. In [35], Chen et al. proposed proactive scheduling methods to minimize response time under the thermal constraint, and also the reduction of the peak temperature under timing constraints. Fisher et al. [48] proposed method to minimize the peak temperature in a homogeneous multi-core system, by deriving an ideally preferred speed for each core in a global task scheduling environment.

As we know escalating energy consumption is a serious problem, that worsen with leakage and temperature interplay, authors in [57, 132, 17, 53, 85] proposed techniques to minimize energy consumption under peak temperature constraints. For instance, Huang et al. [57] derived a closed-form energy calculation equation based on which they further proposed an energy minimization scheduling method for periodic task sets. In [132], Yang et al. presented a procedure to find the optimal pattern of schedule with the minimum energy consumption at the steady state. Hanumaiah et al. [53] formulated energy minimization as a quasiconcave optimization problem and employed DVFS, task migration and cooling methods to optimize the objective function on a multi-processor system. On the other hand, Liu et.al [85] developed a thermal-constrained energy optimization procedure to minimize system energy consumption under peak temperature constraint.

Performance optimization has always been the first class design objective specially with temperature constraints. In [30, 58, 80, 54] and [55] authors proposed techniques to improve processor performance by maximizing the throughput or minimizing the makespan time under peak temperature limitations. Chantem et al. [30] proposed to run real-time tasks by frequently switching between the two speeds which are neighboring to the constant speed whose stable temperature is the given peak temperature. In [58], Huang et al. proposed two approaches to maximize the throughput for a periodic real-time system under the given peak temperature constraint, one for processor with simple active and sleep mode and the other for more complicated processors with DVFS capabilities. When maximizing throughput on multi-core platforms, authors in [55] address task-to-core allocation over migration intervals and voltage speed scaling within migration intervals as a separate problem and translated task-to-core allocation in *MILP*-formulation. To reduce the non-linearity of the function several assumptions were made to solve the optimization functions.

Including leakage power model, Zhu et al. [147] proposed a run-time thermal management technique that exploits the heterogeneity of execution cores and workload. They proposed a proactive hardware-OS assisted technique to achieve efficient thermal management. A closely related adaptive approach that balances the temperature on the cores,' Adapt3D' was proposed by Coskun et al. [40]. Coskun et al. used a second order polynomial temperature dependence leakage model in their method and proposed a thermal-aware job scheduling technique that uses a thermal history of neighbors in job allocation decisions.

2.5 Conclusions

In this chapter, we discussed several relevant background of our research and related work. We presented a general introduction to the basics of real-time systems, task models and real-time scheduling. Next, we discussed power/thermal modeling and several effective ways to address power/thermal challenges from different design abstraction levels. We then, presented a literature review on the existing power/thermal aware scheduling methods that are closely related to this dissertation.

In this dissertation, our goal is to develop power/thermal aware scheduling solutions for both single core and multi-core, hard real-time system with deterministic workload under various design constraints. In the next four chapters, i.e. Chapters 3, 4, 5 and 6 we present our contributions. We will conclude this dissertation in Chapter 7.

CHAPTER 3

Fundamentals of Leakage Aware Real-Time DVS Scheduling for Peak Temperature Minimization

In the previous chapter, we discussed some of the background information that is closely related to our research work. We now present a detail discussion on our first contribution. In this chapter, we study the fundamental principles on how to employ dynamic voltage scaling (DVS) to reduce the peak operating temperature. Our goal is to formulate guidelines and principles that can be used to make effective decisions when applying DVS for thermal management.

Since the thermal management problem is closely related to the power reduction problem, we started our research by investigating how effective the basic principles for energy reduction can be, when applied for dynamic thermal management. We began with the two well-known principles ([134, 64]), for dynamic energy reduction,

- *Principle 1*: Using the lowest constant speed leads to the schedule that consumes the minimum dynamic energy;
- *Principle 2*: If a single lowest constant speed is not available, then using two closest neighboring speeds is the optimal solution in dynamic energy reduction.

The question then becomes: when considering the complex relationship among the leakage power, the temperature, and the supply voltage, is it still true that a real-time schedule employing the lowest constant speed will lead to the lowest peak temperature within the scheduling interval? We find that, for a specific workload and interval, the schedule that uses the lowest constant speed is not necessarily optimal anymore in reducing the peak temperature. We identify the scenarios when a schedule that uses two different speeds may in fact lead to lower peak temperature. We also find that principles similar to the two listed above do exist to minimize the peak temperature during the temperature *stable status* when scheduling a periodic task. We formulate our observations into several theorems with formal proofs. The significance of this chapter is that it uncovers a number of fundamental principles in the development of effective DVS techniques for thermal aware computing.

The rest of the Chapter is organized as follows. Section 3.1 introduces the related work. System models used in this chapter are defined in Section 3.2. Section 3.3 presents our empirical results to motivate our research. Fundamental principles are formulated and proved in Section 3.4 and 3.5. Section 3.6 concludes the chapter.

3.1 Related Work

Closely related to our work, there are many thermal aware scheduling researches that seek to reduce the maximum operating temperature for a computing system. For example, Bansal et al. [14] modeled the cooling behavior of a device using first-order approximation to manage the temperature and energy of the system. Zhang et al. [139] proposed performance optimization by latency minimization under thermal constraints. Chantem et al. [30] proposed a dynamic work maximization technique by using DVFS with non-negligible transition overheads and under the temperature constraint. In [66] Jayaseelan et al. proposed different iterative job sequencing techniques to identify the peak temperature and to reduce the peak temperature of the system. Chaturvedi et al. developed [31] a so-called "*M-Oscillations*" scheduling method to minimize the peak temperature for a periodic task set. In their approach they used two-neighboring speeds to oscillate alternately to reduce the peak temperature of the system. Liu et al. [82] proposed to reduce the temperature of the system by properly sequencing hot and cool jobs and allocating slack time to hot jobs based on the duration of execution. Kumar et. al [73], proposed a stop-n-go approach to reduce the peak temperature for task with data dependencies. They distribute the slack time between jobs such that temperature can be minimized and there is no make-span violation. In [72], Kumar et al. developed a novel online technique that uses shapers to insert idle-time between the task set to reduce the peak temperature of the system. They modeled the task set by a resource-based curve on a single speed processor.

One distinct difference between our research and the existing researches is the way we deal with the leakage/temperature dependency. Some of the existing work totally ignore the leakage power or the leakage/temperature dependency(e.g. [72, 73, 17, 139]). As discussed before, a thermal aware scheduling technique becomes out of sync with the current IC technology in the deep submicron domain without taking the leakage/temperaure dependency into considerations. As the leakage power becomes more prominent, the heat generated by the processor can dramatically increase the leakage power and thus the overall power consumption. At the same time, the increased overall power consumption will in turn drive the temperature to an even higher level. Therefore, to build an appropriate model that can effectively handle the sensitive relationship between leakage/temperature, is the key to success when developing thermal aware scheduling techniques.

One way to deal with the leakage/temperature dependency is to incorporate the complex circuit level leakage model into system analysis ([56, 137]). For instance, He et al. [56] and Yuan et al. [137] studied how to reduce the leakage power at the system level. Yuan et al. [137] introduced an off-line and an on-line scheduling algorithm that take into account the leakage/temperature interactions when scheduling a set of soft real-time jobs.

Another common approach to address the leakage/temperature dependency that is adopted by existing work (such as [35, 30, 17, 11, 140, 96, 73, 72]) is to assume that the *leakage power* changes linearly or quadratically with temperature. Since leakage current changes super linearly with temperature [84], this model works well if the supply voltage do not change. Otherwise, as evidenced by the empirical results in Huang et al. [59], this model can lead to large discrepancies in either power consumption or peak temperature calculation in a DVS system.

3.2 System Model Definitions

In this section we define the system models used in this chapter, which include the task model, the processor model, the power model, and the thermal model.

3.2.1 Task Model

We consider two real-time application scenarios. In the first case, we assume that a number of real-time tasks within total execution cycles of c start at time 0 must be finished within the interval of $[0, p]$. Since all tasks have the same deadline, for simplicity, we assume there is only one task with execution cycle of c and deadline of p. In the second case, we further assume that this task is periodic with period of p.

3.2.2 Processor Model

The processor that we consider can run in different modes, with each mode being characterized by a pair of parameters (v_i, f_i), where v_i is the supply voltage and f_i is the working frequency in mode i. Even though the circuit delay changes with the circuit temperature dynamically, as given by equation (5.3) [77],

$$f_i = \frac{1}{t_d} \propto \frac{(v_i - v_t)^\mu}{v_i T^\eta}, \tag{3.1}$$

where v_t is the threshold voltage, t_d is the circuit delay, and μ and η are technology-related constants, we assume that the processor working frequency in each mode is fixed, and is the one that can accommodate the peak temperature (i.e. by assigning the peak temperature in equation (5.3) across the entire chip). Let f_{max} be the largest f_i among different modes. We can normalize the processor working frequency with f_{max} and get the normalized processor speed for each mode. In what follows, unless otherwise specified, we use the term processor speed or working frequency interchangeably.

3.2.3 Power Model

The power consumption of the processor consists of the dynamic power P_{dyn} and the leakage power P_{leak}. P_{leak} changes with both temperature and supply voltage. Specifically, the leakage current for a single transistor I_{leak} can be formulated as follows [77]:

$$I_{leak} = I_s \cdot (\mathcal{A} \cdot T^2 \cdot e^{((\alpha \cdot V_{dd} + \beta)/T)} + \mathcal{B} \cdot e^{(\gamma \cdot V_{dd} + \delta)}) \tag{3.2}$$

where I_s is the leakage current at certain reference temperature and supply voltage, T is the temperature, $\mathcal{A}, \mathcal{B}, \alpha, \beta, \gamma, \delta$ are empirically determined constants. Liu et al. [84] found that using linear approximation method to model the leakage current/temperature dependence can achieve reasonable accuracy with greatly simplified leakage power model. In our work, we adopt this method and simplify the leakage power model as follows:

$$P_{leak}(k) = C_0(k)v_k + C_1 T, \tag{3.3}$$

where $k = 0, \cdots, m-1$ represents m different processor modes. $C_0(k)$ and C_1 are constants that can be obtained by curve fitting for a particular processor under its operating environment conditions. In section 3.3, we use empirical study results to justify the appropriateness of this leakage model.

Therefore the total power consumption at processor mode k can be formulated as

$$P(k) = C_0(k)v_k + C_1 \cdot T + C_2 v_k^3. \tag{3.4}$$

3.2.4 Thermal Model

We use the lumped RC model similar to Skadron et al. [118] to capture the thermal phenomena of the processor. Specifically, assuming a fixed ambient temperature (T_{amb}), let $T(t)$ denote the temperature at time t. Then we have

$$RC\frac{dT(t)}{dt} + T(t) - RP(t) = T_{amb}, \tag{3.5}$$

where $P(t)$ denotes the power consumption (in $Watt$) at time t, and R, C denote the thermal resistance ($^oC/W$), and thermal capacitance (in $J/^oC$). We can then scale T such that T_{amb} is zero and get

$$\frac{dT(t)}{dt} = aP(t) - bT(t), \tag{3.6}$$

where $a = 1/C$ and $b = 1/RC$.

Putting Equation 3.4 in Equation 5.1, we formulate the temperature dynamics of the processor such that when a processor running in mode k for interval $[t_0, t_e]$, let the starting temperature be T_0, then solving equation (5.2), the ending temperature can be formulated as below:

From equation (3.4) and (5.2),

$$\begin{aligned} T_e &= \frac{A(k)}{B} + (T_0 - \frac{A(k)}{B})e^{-B(t_e - t_0)} \\ &= G(k) + (T_0 - G(k))e^{-B(t_e - t_0)}. \end{aligned} \tag{3.7}$$

where

$$A(k) = a(C_0(k)v_k + C_2v_k^3), \tag{3.8}$$

$$B = b - aC_1, \tag{3.9}$$

and

$$G(k) = \frac{A(k)}{B}. \tag{3.10}$$

Equation (3.7)-(3.10) play a critical role in our analytical analysis. For the sake of simplicity, we use A_k and G_k to denote $A(k)$ and $G(k)$, respectively, if there is no confusion.

3.3 The Empirical Studies

Considering that the leakage power changes with both temperature and supply voltage, is the constant speed schedule still the optimal choice in minimizing the peak temperature within a specific interval? Before we draw any conclusions, we first launched a number of empirical studies to obtain some intuitions. We also conducted several experiments to justify our leakage power model as well as to study the thermal characteristics of the processor based on our thermal model. For ease of our presentation, we first define several representative real-time schedules as follows.

Definition 3.3.1. *The* constant-speed schedule $\hat{S}(S_c)$ *within an interval* $[t_0, t_p]$ *is the schedule that employs the lowest constant processor speed* S_c *to complete the workload within the interval.*

Definition 3.3.2. *A* two-speed schedule $\hat{S}(S_1, S_2)$ *within an interval* $[t_0, t_p]$ *is the schedule that uses the two different speeds* S_1 *and* S_2 *with at-most two transitions between the speed levels to complete the workload within the interval.*

We further define four different types of two-speed schedules.

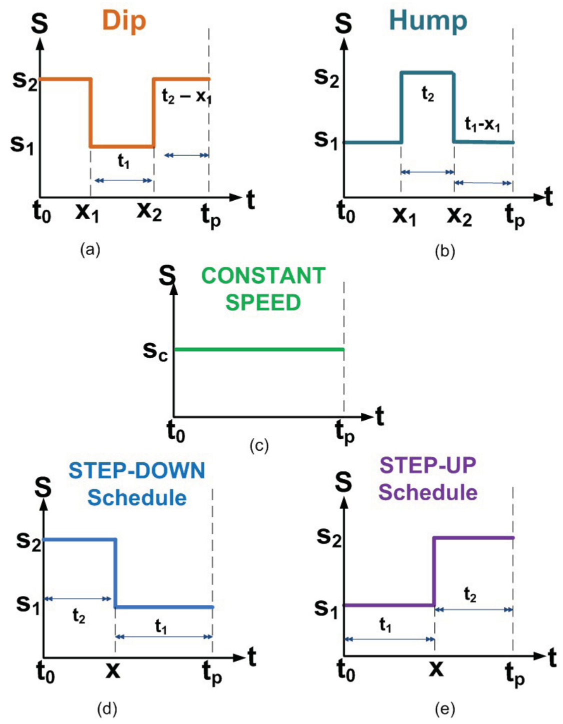


Figure 3.1: Different Speed Schedules: (a) The Dip Schedule; (b) The Hump Schedule; (c) The Constant Schedule; (d) The Step-Down Schedule; (e) The Step-Up Schedule.

Definition 3.3.3. *A* dip schedule $\hat{S}(S_1, S_2)$ *within an interval* $[t_0, t_p]$ *is a two-speed schedule that uses* S_1 *during the interval* $[x_1, x_2](t_0 \leq x_1 < x_2 \leq t_p)$*, and* S_2 *in the rest of the intervals, with* $S_1 < S_2$*.*

Definition 3.3.4. *A* hump schedule $\hat{S}(S_1, S_2)$ *within an interval* $[t_0, t_p]$ *is a two-speed schedule that uses* S_2 *during the interval* $[x_1, x_2](t_0 \leq x_1 < x_2 \leq t_p)$*, and* S_1 *in the rest of the intervals, with* $S_1 < S_2$*.*

Definition 3.3.5. *A* step-up schedule $\hat{S}(S_1, S_2)$ *within an interval* $[t_0, t_p]$ *is a two-speed schedule that uses* S_1 *during the interval* $[t_0, x]$*, and uses* S_2 *in interval* $[x, t_p]$ *with* $S_1 < S_2$*.*

Definition 3.3.6. *A* step-down schedule $\hat{S}(S_1, S_2)$ *within an interval* $[t_0, t_p]$ *is a two-speed schedule that uses* S_2 *during the interval* $[t_0, x]$*, and uses* S_1 *in the interval* $[x, t_p]$ *with* $S_1 < S_2$*.*

Figure 3.1 shows an example of different speed schedules defined above. Note that, according to Definition 2, once the two speeds and total workload within the interval are defined, the total length of the interval to run processor with each speed is also defined (e.g. t_1 and t_2 in Figure 3.1). In what follows, we present several empirical results that help to obtain some intuitions on the applicability—in the context of peak temperature minimization—of the two power reduction principles mentioned above.

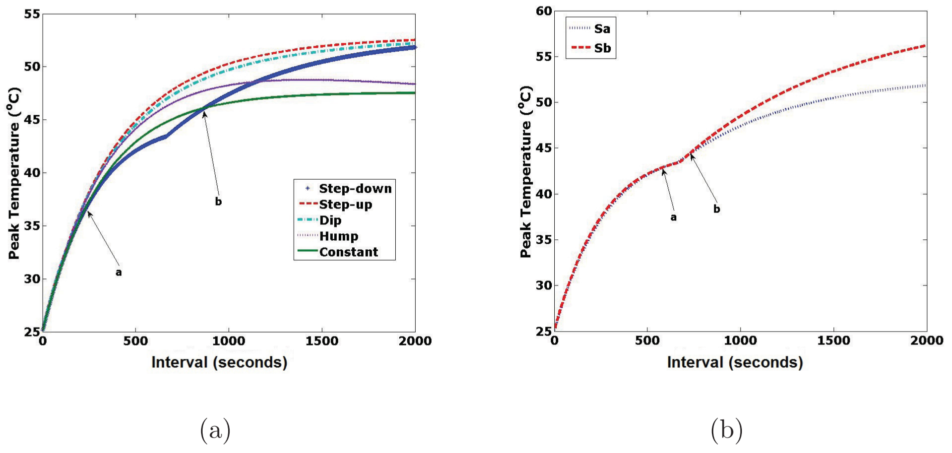


Figure 3.2: The Peak Temperatures by Different Schedules Within a Given Interval

Empirical Study 1 First, we wanted to verify if two principles listed before are still valid in terms of the peak temperature minimization *for a given interval*. We constructed our processor similar to the one shown in [104, 100], based on the 65nm technology and with the conventional air cooling option of $Rth = 0.8^oC/W$, $Cth = 340J/^oC$ [119]. We assumed that the processor can run on four active modes i.e. 0.95V, 1.0V, 1.05V and 1.10V, and one shut-down mode. The corresponding frequency was calculated using equation (5.3). The values of the remaining parameters are taken from [100]. The ambient temperature was set to 25^oC.

We selected three available processor speeds with corresponding supply voltages as 0.95V, 1.0V, and 1.05V respectively. Five different types of schedules, i.e. the constant-speed schedule, the step-down, the step-up schedule, the dip schedule and the hump schedule were constructed that run the same length and complete same workload. For dip and hump schedule, the value of x_1 (see Figure 3.1) was randomly selected. We then varied the interval length from 10 to 2000 seconds to get different schedules. For each test case, the highest temperature within the corresponding interval by each schedule was collected and plotted in Figure 3.2(a).

As can be seen from Figure 3.2(a), while the maximum temperature using the step-up schedule is always higher than that by the constant-speed schedule, the peak temperature by the other two-speed schedules can in fact be lower sometimes. For instance, when the period is equal to 700 seconds, the peak temperature of the step-down, the constant, the hump, the dip and the step-up speed schedules are 37.84^oC, 43.95^oC, 45.07^oC, 46.5^oC and 47.32^oC, respectively. This result clearly contradicts the conclusion that the constant-speed schedule is the optimal schedule in terms of minimizing the peak temperature within a given interval. In the meantime, we can also observe that the peak temperature by the step-up schedule is indeed consistently higher than that of other types of schedules.

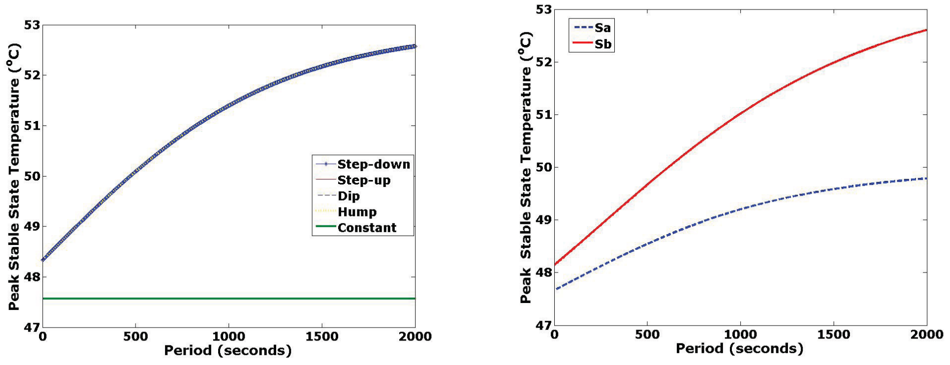
We further studied if using the two closest neighboring speeds is the best choice in terms of peak temperature reduction within a given interval. Two step-down speed schedules Sa and Sb were constructed. Sa uses the speed corresponding to supply voltages 0.95V & 1.05V for low and high speed respectively, and Sb uses speeds corresponding to 0.95V & 1.10V. Both Sa and Sb run the same length and complete same workload. As done before, we then varied the interval length from 10 to 2000 seconds to get different schedules. For each test case, the highest temperature within the corresponding interval by each schedule was collected and plotted in Figure 3.2(b). When comparing their peak temperatures, as shown in Figure 3.2(b), the step-down schedule Sa is not necessarily always better than Sb. When the period is set to 700 seconds, the peak temperature for Sa is 43.65^oC, while for Sb it is 43.56^oC. This result seems to also imply that the second principle is not valid either in terms of the peak temperature reduction.

Empirical Study 2 We next want to verify if the two principles listed before can be used to minimize the peak temperature *when the processor temperature reaches its stable status*.

We constructed the five different schedules, i.e., the constant-speed schedule, the step-down, the dip schedule, the hump schedule and the step-up schedule the same as above, and ran each schedule not one time but periodically until the temperature became stable and do not seem to change anymore. We then varied the periods, collected the maximum temperature for each case and plotted in Figure 3.3(a). From this figure, we can clearly see that the constant-speed schedule always lead to the lowest peak temperature in our experiment, and the peak temperatures by remaining two-speed schedules eventually become the same.

In addition, when we constructed the two step-down schedules Sa and Sb as above and ran them periodically. From Figure 3.3(b), we can see that the step-down schedule Sa using the two closest neighboring speeds is always better than Sb.

These empirical results suggest that using a constant speed in a schedule, or using the neighboring speeds when the constant speed is not available, is still the best way to minimize the peak temperature when the temperature reaches its *stable status*. In the next few sections, we formulate these findings into theorems and prove them formally. Before we introduce the theorems and their proofs, we first use empirical results to justify our leakage model and to establish some useful thermal characteristics of our processor model.



(a) The peak temperature by the constant speed is consistently lower than others

(b) The peak temperature by the schedule using neighboring speeds is consistently lower than the one using non-neighboring speeds

Figure 3.3: Peak Temperatures at the Stable Status by Different Schedules

Empirical Study 3 In Section 3.2, we have introduced a simplified linear leakage power model to model the relationship of the leakage, the temperature, and the supply voltages. One immediate question is how accurate this model is? In this empirical study, we use an existing processor model drawn from the existing literature [77] to study this problem. The processor model is the same as used in Empirical study 1 & 2 with conventional air cooling. However, in this study, we assumed that the processor can run on 15 active modes i.e. 0.60V to 1.3V, with step size of 0.05V and one shut-down mode. The corresponding frequency was calculated using equation (5.3). The values of the remaining empirical and technology parameters are taken from [100] and [118]. The ambient temperature was set to 25^oC and we assume the processor's starting temperature is the same as the ambient temperature. Based on this processor model, we compared several existing leakage models.

- The *Actual* leakage model [77]: $P_{leak}(k) = I_{leak}v_k$ where I_{leak} is defined in equation (3.2).

- The simple *Linear* leakage model (e.g. [35, 30, 66, 140]): $P_{leak}(k) = C_0 + C_1 T$ with C_0, C_1 being constants;
- LK_{TV} model [104]: $P_{leak}(k) = I_{leak}(k) v_k$ and $I_{leak}(k) = C_0 + C_1 T$.
- LK_T model: the model defined in section 3.2.

Based on the experimental settings stated above, Figure 3.4 depicts the leakage power consumptions under different supply voltages and temperatures according to different leakage models. As we can see from Figure 3.4, when the supply voltage varies, the leakage power consumption varies dramatically. For instance, if we consider the *Actual* leakage model at 50^oC, the leakage power becomes almost double when supply level changes from 0.95V to 1.1V. Therefore, the simple *Linear* leakage model can only be used when the supply voltage cannot be changed. Otherwise, large discrepancies between the actual leakage power consumption (e.g. the results according to the *Actual* leakage model) and the estimated one with this model may occur. On the other hand, we can see that both LT_{TV} and LK_T match the actual leakage power consumption well, with relative errors under 4% in our study. These results clearly show that LK_T model is a leakage model with very good accuracy[1].

[1]According to our empirical results, the LT_{TV} model is more accurate than LK_T model. However, we are not able to formally prove all theorems in this chapter based on model LT_{TV}.

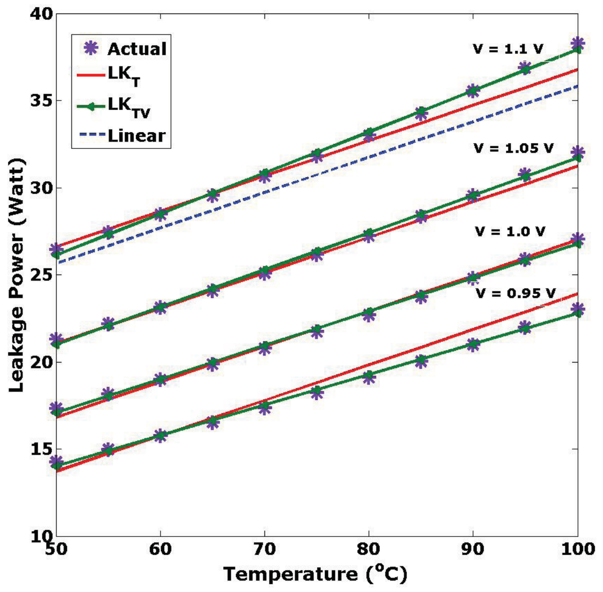


Figure 3.4: Leakage Power Consumptions Calculated using Different Leakage Models under Different Temperatures and Supply Voltages

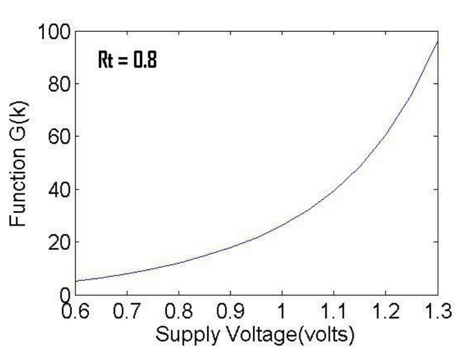
Furthermore, in order to conduct analytical analysis based on temperature dynamics in equation (3.7), it is highly desirable that the characteristics of G_k is known. However, since G_k is determined essentially by the curve-fitting constants C_0 and C_1, it is difficult to analytically study its properties. Therefore we study its attributes empirically.

Figure 3.3 plots the characteristics of the function $G(k)$ under different operating conditions i.e. (a) conventional air cooling option, (b) water spray cooling. As illustrated in Figures 3.5(a) and 3.5(b), we can clearly see that, for both cooling conditions, the function G_k is a positive, monotonic increasing, and convex function of the supply voltage. Also, from equation (5.24), we can see that it is necessary that $B > 0$, or the temperature will run away otherwise. Therefore, in what follows, unless otherwise specified, we assume that

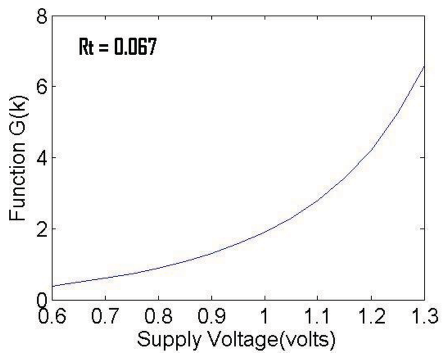
- G_k (or $G(v_k)$) is a *positive*, *monotonic increasing*, and *convex function* of k (or

v_k), respectively;

- $B > 0$.



(a) Conventional Air Cooling



(b) Water Spray Cooling

Our empirical results presented above reveal some strong and interesting findings. In the following sections, we formulate these findings into theorems and formally prove them.

3.4 Peak Temperature Minimization Within a Specified Interval

The empirical study 1 discussed in previous section shows that a constant speed schedule is not the optimal method for the peak temperature reduction within a specified interval. Then the question becomes what the optimal schedule is. To answer this question, in what follows, we formulate several theorems from our empirical study and prove them analytically. These theorems provide us with some insights and guiding principles when developing better DVS schedules for peak temperature minimization within a given interval. Specifically, Theorem 1 characterizes the peak temperature obtained using the step-up schedule within a given interval.

Theorem 3.4.1. *Given two processor speeds S_1 and S_2 with $S_1 < S_2$ and a hard real-time job J, the step-up schedule ($\hat{S}(S1, S2)$) has the highest peak temperature among* *all two-speed schedules within the same interval if the initial temperature $T_0 < G_1$. Where G_k is defined in equation (3.10).*

Proof: We first compare the peak temperature between the step-up and step-down schedules as shown in Figure 3.1(d) and (e). Let T_u be the peak temperature of step-up schedule, which always occurs at the end of the interval [31]. Let t_1 and t_2 be the interval length that the processor runs at speed S_1 and S_2, respectively. From Figure 3.1(e) and based on equations (3.7)-(3.10), we have

$$T_u = G_2(1 - e^{-Bt_2}) + G_1(1 - e^{-Bt_1})e^{-Bt_2} + T_0 e^{-B(t_2+t_1)} \tag{3.11}$$

According to Figure 3.1(d), we can see that, when using the step-down schedule, the peak temperature will occur either at point x or at t_p. Therefore to prove this theorem we need to consider two cases:

- *Case 1: The peak temperature of the step-down schedule appears at point t_p.*

  With the starting temperature T_0, the temperature of the step-down schedule T_d at t_p is given by

$$T_d = G_1(1 - e^{-Bt_1}) + G_2(1 - e^{-Bt_2})e^{-Bt_1} + T_0 e^{-B(t_1+t_2)} \tag{3.12}$$

  To show that $T_u > T_d$, by canceling $T_0 e^{-B(t_1+t_2)}$ from both equation (3.12) and (3.11), we have

$$\begin{aligned} &G_2(1 - e^{-Bt_2}) + G_1(1 - e^{-Bt_1})e^{-Bt_2} > \\ &\quad G_1(1 - e^{-Bt_1}) + G_2(1 - e^{-Bt_2})e^{-Bt_1}. \end{aligned} \tag{3.13}$$

  or

$$G_2 > G_1 \tag{3.14}$$

As G_i is a monotonically increasing function, equation (3.14) is true. Hence, we proved that the step-up schedule results in a higher peak temperature than the step-down schedule when its peak temperature appears at point t_p.

- *Case 2: The peak temperature of step-down schedule T_d appears at point x.*

  The temperature at point x by step-down speed schedule can be formulated as

$$T_d = G_2(1 - e^{-Bt_2}) + T_0 e^{-Bt_2}. \tag{3.15}$$

  To show that $T_u > T_d$, based on equation (3.15) and (3.11), we only need to show

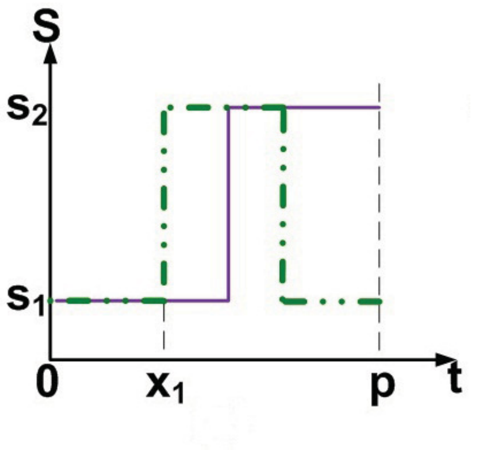
$$\begin{aligned} G_2(1 - e^{-Bt_2}) + G_1(1 - e^{-Bt_1})e^{-Bt_2} + T_0 e^{-B(t_1+t_2)} > \\ G_2(1 - e^{-Bt_2}) + T_0 e^{-Bt_2}. \end{aligned} \tag{3.16}$$

  or

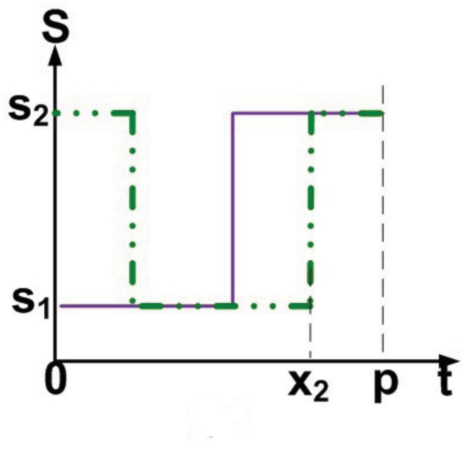
$$(G_1 - T_0)e^{-Bt_2} > (G_1 - T_0)e^{-B(t_1+t_2)} \tag{3.17}$$

  As $T_0 < G_1$, equation (3.17) is true. Hence we proved that the step-up schedule always results in higher peak temperature than step-down schedule when its peak temperature appears at point x.

From above, we can conclude that the step-up schedule always result in a higher peak temperature compared to the step-down schedule within a given interval for $T_0 < G_1$.



(c) The Peak Temperature by the Hump and Step-Up Schedule



(d) The Peak Temperature by the Dip and Step-Up Schedule

Figure 3.5: Peak Temperature

We now compare the step-up schedule with other types of two-speed schedules. We first compare it with the hump schedule as shown in Figure 3.5(c). Let T_x be the temperature at $t = x_1$. Based on equation (3.7), we have

$$T_x = G_1 + (T_0 - G_1)e^{-Bx}. \tag{3.18}$$

Since $T_0 < G_1$, we have $T_x < G_1$. Moreover, in Figure 3.5(c), let $T_u(x_1)$ and $T_f(x_1)$ denote the peak temperature within the interval $[x_1, p]$ by the step-up schedule and the hump schedule, respectively. Since $T_x < G_1$, and from the first part of the proof we know that the step-up schedule always incurs a higher peak temperature than that of a step-down schedule, we immediately prove that $T_f(x) \leq T_u(x)$. Similar conclusions can be proved for the dip schedule shown in Figure 3.5(d). □

Theorem 3.4.1 helps to identify the two-speed schedule that potentially leads to the highest peak temperature. It would be interesting if we can also identify the schedule that potentially leads to the lowest peak temperature. Theorem 3.4.2 can be used for this purpose.

Theorem 3.4.2. *Given two processor speeds S_1 and S_2 with $S_1 < S_2$ and a hard real-time job J, the step-down schedule ($\hat{S}(S1, S2)$) has the lowest peak temperature among all two-speed schedules within the same interval if the initial temperature $T_0 < min(G_1, (G_1 - G_2)e^{Bx} + G_2)$, where x is the length of the interval using S_2. where B and G_k are defined in equation (5.24) and (3.10), respectively.*

Proof: Theorem 3.4.1 already proves that the peak temperature of the step-up schedule is always higher than the step-down schedule under the same given condition, so we only need to compare the peak temperature between the step-down schedule with that by hump and dip schedules.

Consider Figure 3.6(a). Both the step-down schedule and the hump schedule use the same speed between x_2 and p. Also the initial temperature $T_0 < G_1$. Therefore, according to Theorem 3.4.1, we conclude that the peak temperature by the step down schedule is lower than that by the hump schedule.

Now consider Figure 3.6(b). Let the temperature at $t = x_1$ be T_{x1}. Then based on equation (3.7), we have

$$T_{x1} = G_2 + (T_0 - G_2)e^{-Bx1}. \tag{3.19}$$

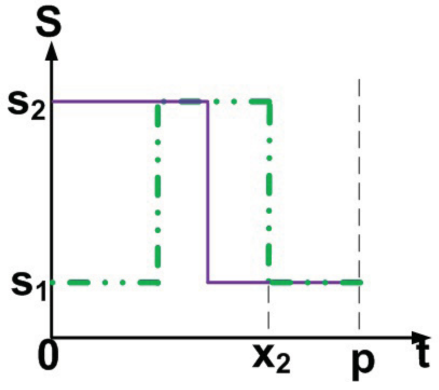
At the same time, since $T_0 < (G_1 - G_2)e^{Bx} + G_2$, we have

$$T_{x1} < G_2 + (G_1 - G_2)e^{B(x-x1)}. \tag{3.20}$$

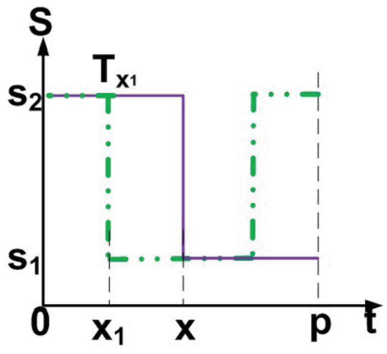
Since $G_1 < G_2$ and $e^{B(x-x1)} > 1$, we have

$$T_{x1} < G_2 + (G_1 - G_2) = G_1. \tag{3.21}$$

Therefore, according to Theorem 3.4.1, we conclude that the peak temperature by the step-down schedule is lower than that by the dip schedule. □



(a) The Peak Temperature by the Hump and Step-Down Schedule



(b) The Peak Temperature by the Dip and Step-Down Schedule

Figure 3.6: Peak Temperature

Furthermore, our empirical study shows that the conclusion that the constant-speed schedule is the optimal choice in terms of peak temperature minimization within a given interval is not true anymore. Even though our empirical results show that in most cases the constant-speed schedule is a better choice, it can be inferior to a step-down schedule sometimes. In Theorem 3.4.3, we formulate this conclusion and present the conditions when a constant-speed schedule becomes inferior to a step-down schedule in terms of peak temperature reduction.

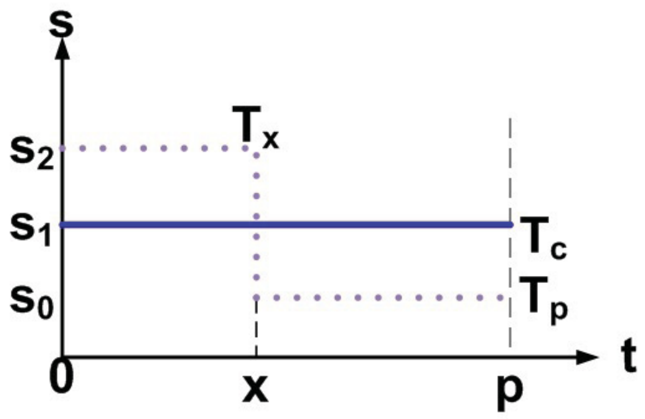


Figure 3.7: The Constant-Speed Schedule and a Step-Down Schedule within a Given Interval.

Theorem 3.4.3. *Given a constant-speed schedule $\hat{S}(S_1)$ and a step-down schedule $\hat{S}(S_0, S_2)$ for a hard real-time job J. Assuming $T_0 < min(G_1, (G_1 - G_2)e^{Bx} + G_2)$ and $S_0 < S_1 < S_2$. Let $T_m(\hat{S}(S_1))$ and $T_m(\hat{S}(S_0, S_2))$ be the peak temperature by $\hat{S}(S_1)$ and $\hat{S}(S_0, S_2)$ within the interval [0,p], respectively. Then,*

$$T_m(\hat{S}(S_1)) > T_m(\hat{S}(S_0, S_2)) \tag{3.22}$$

if and only if

- $\frac{1}{B}ln(\frac{G_2-T_0}{G_2-G_0}) < x < \frac{1}{B}ln(\frac{G_2-T_0}{G_2-G_1(1-e^{-Bp})-T_0e^{-Bp}})$*; or,*
- $x < min(\frac{1}{B}ln(\frac{G_2-T_0}{G_2-G_0}), p - \frac{1}{B}ln(\frac{G_2-G_0}{(G_1-G_0)+(G_2-G_1)e^{-Bp}}))$.

where $S_1 p = S_2 x + S_0(p - x)$, and B, G_k are defined in equation (5.23) and (3.10), respectively

Proof: From Figure 3.7, let T_x and T_p denote the temperatures of the step-down schedule at $t = x$ and $t = p$, respectively. Let T_c be the peak temperature of the constant-speed schedule. Based on equation (3.7), we have

$$T_x = G_2(1 - e^{-Bx}) + T_0 e^{-Bx}. \tag{3.23}$$

$$T_p = G_0(1 - e^{-B(p-x)}) + G_2(1 - e^{-Bx})e^{-B(p-x)} + T_0 e^{-Bp} \tag{3.24}$$

$$T_c = G_1(1 - e^{-Bp}) + T_0 e^{-Bp} \tag{3.25}$$

Note that the peak temperature of the step-down schedule must be either T_x or T_p. Then equation (3.22) becomes true only when

- $T_x < T_c$ when $T_x > T_p$, or
- $T_p < T_c$ when $T_p > T_x$

Replace T_x, T_c and T_p with equation (3.23) to (3.25) and solve for x, we can prove the theorem. □

As implied by Theorem 3.4.3, neither the constant speed schedule nor any particular two-speed schedule is always the optimal schedule to minimize the peak temperature within a given interval. On the other hand, however, Theorem 3.4.2 and Theorem 3.4.3 help to identify the optimal schedules that can potentially lead to the optimal DVS schedule to minimize the peak temperature within an interval. Both Theorem 3.4.2 and Theorem 3.4.3 target a single real-time job. Since most real-time tasks are repetitive in nature, it is highly desirable we can explore some fundamental principles for the periodic tasks as well.

3.5 Peak Temperature Minimization at the Stable State

In this section, we extend our research from a single real-time job to a periodic real-time task. When running a real-time task set periodically, unless the processor temperature "runs away" [77], the processor temperature is eventually stabled. The *stable status* is defined as below.

Definition 3.5.1. *[104]When running a periodic task with period p, the temperature at the processor is called to be* stable *if for a given threshold, i.e.* $0 < \varepsilon << 1$,

$$|T((n+1)p) - T(np)| < \varepsilon, \tag{3.26}$$

where $n \geq 0, n \in Z$, and $T(t)$ is the temperature at t.

Similarly, we want to investigate the validity of applying *Principle* 1 and 2 in the context of minimizing peak temperature when scheduling a periodic task set.

We first present two theorems that act as the basis in formulating the key principles of peak temperature minimization when the processor temperature becomes stable.

Theorem 3.5.2. *Given a hard real-time periodic task τ, the maximum temperature when the processor temperature reaches its* stable status *does not depend upon the initial temperature.*

Proof: Let us consider a step-down speed schedule shown in Figure 3.1, where S_2 and S_1 denotes the high speed and low speed. From Figure 3.1, t_1 and t_2 denotes the duration of S_1 and S_2 in the first period.

Based on equation (3.7), the temperature at $t = x$ and $t = t_p$ can be formulated as

$$T_x = G_2 + (T_0 - G_2)e^{-Bt_2}, \quad T_{t_p} = G_1 + (T_x - G_1)e^{-Bt_1}$$

where B and G_k are defined in equation (5.24) and (3.10), respectively.

From [104], the maximal temperature at the stable state temperature can be formulated as

$$T_{max} = max(T_x^{\infty}, T_{tp}^{\infty})$$

where,

$$T_x^\infty = \frac{G_2(1-e^{-Bt_2})}{1-e^{-Bt_2}} = G_2 \tag{3.27}$$

$$T_{tp}^\infty = \frac{G_1(1-e^{-Bt_1}) + G_2(1-e^{-Bt_2})e^{-Bt_1}}{1-e^{-B(t_1+t_2)}} \tag{3.28}$$

As can be seen from equation (3.27) and (3.28), no matter if the maximal temperature occur at t=x or t=t_p, it does not depend upon the initial temperature T_0. Similar conclusion can be achieved using other speed schedules. $\square$

Based on Theorem 3.5.2, we can show that the maximum peak temperatures at the stable state with any periodic two-speed schedule are the same.

Theorem 3.5.3. *Given a real-time periodic task τ and two processor speeds, then the maximal temperature at the* stable status *with any two-speed schedule using the same two speeds are the same.*

Proof: Consider a periodic step-up speed schedule shown in Figure 3.8. From [104], we can calculate the stable state temperature or the peak temperature as:

$$T_{max} = \frac{G_2(1-e^{-Bt_2}) + G_1(1-e^{-Bt_1})e^{-Bt_2})}{1-K} \tag{3.29}$$

where t_1 and t_2 denotes duration of low speed and high speed. B and G_i are defined in equation (5.24) and (3.10), respectively and

$$K = e^{-B(t_1+t_2)}.$$

From Figure 3.8 we can see that the step-down schedule shown is same as the periodic step-up schedule but with an initial temperature T_a. Similarly, all other periodic two-speed schedules can be viewed as the step-up schedule with an initial shift. Therefore their peak temperatures are equivalent to the one with the periodic step-up schedule with a different initial temperature. Since Theorem 3.5.2, already proves that the stable temperature does not depend on the starting temperature, the conclusion is proved.

□

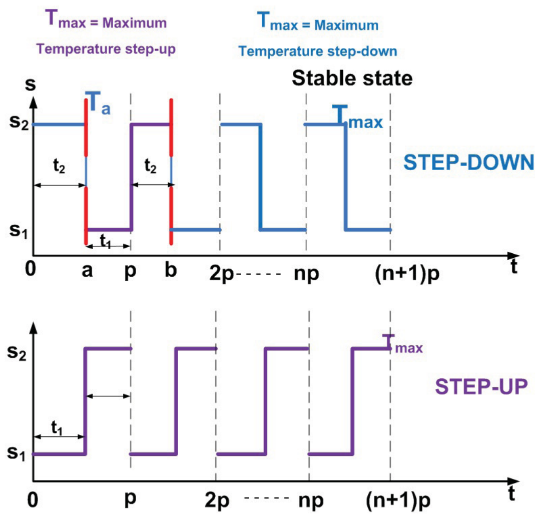


Figure 3.8: Stable Temperature for Step-Down and Step-Up Schedule

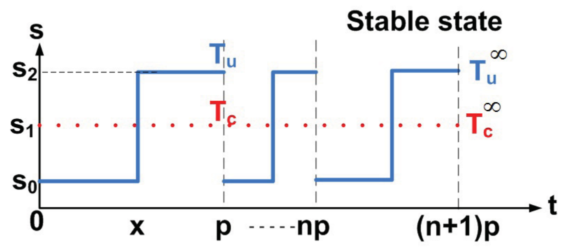


Figure 3.9: Stable Temperature for Step-Up and Constant-Speed Schedule

Based on the conclusions from Theorem 3.5.2 and 3.5.3, we can now formulate an important theorem for the problem of scheduling hard real-time periodic task, with the goal of peak temperature minimization.

Theorem 3.5.4. *Given a real-time periodic task τ, the maximum temperature at the stable state is minimized when running τ using the lowest constant-speed.*

Proof: From Theorem 3.5.3, we know that at *stable status* all the two-speed schedules result in the same peak temperature. Therefore, to prove this theorem we will compare constant-speed schedule with any two-speed schedule. Let T_c^∞ and T_u^∞ denote the maximum stable temperature for the constant-speed (S_1) and the step-up schedule ($S_0 < S_2$) respectively (Figure 3.9).

Without loss of generality, we can assume $p = 1$. Also from the conclusion of Theorem 3.5.2, we know the stable temperature does not depend on the initial temperature, therefore we assume the initial temperature to be zero. Then we have

$$T_c^\infty = \frac{G_1(1-e^{-B})}{1-e^{-B}} = G_1 \tag{3.30}$$

$$T_u^\infty = \frac{G_2(1-e^{-B(1-x)}) + G_0(1-e^{-Bx})e^{-B(1-x)}}{1-e^{-(B(1-x)+Bx)}} \tag{3.31}$$

To show that $T_c^\infty \leq T_u^\infty$, we only need to show that

$$G_1 \leq kG_0 + (1-k)G_2, \tag{3.32}$$

where

$$k = \frac{e^{-B(1-x)} - e^{-B}}{1-e^{-B}}, 1-k = \frac{1-e^{-B(1-x)}}{1-e^{-B}}. \tag{3.33}$$

Since

$$S_1 = S_0x + S_2(1-x), \tag{3.34}$$

and $B > 0$ and G_i is a convex function, we have

$$G_1 \leq xG_0 + (1-x)G_2. \tag{3.35}$$

Therefore, to show that equation (3.32) holds, we only need to show that

$$xG_0 + (1-x)G_2 \leq kG_0 + (1-k)G_2, \tag{3.36}$$

or

$$(G_0 - G_2)(x-k) \leq 0. \tag{3.37}$$

As $G_0 \leq G_2$, we only need to prove that

$$x \geq k = 1 - \frac{1-e^{-B(1-x)}}{1-e^{-B}}. \tag{3.38}$$

Or, equivalently,

$$\frac{1-e^{-B(1-x)}}{1-e^{-B}} \geq 1-x. \tag{3.39}$$

Now consider function

$$F(z) = \frac{1-e^{-Bz}}{1-e^{-B}} - z. \tag{3.40}$$

with $0 \leq z \leq 1$. We can readily show that function $F(z)$ is a concave function since $F''(z) < 0$. Note that the curve $F(z)$ passes two points, i.e. $(0,0)$ and $(1,0)$, as $F(0) = 0$ and $F(1) = 0$. Let $H(z)$ be the line that crosses these two points. Since $F(z)$ is concave, we have $F(z) \geq H(z) \geq 0$ for $0 \leq z \leq 1$.

We therefore prove that the constant speed schedule always outperforms a step-up periodic schedule in minimizing the peak temperature when the temperature reaches the *stable status*. In Theorem 3.5.3, we have already proved that at *stable status* the peak temperature of step-up and any other two-speed are the same. Hence we can immediately conclude that at the stable state the constant-speed outperforms any two-speed schedule in peak temperature reduction. □

Moreover, as constant-speed schedule is not always available and multiple speeds have to be used. In that case, we establish a new principle similar to *Principle 2*.

Theorem 3.5.5. *If a* two-speed schedule *is used for a hard real-time periodic task, then the one that uses the two closest neighboring speeds minimizes the maximum temperature at the stable state .*

Proof: Consider interval $[0, p]$ and step-up schedules $\hat{S}_1(s_1, s_4)$ and $\hat{S}_2(s_2, s_3)$ shown in Figure3.10. Without loss of generality, we assume $s1 \leq s2 < s3 \leq s4$. Let $T(\hat{S})$ represent the maximal temperature at the *stable status* with schedule $\hat{S}$. We want to show that $T(\hat{S}(s_2, s_3)) \leq T(\hat{S}(s_1, s_4))$. Let the speed change occur at x in $\hat{S}_2(s_2, s_3)$. Consider another schedule $\hat{S}_3(s_2, s_4)$ and let its speed change at x'. Then we have $x \leq x'$. Note that $\hat{S}_2$ and $\hat{S}_3$ complete the same workload within interval $[0, x]$ with the same speed, but $\hat{S}_2$ uses a constant speed to complete the rest of the interval and $\hat{S}_3$ uses two different speeds. From Theorem 3.5.4, we can immediately conclude that $T(\hat{S}(s_2, s_3)) \leq T(\hat{S}(s_2, s_4))$. Similarly, we can prove that $T(\hat{S}(s_2, s_4)) \leq T(\hat{S}(s_1, s_4))$. Therefore, $T(\hat{S}(s_2, s_3)) \leq T(\hat{S}(s_1, s_4))$. □

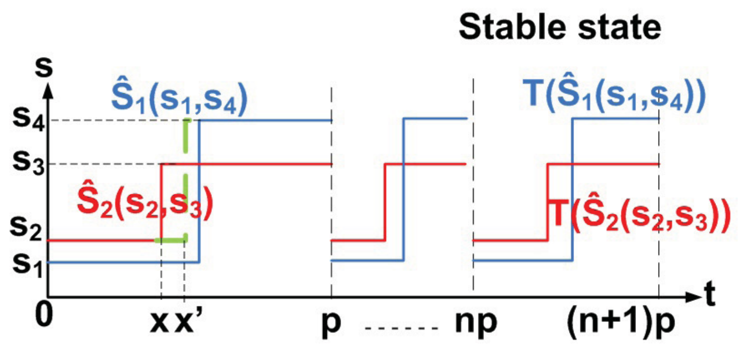


Figure 3.10: Step-Up Schedules $\hat{S}_2(s_2, s_3)$ and $\hat{S}_3(s_2, s_4)$ for a Real-Time Periodic Task.

Note that, even though Theorem 3.5.4 and Theorem 3.5.5 look very similar to the two basic principles that have been widely used for dynamic energy reduction, it does not necessarily imply that the existing energy reduction techniques can be readily applied for the purpose of peak temperature minimization. On the other hand, Theorem 3.4.1 to Theorem 3.5.5 present some fundamental guidelines in the development of new DVS schedule techniques that can minimize the peak temperature.

3.6 Summary

In this Chapter, we incorporated the leakage/temperature/supply voltage dependency into the real-time scheduling analysis that aims at minimizing the peak temperature. We showed that a constant-speed schedule is not always the optimal schedule in terms of peak temperature minimization for a given interval. We further showed that for a given periodic task, the lowest constant-speed is the optimal schedule among all two-speed schedules to minimize the peak temperature at the stable state. If this constant-speed is not available, then the schedule that uses the two closest neighboring speeds is the best choice. These new findings and theorems form the basis for the future study of developing more effective power and thermal aware scheduling techniques for more complicated architectures and real-time systems.

CHAPTER 4

M-Oscillations : A Scheduling Technique for Peak Temperature Minimization

In this chapter, we present a novel real-time scheduling technique, i.e. *the M-Oscillations* algorithm, that oscillates the high and low processor speeds to minimize the peak temperature for a periodic tasks set. We formally proved the correctness of this algorithm based on a processor power model that can capture the leakage/temperature dependency in reasonable accuracy yet simple enough and thus suitable for system level analysis. Furthermore, we validated the effectiveness of our algorithm based on the technology parameters derived from the 65nm technology. The experimental results demonstrated that our proposed scheduling technique can greatly reduce the peak temperature and, as a result, significantly improve the feasibility when scheduling periodic task sets under the maximum temperature constraints.

The rest of the chapter is organized as follows. Section 4.1 discusses the related work. System models are described in Section 4.2. The *M-Oscillations* scheduling algorithm is introduced in Section 4.3. Empirical results are presented in Section 4.4, and Section 4.5 concludes the chapter.

4.1 Related Work

There have been an increasing number of research results published on thermal aware real-time scheduling, for both single and multiple processor platforms (e.g. [14, 32, 105, 139]). Some approaches (e.g. [14, 32]) try to identify the upper bound of the maximum temperature. Some others (e.g. [14, 38, 29, 133]) intend to minimize the peak temperature or to guarantee the given maximum temperature constraints when scheduling a job set or a single copy of a task graph. While it is a common practice to repeat a real time schedule developed for jobs within the first hyperperiod of a periodic task set, as indicated in [35, 105], this approach is not applicable anymore if the temperature constraint is taken into consideration.

For periodic task sets, Wang et al. [126, 127] studied the maximum delay for *periodic* tasks when scheduling real-time tasks based on a two-speed scheduling policy. Zhang et al. [139] proposed to guarantee the temperature feasibility of a periodic system by forcing the temperature at the end of its first hyperperiod to be equal or less than the starting temperature. Quan et al. [105] developed a closed formula for the feasibility analysis under the maximum temperature constraint. None of these researches has taken the temperature/leakage dependency into consideration.

Some researches, such as that by Bao et al. [16], have applied equation (3.2) directly to capture the leakage/temperature dependency for the scheduling analysis. However, due to the non-linear and high-order magnitude terms in equation (3.2), such a model or tool can be too complex and cumbersome to be used for more rigorous real-time analysis and scheduling technique development. For example, Yuan et al. [138] also studied how to directly incorporate equation (3.2) into scheduling decisions. However, due to the complexity of equation (3.2), their approach can only be applied for soft real-time systems. There are also a number of other approaches formulate the temperature-constrained problem as a convex optimization problem [85, 92, 29]. The leakage/temperature dependency (equation (3.2)) may be incorporated into the optimization formulation [85]. The problem is that the computational complexity of the convex optimization problem is very high. Therefore these approaches can only work at system level when the design solution space is small.

A number of recent researches try to simplify the leakage/temperature dependency model. Liu et al. observed that the leakage current changes super linearly with temperature [84]. Based on this observation, a number of researches (such as

[35, 48, 30] adopt a simple temperature/leakage dependency model that assumes the leakage current changes linearly *only* with temperature. However, as can be seen from equation (3.2), leakage varies not only with temperature but also supply voltage as well. Quan et al. [104] introduced a leakage/temperature model that is more practical. According to their model, a processor has different running modes, and leakage varies at different rates with temperature when running at different modes. Based on this model, they presented several conditions to verify the feasibility of a *given* real-time schedule. However, how to develop a feasible and effective schedule for a given periodic task set under the maximum temperature constraint remains the problem. In what follows, with leakage/temperature dependency in mind, we develop a novel and scheduling technique that can effectively reduce the maximum temperature.

4.2 The System Models

In this section, we introduce the system models that are used in this chapter.

The Real-Time Model The real-time system we considered consists of a number of real-time tasks with the same period (such as the MPEG decoder). We can thus simplify this model by assuming that the real-time system has only one periodic task. The period of the task is denoted as p and its worst-case workload is c. We assume that the deadline of the task equals its period.

The Thermal Model We use the RC thermal model same as what we used in Chapter 3 and that has been widely used in the similar research (e.g. [30, 35, 92, 104]). Specifically, assuming a fixed ambient temperature (T_{amb}), let $T(t)$ be the temperature at time t. Then we have

$$RC\frac{dT(t)}{dt} + T(t) - RP(t) = T_{amb}, \tag{4.1}$$

where $P(t)$ denotes the power consumption (in $Watt$) at time t, and R, C denote the thermal resistance (in $J/^oC$) and thermal capacitance (in $Watt/^oC$). We can then scale T such that T_{amb} is zero and get

$$\frac{dT(t)}{dt} = aP(t) - bT(t), \tag{4.2}$$

where $a = 1/C$ and $b = 1/RC$.

The processor and its power model The processor can run in n different modes, with each mode as (v_i, f_i), $i = 0, 1, ..., n-1$. where v_i is the supply voltage and f_i is the working frequency in mode i. We assume that $v_i < v_j$, if $i < j$. We also assume that the processor speed is proportional to the supply voltage. In what follows, we use processor speed and supply voltage interchangeably.

Given a voltage level v, the power consumption is composed of dynamic P_{dyn} and leakage P_{leak}, i.e. $P = P_{dyn} + P_{leak}$. According to Liao et al. [77], the leakage power consumption can be estimated by the following,

$$P_{leak} = N_{gate} \cdot I_{leak} \cdot v \tag{4.3}$$

where N_{gate} represents the number of gate, v is the voltage level, and I_{leak} can be formulated using equation (3.2). As leakage current changes super linearly with temperature [84], we can simplify P_{leak} and define the leakage power for the processor running in mode k as

$$P_{leak}(k) = C_0(k)v_k + C_1(k)Tv_k, \tag{4.4}$$

where $C_0(k)$ and $C_1(k)$ are constants. This model is same as the LK_{TV} model discussed in Section 3.3.

The dynamic power consumption is independent of temperature, and can be formulated $P_{dyn} = C_2 v_k^{\xi} (\xi > 0)$. We choose $\xi = 3$ [107] in this chapter[1]. Hence the total

[1]Choosing other values will not change the conclusions in this chapter.

power consumption at processor mode k is

$$P(k) = C_0(k)v_k + C_1(k) \cdot Tv_k + C_2 v_k^3. \tag{4.5}$$

Based on equation (4.5) and (4.2), when a processor running in mode k, the temperature dynamics can be formulated as

$$\frac{dT(t)}{dt} = A(k) - BT(t) \tag{4.6}$$

where

$$A(k) = a(C_0(k)v_k + C_2 v_k^3) \tag{4.7}$$

$$B(k) = b - aC_1(k)v_k \tag{4.8}$$

For interval $[t_0, t_e]$, let the starting temperature be T_0, by solving equation (4.6), the ending temperature can be formulated as below:

$$\begin{aligned} T_e &= \frac{A(k)}{B(k)} + (T_0 - \frac{A(k)}{B(k)})e^{-B(k)(t_e - t_0)} \\ &= G(k) + (T_0 - G(k))e^{-B(k)(t_e - t_0)}. \end{aligned} \tag{4.9}$$

where

$$G(k) = \frac{A(k)}{B(k)}. \tag{4.10}$$

In what follows, we use A_k, B_k and G_k to denote $A(k)$, $B(k)$ and $G(k)$ respectively when there is no confusion. Equation (4.5) to (4.9) form the basis of our system level thermal analysis with leakage/termpature interplay taken into account.

4.3 Scheduling for Peak Temperature Reduction

In this section, we study how to minimize the maximum temperature when scheduling a periodic task set. Thermal-aware scheduling problems have distinct characteristics in comparison with the power aware scheduling problem as illustrated below.

Consider a simple two-speed schedule, as illustrated in Figure 4.1, that can finish a real-time job at its deadline. Note that, the dynamic energy consumption by the two-speed schedule shown in Figure 4.1 remains the same as long as the length for each individual speed keeps the same, i.e. t_1 and t_2 are constants. However, the temperature at $t = 1$ varies with the value of x. The following theorem captures this characteristic.

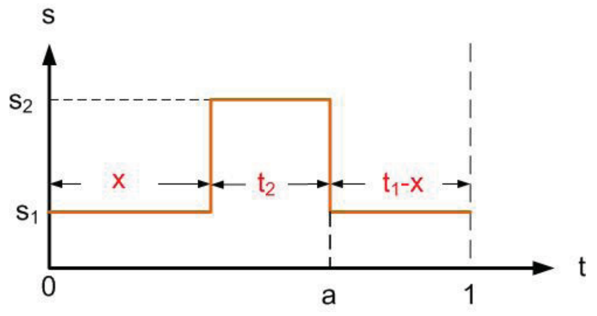


Figure 4.1: A Two-Speed Schedule that Uses Speed s_1 for t_1 Time Units and Speed s_2 for t_2 Time Units. $t_1 + t_2 = 1$.

Theorem 4.3.1. *Given a two-speed schedule as shown in Figure 4.1, and letting $s_2 > s_1$, if for any $s_2 > s_1$, we have $G_2 > G_1$ and $B_1, B_2 > 0$ (with G_k, B_k defined in equation (4.10) and (4.8), respectively), then the temperature at $t = 1$, i.e.T_e is a monotonically increasing function of x.*

Proof sketch: Based on equation (4.9), let T_a be the temperature at point a, then we have

$$T_e = G_1 + (T_a - G_1)e^{-B_1(t_1 - x)} \tag{4.11}$$

Therefore,

$$\begin{aligned}\frac{d(T_e)}{dx} &= \frac{d(G_1 + (T_a - G_1)e^{-B_1(t_1-x)})}{dx} \\ &= (G_2 - G_1)(1 - e^{-B_2 t_2})B_1 e^{-B_1(t_1-x)}. \qquad (4.12)\end{aligned}$$

So T_e decreases with decreasing x if $G_2 > G_1$ and $B_1, B_2 > 0$. □

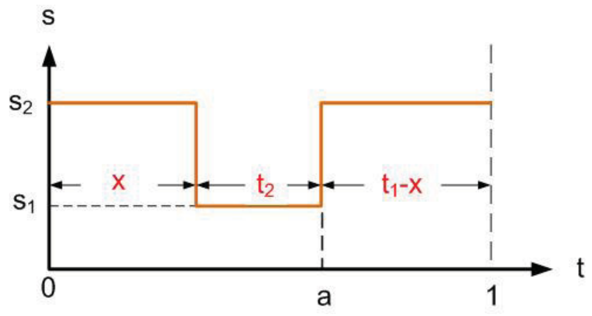


Figure 4.2: A Two-Speed Schedule that Uses Speed s_2 for t_1 Time Units and Speed s_1 for t_2 Time Units. $t_1 + t_2 = 1$.

Similarly, for the two-speed schedule illustrated in Figure 4.2, we have Theorem 4.3.2.

Theorem 4.3.2. *Given a two-speed schedule as shown in Figure 4.2, and letting $s_2 > s_1$, if for any $s_2 > s_1$, we have $G_2 > G_1$ and $B_k > 0$ (with G_k, B_k defined in equation (4.10) and (4.8), respectively), then the temperature at $t = 1$, i.e.T_e is a monotonically decreasing function of x.*

Proof sketch: Based on equation (4.9), T_a be the temperature at a, then we have

$$T_e = G_2 + (T_a - G_2)e^{-B_2(t_1-x)} \qquad (4.13)$$

Therefore,

$$\begin{aligned}\frac{d(T_e)}{dx} &= \frac{d(G_1 + (T_a - G_1)e^{-B_1(t_1-x)})}{dx} \\ &= -(G_2 - G_1)(1 - e^{-B_1 t_2})B_2 e^{-B_2(t_1-x)}. \end{aligned} \tag{4.14}$$

So T_e decreases with increasing x if $G_2 > G_1$ and $B_k > 0$. $\square$

Theorem 4.3.1 and 4.3.2 indicate that temperature at the end of a schedule depends on locations where different running modes are applied. They help to reduce the temperature at the end of schedule, but do not necessarily reduce the maximum temperature within the entire interval. In addition, Theorem 4.3.1 and 4.3.2 are applied for a single job rather than a periodic task set. In what follows, we introduced a novel scheduling algorithm (we call it the *M-Oscillations algorithm*) to minimize the peak temperature for a periodic hard real-time task. We assume that, when a processor runs a periodic task, the temperature will not run away and eventually reach a stable status. The temperature stable status is defined below.

Definition 4.3.3. *When running a periodic task with period p, the temperature at the processor is called to be* stable *if for a given threshold, i.e.* $0 < \varepsilon << 1$,

$$|T((n+1)p) - T(np)| < \varepsilon, \tag{4.15}$$

where $n \geq 0, n \in Z$, *and* $T(t)$ *is the temperature at* t.

Our *M-Oscillations* algorithm works as follows: given a two-speed schedule, we can divide the high speed interval and the low speed interval evenly into m sections, and run the processor with the low speed and high speed alternatively. Apparently, an m-oscillation schedule will complete the same workload as the original schedule in one period and thus guarantee the deadline. At the same time, the maximum temperature can be significantly reduced as stated in the following theorem.

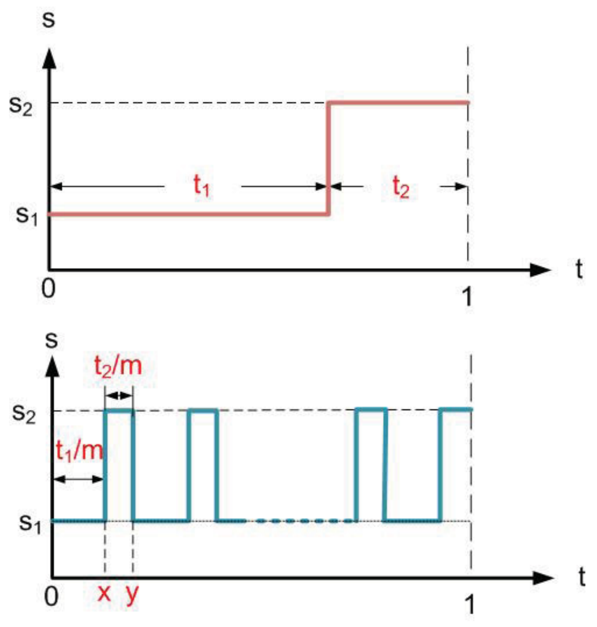


Figure 4.3: A Two-Speed Schedule and Its Corresponding *M-Oscillations* Schedule.

Theorem 4.3.4. *Let $S(t)$ be a two-speed schedule and $\tilde{S}(m,t)$ be the corresponding* M-Oscillations *schedule. Also let $T_{max}(S)$ represent the maximum temperature that a processor can reach when running schedule S. If for any $v_2 > v_1$, we have $G_2 > G_1$ and $B_i > 0, i = 1, 2$, then*

- $T_{max}(\tilde{S}(m,t)) \leq T_{max}(S(t))$;
- $T_{max}(\tilde{S}(n,t)) \leq T_{max}(\tilde{S}(m,t))$ *if* $m \leq n$.

Proof sketch: Here we only present the partial proof, i.e. for the case shown in Figure 4.3.

For $\tilde{S}(m,t)$ shown in Figure 4.3, base on equation (4.9), the temperature at $t = x$ and $t = y$ can be formulated as

$$T_x = G_1(1 - e^{-B_1 t_1/m}), \quad T_y = G_2 + (T_x - G_2)e^{-B_2 t_2/m}$$

From [104], when the temperature reaches the stable status, we have

$$T_{max}(\tilde{S}(m,t)) = T_y^{\infty} = T_y + \frac{T_y}{1-K_y} K_y$$

where

$$K_y = e^{-\frac{(B_1 t_1 + B_2 t_2)}{m}}.$$

Expand T_y^{∞}, we have

$$T_y^{\infty} = (G_2 - G_1) \frac{1 - e^{-\frac{B_2 t_2}{m}}}{1 - e^{-\frac{(B_1 t_1 + B_2 t_2)}{m}}} + G_1$$

Let $B_2 t_2 = m(m+1)p$ and $B_1 t_1 = m(m+1)q$, $p, q > 0$ and let

$$f(m) = \frac{1 - e^{-mp}}{1 - e^{-m(p+q)}}.$$

Then

$$\begin{aligned} T_{max}(\tilde{S}(m,t)) &= (G_2 - G_1) f(m+1) + G_1 \\ T_{max}(\tilde{S}(m+1,t)) &= (G_2 - G_1) f(m) + G_1 \end{aligned}$$

To show that $f(m+1) > f(m)$, we only need to note that

$$f(m) = \frac{1 - e^{-p}}{1 - e^{-(p+q)}} \cdot \frac{\sum_{i=0}^{m-1} e^{-ip}}{\sum_{i=0}^{m-1} e^{-i(p+q)}}.$$

Also,

$$
\begin{aligned}
& \frac{\sum_{i=0}^{m-1} e^{-ip}}{\sum_{i=0}^{m-1} e^{-i(p+q)}} < \frac{\sum_{i=0}^{m} e^{-ip}}{\sum_{i=0}^{m} e^{-i(p+q)}} \\
\Longleftarrow \quad & e^{-m(p+q)} \cdot \sum_{i=0}^{m-1} e^{-ip} < e^{-mp} \cdot \sum_{i=0}^{m-1} e^{-i(p+q)} \\
\Longleftarrow \quad & e^{-mq} \cdot \sum_{i=0}^{m-1} e^{-ip} < \sum_{i=0}^{m-1} e^{-i(p+q)} \\
\Longleftarrow \quad & \sum_{i=0}^{m-1} e^{-ip} < \sum_{i=0}^{m-1} e^{-ip} \cdot e^{(m-i)q}.
\end{aligned}
$$

With $i \leq m$, $e^{(m-i)q} \geq 1$, therefore $f(m+1) > f(m)$, and so

$$T_{max}(\tilde{S}(m,t)) > T_{max}(\tilde{S}(m+1,t)). \tag{4.16}$$

□

Theorem 4.3.4 implies that, by dividing the high speed interval and the low speed interval each into m equal sections and running them alternatively, an m-oscillating schedule can always reduce the maximum temperature when a processor reaches its stable status. The larger the m is, the lower the maximum temperature becomes.

Note that the conclusion in Theorem 4.3.4 and its proof are contingent upon two important assumptions, i.e. (i) $G_2 > G_1$ for any $v_2 > v_1$ and (ii) $B_i > 0, i = 1, 2$. It is difficult, however, to analytically validate these two assumptions since the temperature invariants C_0 and C_1 in equation (4.8) and (4.10) depend on the technology parameters. In addition, C_0 and C_1 are obtained through curve-fitting rather than from a closed analytical formula. In Section 3.3, we have already validated all the assumptions empirically, including the accuracy of the temperature sensitive linear leakage model.

4.4 Experiments and Results

In this section, we use experiments to examine the *M-Oscillations* scheduling algorithm. We evaluate its performance by comparing it with a previous work, i.e. the two-speed scheduling method [127], in terms of the feasibility and peak temperature.

4.4.1 Performance Evaluation

We next study the performance of *M-Oscillations* scheduling algorithm by comparing with the existing approaches. The proactive scheduling method introduced in [35] intends to minimize the task response time under given maximum temperature constraints. However, it is developed based on a processor model with continuously changeable speed, and to extend the proposed scheduling technique to a more practical processor model (i.e. with discrete supply voltages) as we used in this chapter is far from a trivial and straight forward effort. Therefore, we compare our approach with a more general one, i.e. the reactive two-speed scheduling approach introduced in [127]. The reactive two-speed schedule [127] work as follows. For a given maximum temperature constraint, the processor works at the highest speed until it reaches the maximum temperature. Then it runs at an equilibrium speed to maintain the temperature.

First, we want to investigate the feasibility of the two scheduling policies, i.e. the *M-Oscillations* schedule and the reactive two-speed schedule, under the same maximum temperature constraints and workloads.

Note that for a given maximum temperature and a processor with discrete speeds, the equilibrium speed is not necessarily one of the available speeds. We therefore fixed the equilibrium speed to one of the available speeds of the processor, and then used the stable temperature as the maximum temperature constraint to test both scheduling policies.

Table 4.1: Equilibrium Speeds & Corresponding Maximum Temperatures

$V_{Equil}(V)$	$T_{max}(^{o}C)$
0.80	33.99
0.90	38.88
1.0	46.16

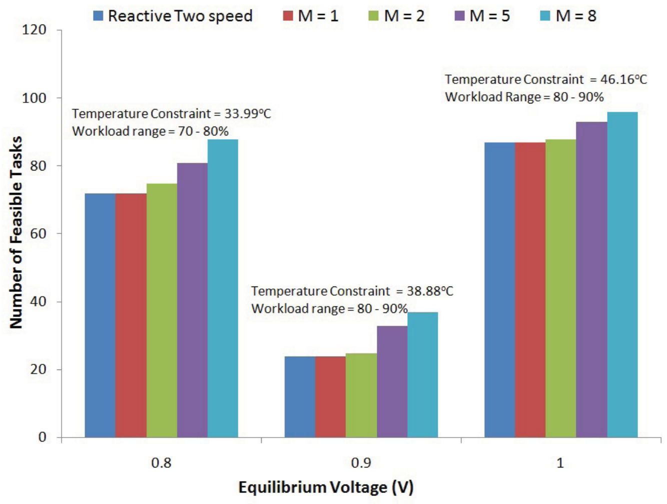


Figure 4.4: Feasibility Comparison Between the *M-Oscillations* Scheme and the Reactive Two-Speed Scheme under Different Maximum Temperature Constraints

We randomly generated real-time tasks with period of 2000 seconds and workload evenly distributed within range of [0, 100%], with 100% indicating that the processor has to run at the maximum speed all the time (i.e. 100%) to complete the workload. We divided the task workload into 10 equal intervals, i.e. 0-10%,10-20% and so on, and 100 random tasks were generated within each interval. The equilibrium voltages were set to be 0.8V, 0.9V and 1.0V, and the corresponding stable temperature were set as the maximum temperature constraint. Table 6.2(a) lists the values of the equilibrium voltages and their corresponding stable temperatures.

For *M-Oscillations* schedule, we first calculated the constant speed that will guarantee workload. Then the two neighboring speeds were used to construct our *M-Oscillations* schedule algorithm described in section 4.3. Figure 4.4 presents the feasibility differences between active two-speed scheduling and M-Oscillations scheduling with m=1, 2, 5, and 8. When the randomly generated workload is very low, all above scheduling policies can schedule the task feasibly; and when the workload is high, none four scheduling policies can make the task feasible. Therefore Figure 4.4 only depicts the workload regions that there exist differences in terms of feasibility among different scheduling choices. From Figure 4.4, we can clearly see that *M-Oscillations* scheduling shows higher feasibility as compared to reactive two-speed schedule. The larger the m is, the higher the feasibility can be. At the equilibrium voltage of 0.8V, the feasibility by the active two-speed scheduling policy is very close to the *M-Oscillations* scheduling algorithm. However, when m is increased to 5, the feasibility is improved over 13%, and up to 20% when $m = 8$. At the equilibrium voltage of 0.9V and 1.0V, we can see the feasibility improvement of 35% and 10%, respectively, by *M-Oscillations* algorithm to the two-speed scheduling algorithm.

Even though a task can be feasibly scheduled, a higher peak temperature is not desirable since it increases packaging and cooling costs, degrade the performance, life span, and reliability of a computing system. We therefore collected the maximum temperatures of all feasible tasks under different scheduling policies and compared their average maximum temperatures as shown in Figure 4.5. Figure 4.5 clearly demonstrates that the *M-Oscillations* algorithm is very effective in reducing the peak temperature. Note that at the equilibrium voltage of 0.8V, the average maximum temperature of reactive two-speed schedule is 31.84^oC, and is reduced to 28.64^oC when $m = 1$ for the *M-Oscillations* scheduling algorithm. It is further reduced to 27.85^oC for $m = 5$. At equilibrium voltage of 0.9V and 1.0V, the average maximum temperatures are reduced by 7.78^oC and 14.0^oC, respectively.

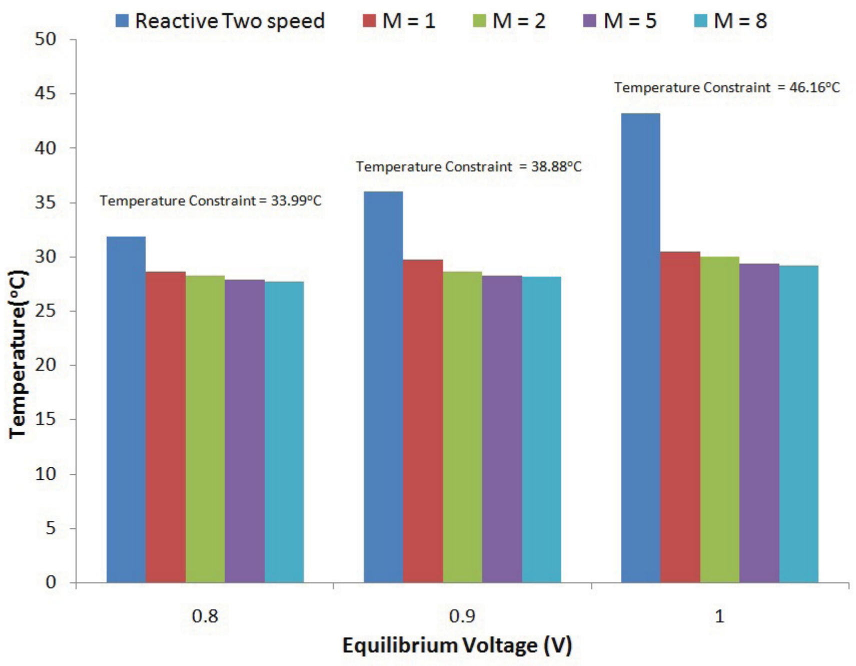


Figure 4.5: Average Maximum Temperature Comparison Between the *M-Oscillations* Scheme and the Reactive Two-Speed Scheme

4.4.2 Performance Evaluation on a Practical Simulation Platform

After evaluating the performance of our proposed *M-Oscillations* scheduling algorithm on a synthetic simulation platform, we now extend our experiments on to more practical simulation framework. We developed a practical simulation framework by combining some of the most practical simulators available in the research domain.

Practical Simulation Platform Set-Up

The simulator model is based on SimpleScalar [2, 13] and it is composed of two major parts. The first part is power simulation part which uses Wattch [3, 27] as the simulation tool. After Wattch generates the power output for each module of the platform, we calculate the temperature using HotSpot [4] in the second part of simulation. Hotspot uses the existing power information to find the temperature characteristics for each benchmark program in SPEC CPU2000.

Table 4.2: The Different Supply Levels and Corresponding Frequencies

$Supplylevel(V)$	$Frequency(GHz)$
0.90	5.0
1.1	5.57
1.3	6.03

The selected platform is Alpha EV6 which is available in Hotspot [4] and has its power model in Wattch and SimpleScalar. We select gcc integer benchmark program from SPEC CPU2000 to perform the simulation. The input for gcc benchmark program is gcc.lgred.cp-decl.i.

Modifications

To match the requirements for our experiments, we did several modifications in the simulation models.

We modified the Wattch power simulator to be able to calculate the power of each floorplan unit every single clock cycle. Furthermore, SimpleScalar and Wattch have been modified such that the power can be calculated from different supply voltage levels and frequencies. The original power formulas are maintained. The modules for power calculation are duplicated and slightly modified to be compatible with different supply voltage levels and frequencies. In the experiments, we select 100 kilocycles as our simulation interval size. Other configurations are kept default.

Hotspot has been modified to compute the temperature according to the interval size. The initial temperature of each floorplan unit has been redefined to 25 degree centigrade (298 Kelvin). Based on the work by Liao at al. [77], the processor is provided with three different supply levels and corresponding frequencies(Table 6.2(a)). Different supply voltages and frequencies can be redefined at any specific clock cycle of execution.

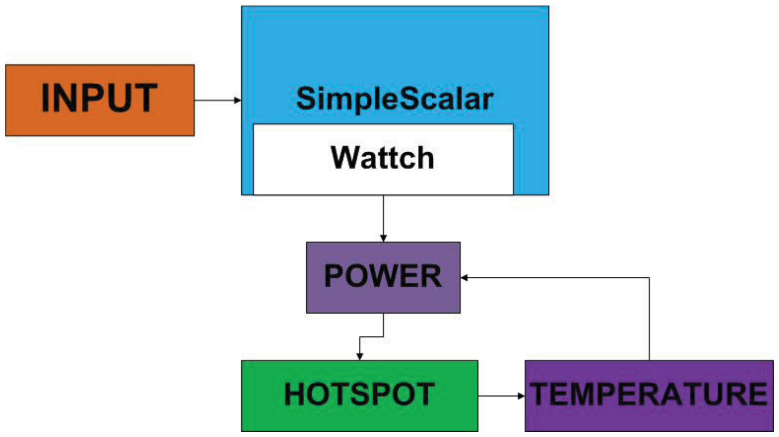


Figure 4.6: Conceptual Flow Diagram of the Simulator Platform Set-Up

Execution Flow

The execution of our simulation starts with the power simulation, the SPEC CPU2000 is executed within SimpleScalar which will subsequently call Wattch to calculate power consumed by each floorplan unit during the execution of SPEC CPU2000 benchmark program. If the supply voltage or the frequency is changed during the execution, Wattch will update all its defined parameters that are related to the supply voltage and frequency and continue its power calculation for each clock cycle.

Next, the Hotspot is called after the SimpleScalar and Wattch have finished generating the power profile. Hotspot will open the initial configuration files given including the recorded power file from Wattch. Hotspot calculates temperature from the existing power parameters and output the instantaneous temperature to a file and also update the power profile with the change in power values that occur due to increase in leakage power as a result of increase in temperature (Figure 4.6).

Experiments

In this section, we discuss the different experiments that we have performed on our novel simulation platform discussed in Section 4.4.2. In our experiments, we assumed that the processor is provided with discrete supply levels. The available supply levels

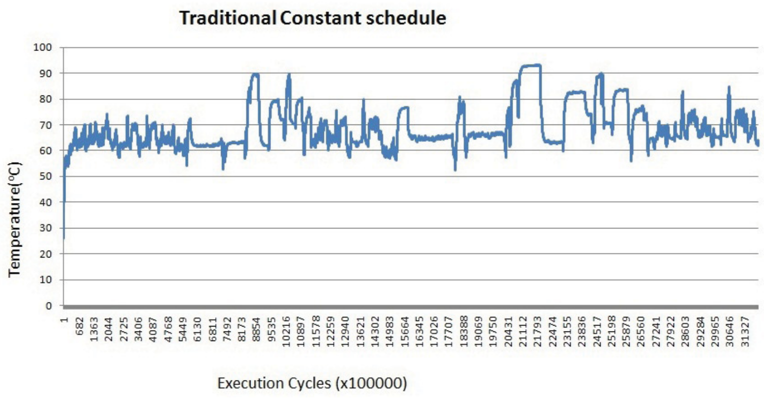


Figure 4.7: Temperature Pattern Running a Traditional Constant Speed Schedule

and the corresponding frequencies are shown in the table 6.2(a). These values are taken from the work shown in [77]. We start our experiments with first running the entire gcc program from SPEC CPU2000 benchmark using a constant speed schedule with supply level of 1.1 volts and the corresponding frequency of 5.57 GHz. We followed the execution flow discussed in the previous section and generated a temperature profile for each module of the Alpha EV6 processor model. To get the detail of each module, interested reader can refer to the reference paper [2, 13].

We next execute the entire gcc benchmark with *M-Oscillations* schedule using the two neighboring supply levels i.e. 0.9 and 1.3 volts and their corresponding frequencies i.e. 5.0 and 6.03 GHz respectively. We ran the experiment for different values of m, for example $1, 2, 4, 10, 100, 1000, 5000, 9000$ etc.

As running a complete benchmark program results in extensive volume of data, it is not possible for us to present the entire data that we have collected. We therefore present the comparison of the temperature pattern due to the *integer-register1* module of the processor model which results in the highest peak temperature among all the modules on the platform. In Figure 4.7, 4.8, 4.9 and 4.10, we have shown the temperature pattern for traditional constant schedule and *M-Oscillations* ($m = 100, 1000, 10000$) on the *integer-register1* module of the platform. From the

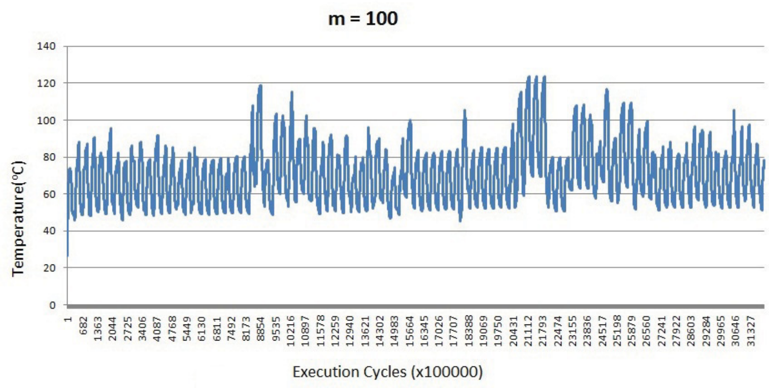


Figure 4.8: Temperature Pattern Running a *M-Oscillations* with $m = 100$

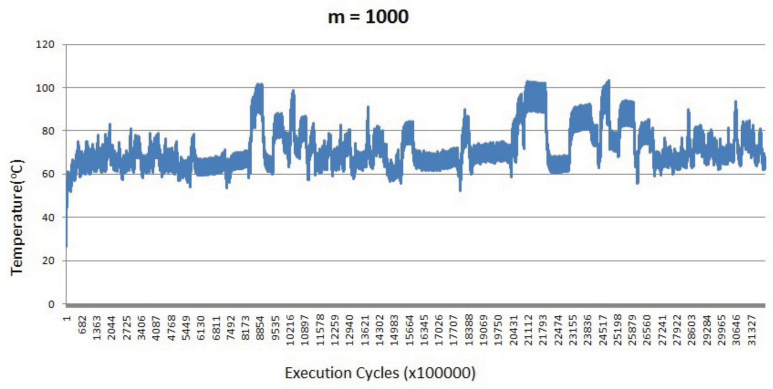


Figure 4.9: Temperature Pattern Running a *M-Oscillations* with $m = 1000$

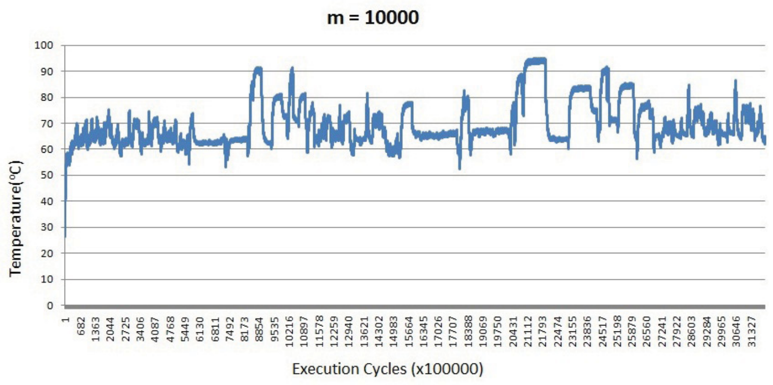


Figure 4.10: Temperature Pattern Running a *M-Oscillations* with $m = 10000$

figures, we can observe two important characteristics of *M-Oscillations*. Firstly, with increasing value of m, the peak temperature of the system decreases; for example, when the value of m is equal to 100, peak temperature is 125^oC, and when m is equal to 10000, peak temperature reduces to 94^oC.

Secondly, for an appropriate value of m, *M-Oscillations* can be as effective as traditional constant speed schedule in reducing the peak temperature; for example, the peak temperature due to the constant speed schedule is 92^oC and the peak temperature due to *M-Oscillation* for m equal to 10000 is 94^oC. Calculation of the appropriate value of m on such a practical platform is a non-trivial task, however it is a part of our future work to develop algorithm to calculate optimum value of m on such a platforms.

From the results presented, we can see that on a very practical architectural platform, the *M-Oscillations* schedule is effective in reducing the peak temperature of the system and can be used as an effective substitute to traditional approach, when it is not possible to implement the constant speed schedule due to unavailability of the desired lowest constant speed to execute the workload.

4.5 Summary

As semiconductor technology continues to scale down, the positive feedback loop between temperature and leakage exacerbates not only the power/energy minimization problem but also the thermal management problem. In this chapter, we incorporated the leakage/temperature dependency into the real-time scheduling analysis that aims at minimizing the maximum temperature. We presented and proved a number of theorems and exhibit the distinct characteristics of thermal aware real-time scheduling. We also proposed a new scheduling technique, i.e. the *M-Oscillations* scheduling that can effectively reduce the peak temperature when executing a hard real-time periodic task set. These theorems and techniques form a solid basis for further leakage-aware temperature-constrained researches in design and development of practical real-time systems. Our future research will be based on the theorems presented in this chapter and extended in a number of ways, including more complex real-time system models, processors with non-trivial transition overhead, and multiple-core type of architectures.

CHAPTER 5
Feasibility Analysis for Temperature-Constraint Hard Real-Time Periodic Tasks

In the previous two chapters, our focus was on developing novel solutions to minimize the peak temperature of the uniprocessor system by judiciously applying DVS algorithms. In this chapter, we extended our research work to an equally important problem of insuring successful execution of the given tasks set within a given deadline without violating any design constraints. Specifically, we studied the problem on how to guarantee the feasibility of a periodic tasks set under the maximal temperature constraint.

Traditionally, one common strategy is to check if each task instances of the task set can meet their deadlines within the first hyperperiod, i.e. the least common multiple (LCM) of the task periods. However, when we consider the maximal temperature constraints, this strategy does not apply anymore. As shown in [105, 35] as well as later in the chapter, a schedule for a periodic task set that can satisfy both the timing and the maximal temperature constraint within the first hyperperiod is not necessarily feasible later in the schedule. Therefore, new techniques need to be developed for checking the schedulability of the real-time periodic tasks set under the maximal temperature constraint.

We then present new necessary and sufficient conditions to check the feasibility of real-time schedules. We further incorporate the leakage/temperature dependency into our feasibility analysis, and develop more elaborated feasibility conditions. Our experiments, based on technical parameters derived from a processor using the 65nm IC technology, demonstrate the effectiveness of our feasibility conditions and, at the same time, highlight the fact that a power/thermal-aware computing technique becomes ineffective if the temperature/leakage dependency is not properly addressed.

The rest of the chapter is organized as follows. We introduce the related work in section 5.1. Section 5.2 discusses the system models and formulates our problem formally. In section 5.3, we study the unique characteristics of feasibility analysis problem for periodic tasks under maximal temperature constraints. In section 5.4, we incorporate the leakage/temperature dependency into our feasibility analysis and introduce several feasibility checking methods. We present our experimental results in section 5.5 and draws conclusions in section 5.6.

5.1 Related Work

In this chapter, we focus on the feasibility checking problem for real-time tasks running on a single processor. There are several closely related research works have been proposed, for instance some of the researches (e.g. [14, 32]) try to identify the upper bound of the maximal temperature when executing real-time tasks on a single processor. These techniques cannot guarantee that real-time tasks can still meet deadlines when the maximal temperature is given. Some others (e.g. [14, 38, 29, 109, 133]) intend to minimize the peak temperature or guarantee the given maximal temperature constraints when scheduling a job set or a single copy of a task graph. For example, Bansal et al. [14] introduced an off-line technique to minimize the energy consumption for a *job set*, and Chantem et al. [29] proposed an MILP-based solution to minimize the peak temperature when executing a task graph. While it is a common practice to repeat a real time schedule developed for jobs within the first hyperperiod of a periodic task set, as noted by Quan et al. [105] and Chen et al. [35], this approach is not applicable anymore if the temperature constraint is taken into consideration.

For periodic task sets, Wang et al. [127, 126] considered the problem of using two processor speeds to schedule a hard real-time task set. A processor runs at the highest possible speed until the temperature reaches the temperature threshold. Then the processor is set to run with the “equilibrium speed“ at which the processor enters the equilibrium state with its temperature unchanged. This approach does not take the advantages that many modern processors support more than two levels of running speeds. In addition, it is not always possible that the “equilibrium speed“ is exactly one of the available processor speeds. Zhang et al. [139] proposed to guarantee the temperature feasibility of a periodic system by forcing the temperature at the end of its first hyperperiod to be equal or less than the starting temperature. However, as shown later in this chapter, this constraint can be overly pessimistic. In addition, none of these researches has taken the temperature/leakage dependency into consideration.

Researchers have already studied in depth the complex relationship between the leakage and temperature at the circuit and micro architecture level [77, 145]. Following what, few recent papers incorporate the temperature/leakage dependency into the energy- or thermal-aware scheduling. He et al. [56] and Yuan et al. [137] studied how to reduce the leakage power at the system level. Yuan et al. [138] introduced an offline and an on-line scheduling algorithm that take into account the leakage/temperature interactions when scheduling a set of soft real-time jobs. This approach cannot guarantee that real-time periodic tasks can meet deadlines under the given maximal temperature. A number of other approaches formulate the temperature-constrained problem as a convex optimization problem (e.g. [85, 92]). The leakage/temperature relationship can thus be formulated as one of the constraints. The problem is that the computational complexity for convex optimization problems is very high. Therefore these approaches can only work at system level when the design solution space is small.

Liu et al. showed that linear models can be used to permit highly-accurate leakage estimation over the operating temperature ranges in real ICs [84]. A number of researches (such as [35, 48, 104, 30]) simplify the leakage/temperature relationship based on this idea. Specifically, Chen et al. [35] and Chantem et al. [30] adopted a simple temperature/leakage dependency model that assumes the leakage power changes linearly *only* with temperature. As can be seen from Equation 3.2, leakage varies not only with temperature but also supply voltage as well. In Section 5.5, we use experiments to study the accuracy of this model and its impacts to the schedulability analysis results.

5.2 Preliminary

The real-time system considered in this chapter contains n independent periodic tasks, $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{n-1}\}$. Task τ_i is characterized using three parameters, *i.e.*, $\tau_i = (p_i, d_i, c_i)$. p_i, d_i ($d_i \leq p_i$), and c_i represent the period, the deadline and the worst case execution time for τ_i, respectively.

We use the lumped RC model similar to the model in Chapter 3([117]) to capture the thermal phenomena of the processor. Specifically, assuming a fixed ambient temperature (T_{amb}), let $T(t)$ denote the temperature at time t. Then we have

$$RC\frac{dT(t)}{dt} + T(t) - RP(t) = T_{amb}, \tag{5.1}$$

where $P(t)$ denotes the power consumption (in $Watt$) at time t, and R, C denote the thermal resistance (in $J/^oC$) and thermal capacitance (in $Watt/^oC$). We can then scale T such that T_{amb} is zero and get

$$\frac{dT(t)}{dt} = aP(t) - bT(t), \tag{5.2}$$

where $a = 1/C$ and $b = 1/RC$. For the rest of the chapter, we assume that the initial temperature of the processor equals the ambient temperature. The ambient temperature is assumed to be constant.

We assume the processor can run in different modes, with each mode being characterized by a pair of parameters (v_i, f_i), where v_i is the supply voltage and f_i is the working frequency in mode i. Even though the circuit delay changes with the temperature dynamically, as shown in Equation 5.3 [77],

$$f_i = \frac{1}{t_d} \propto \frac{(V_i - v_t)^{\mu}}{V_i T^{\eta}}, \tag{5.3}$$

where v_t is the threshold voltage, t_d is the circuit delay, and μ and η are technology-related constants, we assume that the processor working frequency in each mode is fixed, and is the one that can accommodate the peak temperature (i.e. by assigning the peak temperature in Equation 5.3) across the chip. Let f_{max} be the largest f_i among different modes. We can normalize the processor working frequency with f_{max} and get the normalized processor speed for each mode. In what follows, unless otherwise specified, we use the term processor speed or working frequency interchangeably.

The power consumption (P) of the processor consists of two parts: the dynamic power(P_{dyn}) and the leakage power (P_{leak}).

$$P = P_{dyn} + P_{leak}. \tag{5.4}$$

The dynamic power consumption is independent of the temperature and can be formulated as $P_{dyn} \propto v_k^{\xi}$ with $\xi > 1$ [107]. For simplicity, we choose $\xi = 3$.

The leakage power is sensitive to the temperature and can be estimated using the following formula,

$$P_{leak} = N_{gate} \cdot I_{leak} \cdot V_{dd} \tag{5.5}$$

where N_{gate} is the total number of gates, V_{dd} is the supply voltage and I_{leak} can be determined by Equation 3.2.

Varying processor supply voltage and working frequency is one of the most effective ways to manage the power consumption dynamically. We call a schedule that dictates how to vary the processor supply voltage and working frequency as the *speed schedule*, which is formally defined as follows.

Definition 5.2.1. *Given periodic task set $\mathcal{T}$, let L be the least common multiple (LCM) of the periods, i.e.,$p_0, p_1, \cdots, p_{n-1}$. The speed schedule $\hat{S}(t)$ is defined as a sequence of $< [st_i, ed_i], mode_i >$, where*

- *$[st_i, ed_i]$ is an interval in which the processor runs in $mode_i$,*
- *$\bigcup_i [st_i, ed_i] = [0, L]$, and*
- *$[st_i, ed_i] \bigcap [st_j, ed_j] = \emptyset$ if $i \neq j$.*

With the thermal and processor models introduced as above, our problem can be formulated as follows:

Problem 5.2.2. *Given*

- *a hard real-time task set $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{n-1}\}$,*
- *a variable voltage processor that can run in m different modes, i.e. (v_i, f_i), $i = 0, \cdots, m-1$,*
- *the maximal allowable temperature T_{max},*
- *and a speed schedule $\hat{S}(t)$ with l intervals, i.e. $< [t_i, t_{i+1}], mode_i >$, $i = 0, 1, \cdots, l-1$,*

determine if $\mathcal{T}$ can meet the required deadlines using $\hat{S}(t)$ with the temperature stays below T_{max} all the time.

5.3 The Leakage Oblivious Feasibility Analysis

In this, we study Problem 5.2.2 assuming that the leakage power is negligible. Under this assumption, the overall power consumption is therefore independent of temperature. Through this study, we intend to gain some valuable insights on how to deal with the maximal temperature constraints in the feasibility analysis for periodic task systems. We then incorporate the leakage/temperature dependency and develop several more elaborated feasibility conditions.

One common practice to ensure the feasibility of a periodic real-time task set is to construct a feasible schedule with interval $[0, L]$, where L represents the hyperperiod, i.e. the least common multiple (LCM) of task periods. As long as the tasks are feasible in $[0, L]$, by replicating the schedule, the timing feasibility of the real-time system is guaranteed. However, when the execution of the real-time tasks are further constrained by a maximal temperature, is this approach still feasible?

We first introduce Theorem 5.3.1 which helps to answer this question.

Theorem 5.3.1. *Given periodic task set $\mathcal{T}$, let*

- *L be the LCM of the periods, i.e.,$P_0, P_1, \cdots, P_{n-1}$,*
- *$\hat{S}(t)$ be the speed schedule within interval $[0, L]$ that can guarantee the deadlines of $\mathcal{T}$ under the maximal temperature constraints T_{max} with the initial temperature $T(0)$.*

Then, when repeating $\hat{S}(t)$ later in the schedule, all task deadlines can be guaranteed under initial temperature $T(0)$ if $T(L) \leq T(0)$.

Proof. For interval $[t_0, t_1]$, let the temperature at $t = t_0$ be $T(t_0)$. Assuming there is no leakage power, based on Equation 5.4, we can simplify the overall power consumption formulation as $P = v_k^3$, where v_k is the supply voltage when the processor is running in mode k. By solving Equation 5.2, we have

$$T(t_1) = \int_{t_0}^{t_1} av^3(\tau)e^{-b(\tau - t_0)}d\tau + T(t_0)e^{-b(t_1 - t_0)}. \tag{5.6}$$

So we have

$$T(L) = \int_0^L av^3(\tau)e^{-b\tau}d\tau + T(0)e^{-bL}. \tag{5.7}$$

If we repeat $\hat{S}(t)$ for interval $[L, 2L]$, we have

$$T(2L) = \int_L^{2L} av^3(\tau)e^{-b(\tau - L)}d\tau + T(L)e^{-bL} \tag{5.8}$$

$$= \int_0^L av^3(\tau - L)e^{-b\tau}d\tau + T(L)e^{-bL} \tag{5.9}$$

Note that $v^3(\tau) = v^3(\tau - L)$. Thus we have

$$T(2L) - T(L) = (T(L) - T(0))e^{-bL}. \tag{5.10}$$

So, for the $(k+1)$th LCM interval, we have

$$T((k+1)L) - T(kL) = (T(L) - T(0))e^{-kbL}. \tag{5.11}$$

Therefore, when $T(L) < T(0)$, the temperatures at $t = 0, L, 2L, \cdots$ will be monotonically decreasing.

This ensures that if the maximal temperature constraint is not violated within $[0, L]$, it will not be violated within interval $[L, 2L]$, $[2L, 3L]$,$\cdots$. Therefore, under this scenario, $\hat{S}(t)$ must be globally schedulable. $\square$

Theorem 5.3.1 states that as long as the temperature at the ending point of a schedule is no more than the initial temperature at $t = 0$, repeating the schedule that is feasible during the first LCM interval is safe to guarantee the temperature and timing constraint. The question is then what if $T(L) > T(0)$. We present another theorem for this case. We use the same notation as that in Theorem 5.3.1.

Theorem 5.3.2. *If $T(L) > T(0)$, when repeating $\hat{S}(t)$, all task deadlines can be guaranteed with initial temperature $T(0)$ if and only if*

- Condition 1: $T(L) \leq (T_{max} - T(0))(1 - e^{-bL}) + T(0)$;
- Condition 2: $T(t_m) \leq T_{max} - \frac{T(L)-T(0)}{1-e^{-bL}} e^{-bt_m}$ *for all* $t_m \in [0, L]$ *such that* $T(t_m) \geq T(t), t \in [0, L]$.

Proof. From Equation 5.11, when $T(L) > T(0)$, the temperatures at $t = 0, L, 2L, \cdots$ will be monotonically increasing. Also $T(L)-T(0), T(2L)-T(L), T(3L)-T(2L), \cdots, T((k+1)L, kL)$ forms a geometric series and we have

$$T((k+1)L) = T(0) + \frac{(T(L) - T(0))(1 - e^{-kbL})}{1 - e^{-bL}}. \tag{5.12}$$

As $k \to \infty$, we have

$$\lim_{\to\infty} T(kL) = T(0) + \frac{(T(L) - T(0))}{1 - e^{-bL}}. \tag{5.13}$$

So, $T(kL) \leq T_{max}$ if and only if

$$T(L) \leq (T_{max} - T(0))(1 - e^{-bL}) + T(0). \tag{5.14}$$

We next examine the temperature feasibility for the points within each LCM. Let $t_m \in [0, L]$ such that $T(m) \geq T(t)$ for any $t \in [0, L]$. Let $t_{m'} \in [kL, (k+1)L]$ and $t'_m = t_m + kL$. We want to show that $T(t_{m'}) \leq T_{max}$ if and only if Condition 2 holds. Based on Equation 5.6, similarly, we have

$$T(t_{m'}) = \int_{kL}^{(k+1)L} av^3(\tau)e^{-b(\tau-kL)}d\tau + T(kL)e^{-b(t_{m'}-kL)}, \tag{5.15}$$

and

$$T(t_m) = \int_0^L av^3(\tau)e^{-b\tau}d\tau + T(0)e^{-bt_m}. \tag{5.16}$$

Since $t_m = t_{m'} - kL$, we have,

$$T(t_{m'}) = T(t_m) + \frac{(T(kL) - T(0))}{e^{bt_m}}. \tag{5.17}$$

Since

$$\lim_{k\to\infty} T(kL) = T(0) + \frac{T(L) - T(0)}{1 - e^{-bL}}, \tag{5.18}$$

so, $T(t_{m'}) \leq T_{max}$ if and only if

$$T(t_m) \leq T_{max} - \frac{T(L) - T(0)}{1 - e^{-bL}}e^{-bt_m}. \tag{5.19}$$

□

Theorem 5.3.1 and Theorem 5.3.2 provide the necessary and sufficient condition to predict if a schedule feasible within the first LCM is globally feasible. On the other hand, Theorem 5.3.2 also implies that not all schedules are feasible under the maximal temperature constraint even if they can guarantee the deadlines and maintain the maximal temperature below T_{max} during the first LCM, i.e., $[0, L]$. This is another example that the temperature-constrained real-time scheduling problem has its unique characteristics, compared with the corresponding power-aware scheduling problems.

5.4 The Leakage Conscious Feasibility Analysis

The results in previous section reveal some interesting and important characteristics in feasibility analysis for periodic tasks under the maximal temperature constraint, with the leakage power consumption ignored. However, the leakage power consumption is too significant to be ignored in the deep sub-micron domain, as stated before. In this section, we take the leakage power consumption into account and conduct a more sophisticated study on feasibility analysis.

5.4.1 Simplifying the leakage/temperature dependency

When taking the leakage into account, one of the biggest challenges is to deal with the complex behavior of the leakage current, as formulated in Equation 3.2. While Equation 3.2 can capture accurately the characteristics of leakage current, the high order and non-linear terms make it prohibitive for our real-time feasibility analysis. Liu et al. [84] found that using linear approximation method to model the leakage/temperature dependence can maintain reasonable accuracy, i.e. with error within 1% using the piece-wise linear function or less than 5.5% using single linear function, but the leakage model is significantly simplified. Based on this idea, we define the leakage power for the processor running in mode k as

$$P_{leak}(k) = (C_0(k) + C_1(k)T) \cdot v_k, \tag{5.20}$$

where $C_0(k)$ and $C_1(k)$ are constants that depend on the running mode, i.e. k. This model is same as the LK_{TV} discussed in Chapter 3. In what follows, we omit variable k for sake of conciseness.

The overall power consumption in mode k is thus given by the following formula.

$$P(k) = (C_0 + C_1T) \cdot v_k + C_2v_k^3. \tag{5.21}$$

C_0, C_1 and C_2 can be determined in practice once the practical power consumptions at different temperatures are profiled.

With the simplified leakage power formulation, we are now able to formulate the temperature dynamics in a closed form. Note that, when the processor runs in mode k, by combining Equation 5.2 and 5.21, we have temperature variations as follows:

$$\frac{dT(t)}{dt} = A(k) - B(k)T(t), \tag{5.22}$$

where

$$\begin{aligned} A(k) &= a(C_0 v_k + C_2 v_k^3) && (5.23) \\ B(k) &= (b - aC_1 v_k) && (5.24) \end{aligned}$$

If we run processor in mode k during interval $[t_1, t_2]$, with temperature at $t = t_1$ be $T(t_1)$, by solving equation 5.22, we can thus get the temperature at $t = t_2$ as

$$T(t_2) = \frac{A(k)}{B(k)} + (T(t_1) - \frac{A(k)}{B(k)})e^{-B(k)(t_2 - t_1)}. \tag{5.25}$$

Equation 5.20 to 5.25 form the basis of feasibility analysis with leakage/termpature interplay taken into account. In what follows, we introduce several feasibility conditions developed based on this leakage power consumption model.

5.4.2 Checking the Temperature at the End of First Hyperperiod

The reason that a periodic task set feasible within the first hyperperiod is not necessarily feasible later in its life time is that the temperature at the end of a hyperperiod may be higher than that at the beginning of the hyperperiod. If this is case, starting at a new hyperperiod, the processor will run at a higher initial temperature and continue to reach an even higher temperature at the end of this hyperperiod. As this process continues, the temperature may eventually exceed the peak temperature. Conversely, from Theorem 5.3.1, as long as we can ensure that the temperature within the first hyperperiod is not higher than T_{max}, and as long as the ending temperature is not higher than the initial temperature, we can determine that a schedule must be feasible under the given maximal temperature constraint. However, assuming the initial temperature be the ambient temperature, unless some aggressive cooling strategies are applied, the temperature of a processor will always increase when executing tasks. Therefore, the applicability of Theorem 5.3.1 is very limited. Next, we introduce two other theorems that can effectively deal with the case when $T(L) > T(0)$.

5.4.3 Checking the Temperature Safe Modes

Recall that, in Section 5.2, the processor can work in different modes, each of which is associated with a distinct pair of supply voltage and working frequency. For some of the modes, no matter how long the processor runs in that mode, the final temperature will never exceed the given T_{max}. We call these processor modes as the *safe modes*.

To determine if a processor mode, i.e. mode k, is safe, we can set

$$\frac{dT(t)}{dt} \,|_{T(t)=T_{max}} = 0, \tag{5.26}$$

Based on Equation 5.2 and 5.21, we have

$$a((C_0 + C_1 T_{max}) \cdot v + C_2 v^3) - bT_{max} = 0. \tag{5.27}$$

Note that Equation 5.27 is the classic *depressed cubic equation* [93]. In addition, if we transform Equation 5.27 slightly, we have

$$aC_2 v^3 = -a((C_0 + C_1 T_{max})) \cdot v + bT_{max}. \tag{5.28}$$

From Section 5.4.1, it is not difficult to see that $aC_2 > 0$, and $a((C_0 + C_1T_{max})) > 0$. Therefore, as illustrated in Figure 5.1, Equation 5.27 has only one single *real* root for v, which can be solved analytically [128]. We call the solution to Equation 5.27 as the *equilibrium voltage*. Note that different processor running modes may have different equilibrium supply voltages since C_0 and C_1 in Equation 5.28 are different in different modes.

Formally, we have the following lemma to determine whether or not a processor running mode is a safe mode.

Lemma 5.4.1. *Let v_e be the equilibrium voltage (i.e. the solution to Equation 5.27) for processor's mode k (i.e. (v_k, f_k)). Then this mode is a* safe mode *if $v_e \geq v_k$.*

Proof. We prove it by contradiction. Assume that at time $t \leq t_0$ we have $T(t) = T_{max}$ but at $t_0 + \triangle t$, we have $T(t_0 + \triangle t) > T_{max}$. So we must have

$$\frac{dT(t)}{dt}\mid_{t=t_0} = \frac{dT(t)}{dt}|_{T(t)=T_{max}} > 0. \tag{5.29}$$

On the other hand, since $v_e \geq v_k$, from Equation 5.27, we have

$$\begin{aligned} \frac{dT(t)}{dt}|_{t=t_0} &= a((C_0 + C_1T_{max}) \cdot v_k + C_2v_k^3) - bT_{max} \\ &\leq a((C_0 + C_1T_{max}) \cdot v_e + C_2v_e^3) - bT_{max} \\ &= 0, \end{aligned} \tag{5.30}$$

which contradicts Equation 5.29. □

Based on Lemma 5.4.1, we can formulate our second feasibility checking method in the following theorem.

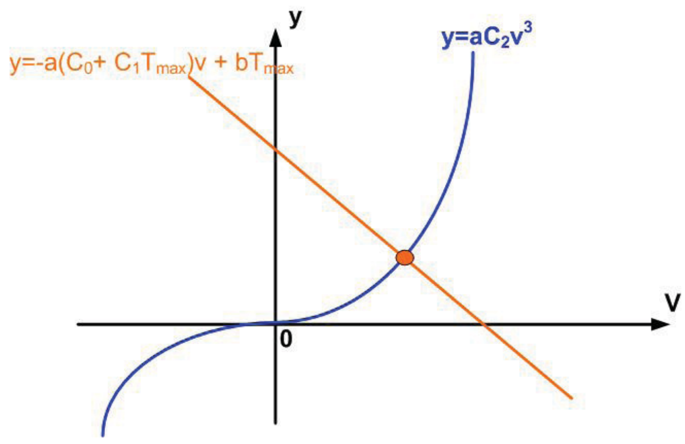


Figure 5.1: Since the slope for the linear function $y = -a((C_0 + C_1 T_{max})) \cdot v + bT_{max}$ is less than zero, there is only one cross point for function $y = -a((C_0 + C_1 T_{max})) \cdot V + bT_{max}$ and function $y = aC_2V^3$. So Equation 5.27 has only one *real* root. [104]

Theorem 5.4.2. *Let $\hat{S}(t)$ be the speed schedule within interval $[0, L]$ that can guarantee the deadlines of $\mathcal{T}$ under the maximal temperature constraint T_{max}, and let s_{max} be the maximal speed in $\hat{S}(t)$. Also let s_{mf} be the highest speed among the processor safe modes. Then if $s_{max} \leq s_{mf}$, when repeating $\hat{S}(t)$ later in the schedule, the temperature will never exceed T_{max}.*

Theorem 5.4.2 can be easily proved following the similar proof as that for Lemma 5.4.1. Note that, as long as the maximal temperature T_{max} and the processor is given, the highest speed (s_{mf}) among the processor safe modes is well determined. Also, it is much less costly to get the maximal speed (s_{max}) in a schedule (such as those generated by the approach in [134]) rather than to get the entire speed schedule for a periodic task set. Therefore this approach can be effectively used for the purpose of design space exploration.

Even though the feasibility condition formulated in Theorem 5.4.2 can be used for cases when the ending temperature of the first hyperperiod is higher than the initial temperature, this feasibility condition is still only a sufficient condition. In

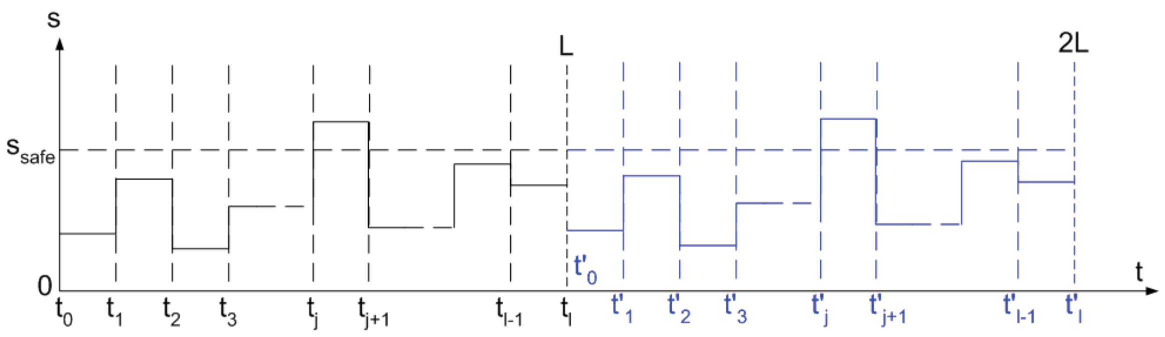


Figure 5.2: A speed schedule within 2 hyperperiods.

other word, when the maximal processor speed is higher than the maximal safe speed of the processor, the schedule may still be feasible under the maximal temperature. In what follows, we introduce the third feasibility condition, which is also a stronger condition and can be used to check the schedulability for these cases.

5.4.4 The Necessary and Sufficient Condition

To guarantee the maximal temperature constraint, we need to make sure that this constraint is not violated at the end point of each hyperperiod and anywhere inside the hyperperiod. It helps then to identify the possible locations within the hyperperiod that this constraint may be violated. In what follows, we first introduce the term, *island interval*.

Definition 5.4.3. *An interval $[t_i, t_j]$ in $\hat{S}(t)$ is called an* island interval *if the processor needs to run in a non-safe processor mode within this interval, or a* non-island interval *otherwise.*

For an island interval, we have the following observation.

Lemma 5.4.4. *Let $[t_1, t_2]$ be an island interval. If for any $t \in [t_1, t_2]$, we have $T(t) \leq T_{max}$, then we have $T(t) \leq T(t_2)$.*

Proof. For any $t \in [t_1, t_2]$, since the processor must be running at a non-safe mode at

t and $T(t) \leq T_{max}$, we have

$$\frac{dT(t)}{dt} \mid_{t \in [t_1, t_2]} > 0, \tag{5.31}$$

Therefore, $T(t)$ monotonically increases with t. So $T(t_2)$ must be the highest within the interval. □

According to Lemma 5.4.4, the highest temperature for an island interval always occurs at its end, given that the maximal temperature constraint is not violated. Therefore, to verify if the temperature constraint is violated within a given hyperperiod, we only need to check temperatures at at the ends of all island intervals, plus the one at the end of the hyperperiod.

Our goal is to make sure that the temperature constraint is not violated during the entire life cycle when executing a periodic task set. Exhaustively checking temperature constraint for all hyperperiods is apparently impossible. In addition, it is not adequate to draw a conclusion that a schedule is feasible under the maximal temperature constraint simply because the temperature constraint is not violated within the first hyperperiod. So, if we want to check temperatures only at the first hyperperiod, additional constraints must be imposed. The following theorem provides such "additional" constraints.

Theorem 5.4.5. *Let the ith interval in $\hat{S}(t)$ be $[t_i, t_{i+1}]$ and let its processor mode be k. Define A_i, B_i such that*

$$A_i = A(k) = a(C_0 v_k + C_2 v_k^3), \tag{5.32}$$

$$B_i = B(k) = (b - aC_1 v_k). \tag{5.33}$$

Let $[t_{(j-1)}, t_j]$ be an arbitrary island interval in $\hat{S}(t)$, and let

$$K_j = exp(-B_0(t_1 - t_0) - \cdots - B_j(t_j - t_{(j-1)})) \quad (5.34)$$

$$K = exp(-B_0(t_1 - t_0) - \cdots - B_l(t_l - t_{(l-1)})) \quad (5.35)$$

where $t_l = L$. Then repeating $\hat{S}(t)$ later in the schedule, the temperature will never exceed T_{max} iff *the following conditions hold:*

- $0 \leq K < 1$;
- $T(L) \leq T_{max}(1 - K)$;
- $T(t_j) \leq T_{max} - \frac{T(L)}{1-K} K_j$.

Proof. See Figure 5.2. Let starting points for intervals in $\hat{S}(t)$ be $t_0, t_1, \cdots, t_{(l-1)}$, respectively. After repeating $\hat{S}(t)$, let the corresponding points during the second hyperperiod be $t'_0, t'_1, \cdots, t'_{(l-1)}$, correspondingly. Note that $t_0 = 0, t'_0 = t_l = L$ and $t'_l = 2L$.

According to Equation 5.25, we have

$$T(t_1) = \frac{A_0}{B_0} + (T(t_0) - \frac{A_0}{B_0})e^{-B_0(t_1 - t_0)}$$

and

$$T(t'_1) = \frac{A_0}{B_0} + (T(t'_0) - \frac{A_0}{B_0})e^{-B_0(t'_1 - t'_0)}.$$

Since $(t'_1 - t'_0) = (t_1 - t_0)$, we have

$$T(t'_1) - T(t_1) = (T(t'_0) - T(t_0))e^{-B_0(t_1 - t_0)}. \quad (5.36)$$

Similarly, we have

$$T(t_2') - T(t_2) = (T(t_0') - T(t_0))e^{-B_0(t_1-t_0)-B_1(t_2-t_1)},$$

$$\cdots$$

Therefore, we have

$$\begin{aligned}
& T(2L) - T(L) \\
= \;& T(t_l') - T(t_l) \\
= \;& (T(L) - T(0)) \sum_{i=1}^{l} e^{-B_{i-1}(t_i-t_{i-1})} \\
= \;& (T(L) - T(0))K. \qquad (5.37)
\end{aligned}$$

In the same way, we can see that

$$T(3L) - T(2L) = (T(2L) - T(L))K \qquad (5.38)$$

$$T(4L) - T(3L) = (T(3L) - T(2L))K \qquad (5.39)$$

$$\cdots$$

Therefore,$T(L) - T(0), T(2L) - T(L), T(3L) - T(2L), \cdots, T(qL) - T((q-1)L)$ form a geometric series and we have

$$T(qL) = T(0) + \frac{(T(L) - T(0))(1 - K^q)}{1 - K}. \qquad (5.40)$$

Since $0 \leq K < 1$, as $q \to \infty$, we have

$$\lim_{q\to\infty} T(qL) = T(0) + \frac{(T(L) - T(0))}{1 - K}. \qquad (5.41)$$

After T_0 is calibrated to 0, $T(qL) \leq T_{max}$ if and only if

$$T(L) \leq T_{max}(1 - K). \tag{5.42}$$

We now need to make sure that the maximal temperature constraint is not violated in any island interval. Let $[t_{j-1}, t_j]$ be an arbitrary island interval in $\hat{S}(t)$. Follow the same procedure as stated above, we have

$$\begin{aligned} & T(L + t_j) - T(t_j) \\ = \;& (T(L) - T(0)) \sum_{i=1}^{j} e^{-B_{i-1}(t_i - t_{i-1})} \\ = \;& (T(L) - T(0))K_j. \end{aligned}$$

Similarly, we have

$$\begin{aligned} T(2L + t_j) - T(L + t_j) &= (T(2L) - T(L))K_j, \\ T(3L + t_j) - T(2L + t_j) &= (T(3L) - T(2L))K_j, \\ &\dots \\ T(qL + t_j) - T((q-1)L + t_j) &= (T(qL) - \\ & \quad T((q-1)L))K_j, \end{aligned}$$

Add all above questions together, we have

$$T(qL + t_j) - T(t_j) = (T(qL) - T(0))K_j. \tag{5.43}$$

Similarly, since $0 \leq K < 1$, as $q \to \infty$, with Equation 5.40, we can get

$$T(qL + t_j) = T(t_j) + \frac{(T(L) - T(0))}{1 - K} K_j. \tag{5.44}$$

So, after T_0 is calibrated to 0, $T(qL + t_j) \leq T_{max}$ if and only if

$$T(t_j) \leq T_{max} - \frac{T(L)}{1 - K} K_j. \tag{5.45}$$

□

Note that, after the speed schedule $\hat{S}(t)$ is defined, K and K_j are well defined. We then can check temperatures at the end of first hyperperiod as well as those at the end of island intervals. $\hat{S}(t)$ is a feasible schedule if the three conditions in Theorem 5.4.5 hold.

5.4.5 Further Discussions

From Theorem 5.4.5 and its proof, we have a number of interesting observations. Corollary 5.4.6, for example, is a straightforward conclusion from proof of Theorem 5.4.5.

Corollary 5.4.6. *If $K > 1$ and $T(L) > T(0)$, the processor temperature will run away and reach infinity.*

Corollary 5.4.6 can be easily proved from Equation 5.40. When $K > 1$,

$$\begin{aligned}\lim_{q\to\infty} T(qL) &= \lim_{q\to\infty} (T(0) + \frac{(T(L) - T(0))(1 - K^q)}{1 - K}) \\ &= \infty.\end{aligned}$$

This implies that the heat generated by the processor exceeds its cooling capability, and the temperature continues to rise until the system breaks down.

On the other hand, when $0 \leq K < 1$, the processor temperature will eventually enter a *stable status* if the system does not break down before that. The *stable status* is formally defined as follows.

Definition 5.4.7. *Assume a processor is running a periodic schedule $\hat{S}(t)$ with period L, the processor temperature is called to be in a* stable status *if for a given threshold, i.e.* $0 < \varepsilon << 1$,

$$|T((i+1)L) - T(iL)| < \varepsilon, \tag{5.46}$$

where $i \geq 0, i \in Z$.

When the processor temperature enters the stable status, the temperature profile does not change much from one hyperperiod to another hyperperiod. Also, from the proof of Theorem 5.4.5, the temperature when the processor is in its stable status can be analytically formulated as follows.

Lemma 5.4.8. *Assume a processor is running a periodic speed schedule $\hat{S}(t)$ with period L. Let $T(0)$ and $T(L)$ be temperatures at $t = 0$ and L, respectively. Then when the processor temperature reaches its stable status, the temperature at the starting (or ending) point of a period, denoted as $T(L')$, can be formulated as*

$$T(L') = T(0) + \frac{(T(L) - T(0))}{1 - K}. \tag{5.47}$$

Note that similar lemmas can also be developed to calculate the temperature at each specific scheduling point, based on Equation 5.44 in the proof of Theorem 5.4.5.

In summary, we introduce three feasibility testing methods (Section 5.4.2, Section 5.4.3, and Section 5.4.4) to verify if a feasible schedule developed within the first hyperperiod is globally feasible or not under the given maximal temperature constraint. The first two, which are based on Theorem 5.3.1 and 5.4.2, are sufficient conditions. The third one, based on Theorem 5.4.5, is a more elaborated necessary and sufficient condition. All three methods take the leakage/temperature dependency into account based on the processor power model formulated in Equation 5.21. In what follows, we use experiments to further study their effectiveness.

5.5 Experiment

The feasibility conditions introduced in the previous section are established based on the leakage power model proposed in Section 5.2, or more specifically, formulated by Equation 5.21. Therefore to study the performance of the feasibility conditions, we need to first validate the leakage model. It is difficult to analyze the accuracy of the leakage model analytically, since the constants C_0 and C_1 are obtained through polynomial approximation methods rather than from some analytical formulas. Similar to empirical experiments done in Chapter 3, we conduct some new experiments to check the accuracy of various leakage models.

Next, we will examine the performance of different schedulability conditions presented before. We also studied what impacts different leakage models may have in schedulability analysis.

5.5.1 Leakage Model Validation

We constructed our processor model based on the work by Liao et al. [77] for a processor using 65nm technology. We assumed that the processor can run in six different active modes, with the corresponding supply voltages as 0.85V, 0.9V, 0.95V, 1.0V, 1.05V and 1.1V. The processor can also be shut down and consumes no energy. For each mode, we set the frequency of the processor in each mode such that it can accommodate the longest delay at the highest temperature (110^oC) based on Equation 5.3, with $\mu = 1.19$, $\eta = 1.2$, and $v_t = 0.3$ [77]. For the thermal constants, we selected $Rth = 0.8K/W$, $Cth = 340J/K$ [119], and ambient temperature as 25^oC.

We studied four different leakage models shown below:

- $Const_A$ The leakage power is a constant that is independent of both temperature and supply voltage. This is the leakage power model used by many previous researches (such as [105]). The constant is determined by the average leakage power consumption at the ambient temperature.

- $Const_H$ The leakage power is also a constant. But the constant is determined by the average leakage power consumption at the highest temperature.

- LK_T The leakage power changes only with the temperature but not the supply voltage. This model is adopted in previous work such as [35, 48, 30].

- $LK_{T\&V}$ The leakage power varies with both the temperature and supply voltage. This is the model proposed in Section 5.2.

We used the analytical formula, i.e. Equation 3.2, to compute the actual leakage power for temperature from 40^oC to 110^oC with step size of 10^oC. The total number of gate, i.e. N_{gate} in Equation 5.5, was set to be 10^6. The dynamic power consumption was determined based on the experimental results reported in [77] on benchmark *gcc*. The corresponding power consumption results were then used to determine the constants in the leakage models.

Specifically, the power consumptions obtained above were used to determined the curve fitting constants C_0, C_1 and C_2 for Model $LK_{T\&V}$, which are listed in Table 7.1. The leakage power consumptions for $Const_A$ and $Const_H$ were defined as the average results at the ambient temperature ($T = 25^oC$) and the highest temperature ($T = 110^oC$), respectively. For Model LK_T, the average leakage power consumption at each temperature was used to derive the corresponding linear function.

Figure 5.3 plots the leakage power consumptions based on the complex non-linear model (i.e. Equation 3.2) and other four models. As we can see from Figure 5.3,

Table 5.1: Processor parameters and constants for Model $LK_{T\&V}$

$V_{dd}(V)$	C_0	C_1	C_2	Frequency
0.00	0.0	0.0	0.0	0.0
0.85	7.3249	0.1666	15.0	0.8010
0.90	8.6126	0.1754	15.0	0.8291
0.95	10.238	0.1846	15.0	0.8553
1.00	12.315	0.1942	15.0	0.8797
1.05	14.998	0.2043	15.0	0.9027
1.10	18.497	0.2149	15.0	1.0

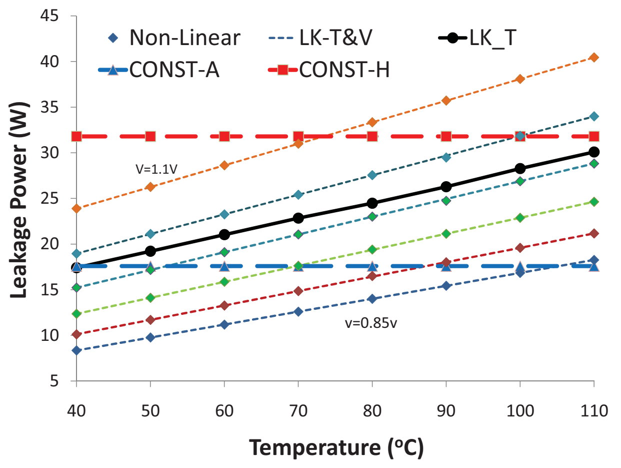
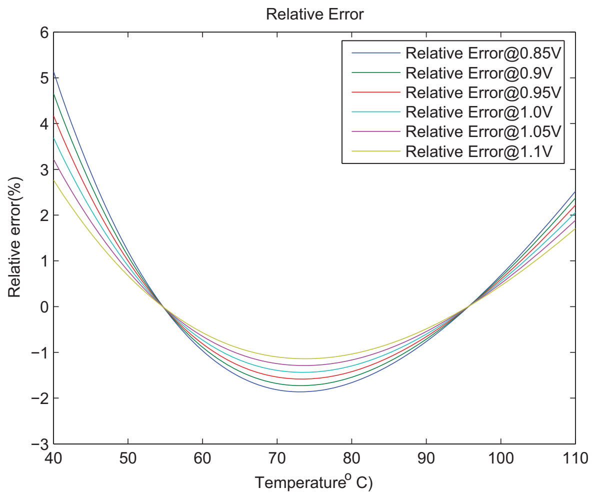


Figure 5.3: The Leakage Power Consumptions Based on Different Leakage Models.



with linear approximation, the leakage power consumptions based on Model $LK_{T\&V}$ match very closely to that by a much more complex method (Equation 3.2). As further shown in Figure 5.5.1, the relative error is less than 5% for 0.85V and less than 3% for 1.1V. The results clearly show that Model $LK_{T\&V}$ is a leakage model with low complexity and very good accuracy.

On the other hand, however, our experimental results also show that if the supply voltage or temperature dependency are not carefully addressed, the leakage model can lead to estimation results deviated far away from the actual values. From Figure 5.3, when the supply voltage is not taken into consideration, the estimation errors by Model LK_T can be as much as 2.08 times higher or 26.7% lower than the actual leakage power consumptions. When also ignoring the leakage/temperature dependency, the estimation errors by Model $Const_A$ and $Const_H$ become even larger, i.e. as much as 3.8 times higher or 56% lower than the actual values. We investigate how significant the leakage power estimation errors may affect schedulability analysis in Section 5.5.3.

5.5.2 The Performance of Feasibility Conditions

We next used the processor model developed above to study the feasibility analysis methods proposed in this chapter. Three feasibility checking methods were implemented and investigated. The first one (namely **EndCheck**), based on Theorem 5.3.1 checks if the temperature at the end of the hyperperiod is no more than the initial temperature. The second one (namely **SafeCheck**) applies Theorem 5.4.2 and uses the processor safe speed to check the feasibility. The third one (namely **Island-Check**) employs Theorem 5.4.5, checking temperatures at the ending points of the first hyperperiod and all island intervals in the first hyperperiod.

The real time tasks were randomly generated with periods distributed evenly in range $[1000, 5000]$ seconds. Deadlines were determined by multiplying periods with a constant, called the *deadline-period ratio*. The execution time for each task was also randomly generated, which is evenly distributed between 1 and its deadline. The feasible speed schedules, generated based on the optimal method to minimize the dynamic energy [134], were used as our test cases. Since the approach in [134] assumes a processor model with continuously variable speed, we always rounded up a processor speed to the next higher available one in our experiments.

In the first set of experiments, we fixed the deadline-period ratio at 0.3 (i.e. *DPratio*=0.3) and varied the peak temperature constraint from 40^oC to 110^oC, with step size of 1^oC. For each peak temperature, we generated 100 test cases that can satisfy deadlines if the temperature factor is not taken into consideration. In the second set of experiments, we fixed the maximal temperature at 50^oC, and varied the deadline-period ratio. The deadline-period ratio was varied from 0.1 to 0.9 with step size of 0.1. All these test cases were then tested using the three methods stated above.

We evaluate the performance of three feasibility conditions using the numbers of schedulable task sets under each feasibility condition, as shown in Figure 5.4 and Figure 5.5. Figure 5.4 and Figure 5.5 show clearly the significant impacts of peak temperature requirement to the schedule's feasibility. Note that in Figure 5.4, when the peak temperature constraint is higher than 58^oC, all 100 schedules randomly generated as above can satisfy the temperature constraints based on **IslandCheck** and **SafeCheck**. When the peak temperature constraint getting tighter, however, the feasibility drops quickly. In Figure 5.4, when the peak temperature is set to 41^oC, more than 30% of the original feasible schedules become infeasible.

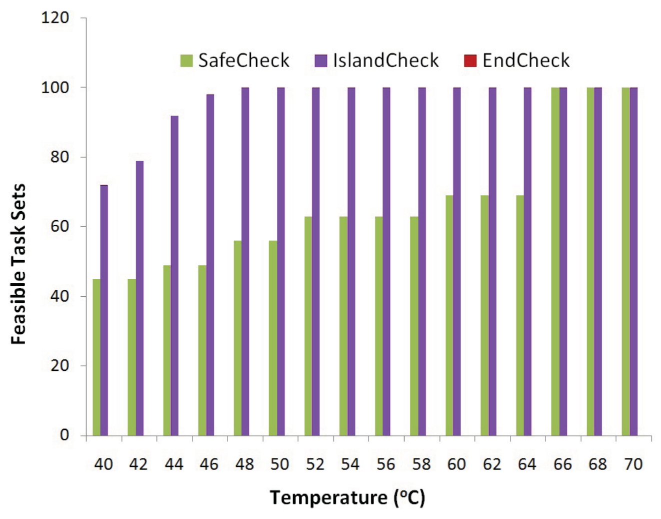


Figure 5.4: Success Rate Under Different Maximal Temperature (deadline-period ratio = 0.3)

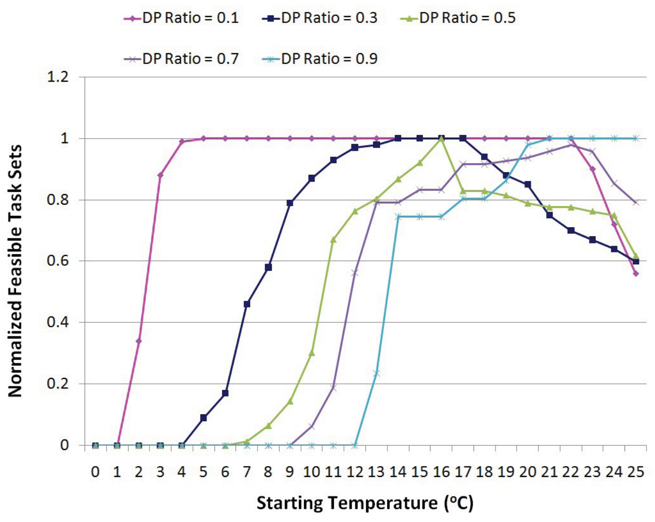


Figure 5.5: Success Rate Under Different Deadline-Period Ratio. ($T_{max} = 50^oC$)

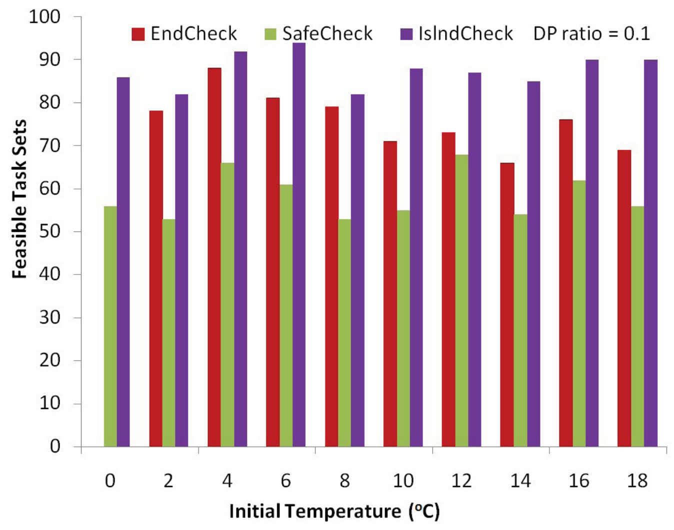


Figure 5.6: Success Rate Under Different Initial Temperature(above ambient temperature). ($T_{max} = 50^oC$ and deadline-period ratio = 0.1)

It is not surprising to see in Figure 5.4 that none of the task sets can be predicted as feasible using **EndCheck**. This is because that, starting from the ambient temperature, the processor temperature always increases to one that is above the ambient temperature after executing tasks, given the thermal settings stated before. Therefore, the usage of this feasibility condition is very limited. Moreover, we can see that **SafeCheck** is pessimistic in predicting the feasibility for a schedule. This is because a schedule occasionally using a speed higher than the processor safe speed can still reach a temperature lower than the required maximal temperature. In Figure 5.4, when the maximal temperature is set to be 57^oC, about 18% of the feasible task sets cannot be properly verified by **SafeCheck**. When the given maximal temperature becomes very high, all processor running modes become safe modes, and thus **SafeCheck** obtains the same results as that by **IslandCheck** in Figure 5.4. Similar conclusions can be drawn from Figure 5.5 when the deadline period ratios are different.

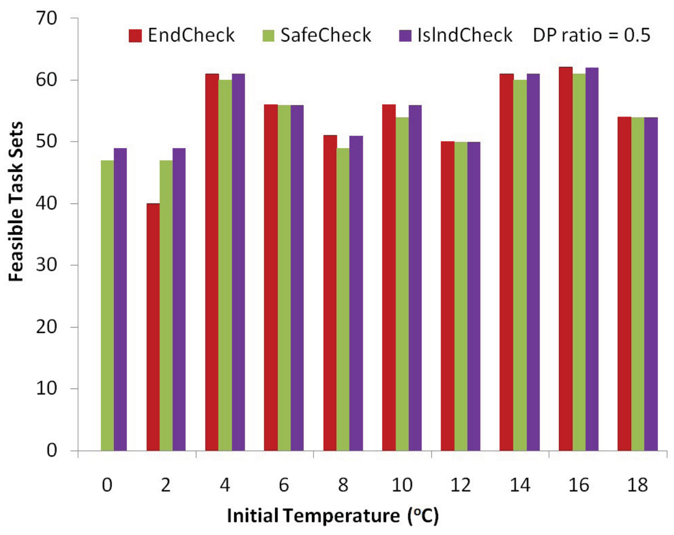


Figure 5.7: Success Rate Under Different Initial Temperature(above ambient temperature). ($T_{max} = 50^oC$ and deadline-period ratio = 0.5)

We next evaluate the success rate of three feasibility conditions with different initial temperature. We fixed the maximum temperature constraint to 50^oC, and deadline-period ratio to 0.1, 0.5 and 0.9. Figure 5.6, 5.7 and 5.8 shows the comparison of success rate by three feasibility methods. In all the figures **X-axis** represents the initial temperature above ambient. From Figure 5.8, we can see that the feasibility by **EndCheck** increases with increasing initial temperature. When the initial temperature becomes as high as 14^oC above ambient, the feasibility of **EndCheck** becomes equal to that of **IslandCheck**. From Figure 5.7, at deadline-period ratio equal to 0.5, **EndCheck** follows **IslandCheck** closely after 4^oC. In Figure 5.6, **EndCheck** is always pessimistic compared to **IslandCheck** when deadline-period ratio is 0.1. The overall result continues to support our claim that the **EndCheck** is a pessimistic method compared to **IslandCheck**. As we can clearly see from our results that it can be very close to **IslandCheck** in many scenarios, but **IslandCheck** is always either same or better than **EndCheck**.

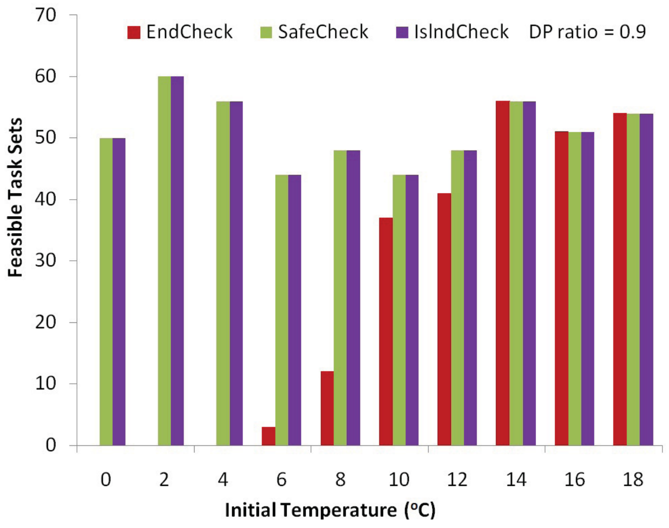


Figure 5.8: Success Rate Under Different Initial Temperature(above ambient temperature). ($T_{max} = 50^oC$ and deadline-period ratio = 0.9)

5.5.3 The Impacts of Different Leakage Models

We further examine how different leakage power models may affect the feasibility analysis results. We used the same test cases generated in Section 5.5.2 and applied **IslandCheck** on all four leakage models introduced in Section 5.5.1. The numbers of feasible task sets are collected and depicted in Figure 5.9. The results shown in Figure 5.9 verify the large discrepancies in terms of task set schedulability caused by the large estimation errors by different leakage models. When assuming the leakage power being a constant at the highest temperature, the leakage model $Const_H$ can lead to a feasibility analysis that is extremely pessimistic. Note that in Figure 5.9 none of the task sets is predicted as schedulable according to $Const_H$ at temperature of 44^oC. According to leakage model $LK_{T\&V}$, however, 74 of the task sets are in fact schedulable.

When assuming the leakage power consumption at the ambient temperature, the feasibility analysis based on leakage model $Const_A$ can be both pessimistic or optimistic. Note that we defined the constant leakage power consumption in $Const_A$

using the average results under different supply voltages at the ambient temperature. This leakage power consumption can thus be over estimated when the actual supply voltage is low or under estimated when the actual supply voltages is high. When the maximal temperature constraint is low, the schedules that employ low processor speed are more likely to satisfy the temperature constraints. As a result, the feasibility analysis results based on $Const_A$ tends to be pessimistic for these test cases. In Figure 5.9, at 40^oC, 82% of the feasible task sets cannot be correctly predicted based on leakage model $Const_A$. As temperature increases, the feasibility analysis results become more and more optimistic. At temperature 49^oC, at least 22% task sets that are predicted as feasible according to $Const_A$ are in fact infeasible according to results based on the leakage model $LK_{T\&V}$.

Even though the leakage/temperature dependency is considered in model LK_T, large estimation errors still exist since the leakage power varies not only with temperature but also supply voltage. As a result, similar to leakage model $Const_A$, the feasibility analysis based on leakage model LK_T can be overly pessimistic or overly optimistic. At 40^oC, as many as 86% of the feasible tasks cannot be properly predicted based on model LK_T, and as many as 15% of the feasible task sets at 53^oC in fact cannot satisfy the temperature constraint. These results clearly demonstrate that the feasibility analysis without appropriately accounting for the leakage power with temperature and supply voltage can deviate far away from the actual results.

5.6 Conclusion

In this chapter, we studied the feasibility checking problem for real-time periodic task sets under the peak temperature constraint. We showed that the traditional scheduling approach, i.e. to repeat the schedule that is feasible through the range of one hyper-period, does not apply any more. We presented three feasibility analysis

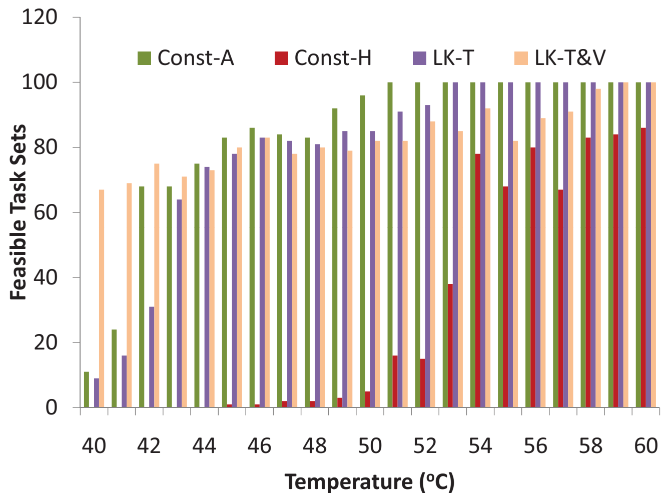


Figure 5.9: Feasible Task Sets Based on Different Leakage Models Under Different Temperature (deadline-period ratio=0.3)

techniques to determine if a period task set can meet deadlines under a given maximal temperature constraint. Our experimental results, based on technical parameters derived from a processor using 65nm technology, showed that our leakage current model have a relative error less than 5%. In addition, our experimental results on the feasibility analysis demonstrated the effectiveness of our methods and clearly highlight the importance to deal with the impacts of leakage/temperature relationship.

CHAPTER 6

Energy Minimization in Multi-Core Processor Systems

In the recent years, advancement in microprocessor architecture design has gain a tremendous momentum. Driven by the growing appetite for high performance under stringent energy constraints, the microprocessor design standard has evolve from the single core to the multi-core architectures. The multi-core systems are gaining widespread popularity and are now the mainstream processor design solution for the high performance computing. However, energy efficiency is still a critical concern in multi-core system designs. It becomes even more challenging when considering the cyclic dependency between leakage power and temperature. Moreover, absence of analytical energy formulation is a fundamental bottleneck in developing effective and efficient system level energy reduction techniques.

In this chapter, we first present a novel method to calculate the energy consumption of a given voltage schedule on a multi-core platform, with the leakage/temperature dependency taken into consideration. Different from the traditional numerical method for energy calculation, this method analytically formulates the overall energy consumption of a given speed schedule. To the best of our knowledge, this is the only work that provides an analytical solution for energy calculation on the multi-core platforms incorporating leakage/temperature dependency into scheduling decisions. Our experiment results demonstrated that this method can achieve a speedup of two orders of magnitude compared with the numerical method, with a relative error of no more than 0.1%.

Next, we combined our energy calculation method with two different task allocation techniques and present new energy minimization(EM) methods, namely *Exhaustive Task Allocation(ETA-EM)* and *Thermal Aware Task Allocation(TATA-EM)*. We compared the efficiency of these methods in minimizing the over all energy of the system. By definition, the *ETA* method is an optimal solution for energy minimization, however it can be computationally expensive with increasing number of processing cores. Our experimental results showed that *TATA-EM* is an computationally inexpensive method and can very well match the optimal solution.

The rest of this chapter is organized as follows. First we discuss the closely related work in Section 6.1. Next, we introduce the system models used in this chapter in Section 6.2. In Section 6.3 we present the formulation of our temperature dynamics and the analytical solution for energy calculation on multi-core systems. We evaluate the accuracy and efficiency of our method in Section 6.4. We present our experimental based comparative study in Section 6.5, and conclude this chapter in Section 6.6.

6.1 Related Work

In the past, extensive research work has been published on both single core and multi-core processor systems for energy-aware designs [134, 75, 68, 16, 132, 57, 85, 17, 150]. A key problem in energy efficient electronic design automation is to calculate the energy consumption for a design alternative. Earlier research, e.g. [134, 75], has been exclusively focused on dynamic energy consumption. Some later research such as that in [68] takes the leakage power into considerations, but assumes that leakage power is constant. Under these assumptions, the calculation of the energy consumption for a given voltage schedule is trivial, since the overall power consumption remains the same as long as a system keeps the same running mode. However, when considering the leakage/temperature dependency, the problem becomes substantially more challenging since the leakage power consumption (and thus the overall power consumption) varies with the temperature, and temperature changes with the power consumption as well. It becomes even more complicated for multi-core platforms when the temperature of one core depends on temperatures from other cores as well.

To calculate the overall energy consumption with leakage/temperatue dependency taken into considerations, one intuitive and commonly adopted approach is to use the numerical method. According to this method, the entire voltage schedule is split into a set of small time intervals such that within each interval the voltage/frequency or temperature of all cores can be regarded as invariant. The temperature and power trace, and thus the energy consumption, for a schedule can thus be obtained accordingly. For example, Liu *et al.* [85] formulated the energy minimization under a peak temperature as a non-linear programming problem, and then employed the above mentioned method to calculate the energy consumption. Bao *et al.* [15] also used the similar approach to keep track of temperature variations in their research on task mapping combined with dynamic voltage frequency scaling (DVFS) to minimize the overall energy consumption for multi-core systems. One major problem of this approach is that the accuracy highly depends on the variation rate of power and temperature. To achieve high accuracy, the length of the interval need to be kept very small and thus the computation cost can be very high.

Huang *et al.* proposed [57] a different approach to calculate the energy consumption. Based on leakage/temperature dependency model proposed in [104], they developed an analytical closed-form energy estimation method for a schedule. However, their work can only be applied for single core platforms only. In another approach, Hanumaiah *et al.* [53] studied the energy efficient problem that optimizes a metric called *performance per watt* (PPW). They transformed this problem as an optimization problem with the objective function as *quasiconcave*. They then used Matlab tools to search for solutions. However, the proposed approach cannot be readily applied to calculate the energy consumption for a given schedule.

6.2 Preliminary

In this section, we present our system models that are used in developing our theoretical framework and validation platform.

6.2.1 Processor and Task Model

The real-time system considered in this chapter consists of M processors, denoted as $\mathbb{P} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_M\}$. Each processor has N *running modes*, each of which is characterized by a pair of parameters (v_k, f_k), where v_k and f_k are the supply voltage and working frequency under mode k, respectively.

Let $\mathbb{S}$ represent a *voltage schedule* or *speed schedule* which indicates how to vary the supply voltage and working frequency for each processor at different time. In this chapter, we use *voltage schedule* and *speed schedule* interchangeably. Let L be the schedule length of $\mathbb{S}$. We define the concept of *state interval* as below:

Definition 6.2.1. *Given a speed schedule $\mathbb{S}$ for a multi-core system, an interval $[t_{q-1}, t_q]$ is called a* state interval *if each processor runs only at one mode during that interval.*

According to Definition 6.2.1, a speed schedule $\mathbb{S}$ essentially consists of a number of non-overlapped state intervals, i.e. Q state intervals, such that

1. $\bigcup_{q=1}^{Q} [t_{q-1}, t_q] = [0, L]$

2. $[t_{q-1}, t_q] \bigcap [t_{p-1}, t_p] = \emptyset$, if $q \neq p$

In addition, for a single state interval $[t_{q-1}, t_q]$, we use κ_q to denote the interval mode, which consists of the running modes of all processors in that interval, i.e. $\kappa_q = \{k_1, ..., k_M\}$ where k_i is the running mode of processor $\mathcal{P}_i$ in that interval.

6.2.2 Power Model

As previously discussed, the overall power consumption (in $Watt$) is composed of dynamic power P_{dyn} and leakage power P_{leak}. In our power model, P_{dyn} is independent of the temperature, while P_{leak} is sensitive to both temperature and supply voltage. The dynamic power consumption of processor $\mathcal{P}_i$ can be formulated as [107]

$$P_{dyn,i} = \epsilon_{k_i} \cdot v_{k_i}^3 \tag{6.1}$$

where v_{k_i} is the supply voltage of processor $\mathcal{P}_i$ and ϵ_{k_i} is a constant, both of which depend on the running mode of processor $\mathcal{P}_i$, i.e. mode k_i.

Similar to the work in [100], we model the leakage power of processor $\mathcal{P}_i$ as follows

$$P_{leak,i} = \big(\sigma_{k_i} + \rho_{k_i} \cdot T_i(t)\big) \cdot v_{k_i} \tag{6.2}$$

where σ_{k_i} and ρ_{k_i} are constants depending on the processor running mode, i.e. mode k_i.

Consequently, the total power consumption of processor $\mathcal{P}_i$ at time t, denoted as $P_i(t)$, can be formulated as:

$$P_i(t) = \big(\sigma_{k_i} + \rho_{k_i} \cdot T_i(t)\big) \cdot v_{k_i} + \epsilon_{k_i} \cdot v_{k_i}^3 \tag{6.3}$$

We rewrite the above power model by separating the elements into temperature independent/dependent parts such that

$$P_i(t) = \lambda_i + \zeta_i \cdot T_i(t) \tag{6.4}$$

where

$$\lambda_i = \sigma_{k_i} \cdot v_{k_i} + \epsilon_{k_i} \cdot v_i^3 \tag{6.5}$$

$$\zeta_i = \rho_{k_i} \cdot v_{k_i} \tag{6.6}$$

Now we can introduce our system power model for multi-core platforms.

$$\mathbf{P}(t) = \mathbf{\Lambda} + \boldsymbol{\zeta}\mathbf{T}(t) \tag{6.7}$$

Note that, to ease our presentation, we use the bold text for a vector/matrix and the normal text for a value, e.g. $\mathbf{T}$ represents a temperature vector while T represents a temperature value.

6.2.3 Thermal Model

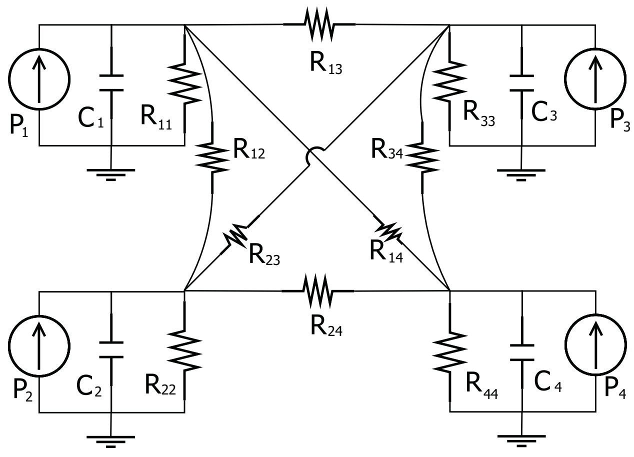


Figure 6.1: Illustration for thermal phenomena on multi-core system

The thermal model used in this chapter is similar to the one used in related researches [113, 123]. Figure 6.1 illustrates the thermal model for a 4-core system. C_i and R_{ij} denote the thermal capacitance (in $Watt/^oC$) of processor $\mathcal{P}_i$ and the thermal resistance (in $J/^oC$) between processor $\mathcal{P}_i$ and $\mathcal{P}_j$, respectively. Let T_{amb} denote the ambient temperature, then in general, the thermal phenomena of processor $\mathcal{P}_i$ can be formulated as

$$C_i \cdot \frac{dT_i(t)}{dt} + \frac{T_i(t) - T_{amb}}{R_{ii}} + \sum_{j \neq i} \frac{T_i(t) - T_j(t)}{R_{ij}} = P_i(t) \tag{6.8}$$

Let $\eta_i = \frac{T_{amb}}{R_{ii}}$ and

$$g_{ij} = \begin{cases} \sum_{j=1}^{M} \frac{1}{R_{ij}}, & \text{if } \ j = i \\ \frac{-1}{R_{ij}}, & \text{otherwise} \end{cases} \tag{6.9}$$

Then the thermal model in equation (6.8) can be rewritten as

$$C_i \cdot \frac{dT_i(t)}{dt} + \sum_{j=1}^{M} g_{ij} \cdot T_j(t) = P_i(t) + \eta_i \tag{6.10}$$

Accordingly, for the entire system, the thermal model can be represented as

$$\mathbf{C}\frac{d\mathbf{T}(t)}{dt} + \mathbf{g}\mathbf{T}(t) = \mathbf{P}(t) + \boldsymbol{\eta} \tag{6.11}$$

Note that $\mathbf{C}$, $\mathbf{g}$ and $\boldsymbol{\eta}$ are all constant that only depend on the multi-core architecture, i.e. capacitance and/or conductance. It is worthy of mentioning that our thermal model in very general and accounts for the heat transfer impacts among different cores. It can be used for thermal analysis for both the temperature transient states as well as the temperature stable state.

6.3 Energy Formulation for Multi-Core Systems

Our goal is to formulate the overall energy consumption for a given schedule. Before we introduce our method, we first present how to formulate the temperature dynamics on multi-core systems analytically.

Note that, by applying the power model (see equation (6.7)) into the thermal model (see equation (6.11)), we can directly obtain that

$$\mathbf{C}\frac{d\mathbf{T}(t)}{dt} + \mathbf{g}\mathbf{T}(t) = \mathbf{\Lambda} + \boldsymbol{\zeta}\mathbf{T}(t) + \boldsymbol{\eta} \tag{6.12}$$

Let $\mathbf{G} = \mathbf{g} - \boldsymbol{\zeta}$, then the above can be rewritten as

$$\mathbf{C}\frac{d\mathbf{T}(t)}{dt} + \mathbf{G}\mathbf{T}(t) = \mathbf{\Lambda} + \boldsymbol{\eta} \tag{6.13}$$

Since $\mathbf{C}$ is the capacitance matrix with none zero values only on the diagonal, we know $\mathbf{C}$ is nonsingular. Thus, the inverse of $\mathbf{C}$, i.e. $\mathbf{C}^{-1}$ exists. Then equation (6.13) can be further represented as

$$\frac{d\mathbf{T}(t)}{dt} = \mathbf{A}\mathbf{T}(t) + \mathbf{B} \tag{6.14}$$

where $\mathbf{A} = -\mathbf{C}^{-1}\mathbf{G}$ and $\mathbf{B} = \mathbf{C}^{-1}(\mathbf{\Lambda} + \boldsymbol{\eta})$. The system thermal model shown in equation (6.14) has a form of first order *Ordinary Differential Equations* (ODE), which has the following solution under constant coefficients:

$$\mathbf{T}(t) = e^{t\mathbf{A}}\mathbf{T}_0 + \mathbf{A}(e^{t\mathbf{A}} - \mathbf{I})\mathbf{B} \tag{6.15}$$

where $\mathbf{T}_0$ is the initial temperature.

Specifically, for a state interval $[t_{q-1}, t_q]$, and let κ_q be the corresponding interval mode, once the temperate at the starting point, i.e. $T(t_{q-1})$, is determined, according to equation (6.15), the ending temperature of that interval, i.e. $T(t_{q-1})$, can be directly formulated as

$$\mathbf{T}(t_q) = e^{\Delta t_q \mathbf{A}_{\kappa_q}}\mathbf{T}(t_{q-1}) + \mathbf{A}_{\kappa_q}^{-1}(e^{\Delta t_q \mathbf{A}_{\kappa_q}} - \mathbf{I})\mathbf{B}_{\kappa_q} \tag{6.16}$$

where $\mathbf{A}_{\kappa_q} = -\mathbf{C}^{-1}\mathbf{G}_{\kappa_q}$, $\mathbf{B}_{\kappa_q} = \mathbf{C}^{-1}(\mathbf{\Lambda}_{\kappa_q} + \boldsymbol{\eta})$, and $\Delta t_q = t_q - t_{q-1}$. Note that since $\mathbf{A}_{\kappa_q}$ and $\mathbf{B}_{\kappa_q}$ are only dependent on the processor running modes, i.e. κ_q, within a state interval $[t_{q-1}, t_q]$, both $\mathbf{A}_{\kappa_q}$ and $\mathbf{B}_{\kappa_q}$ are constant.

Consequently, given a speed schedule $\mathbb{S}$ and the corresponding initial temperature $\mathbf{T}(0)$, with the method introduced above, we can obtain the temperature traces of $\mathbb{S}$ by successively calculating the temperature from one state interval to another.

We now discuss our method to formulate the energy consumption on multi-core systems considering the interdependence of leakage power and temperature. In what follows, we first present an analytical solution to calculated the energy consumption for one state interval. Then we formulate the total energy consumption for the entire speed schedule.

Consider a state interval, i.e. $[t_{q-1}, t_q]$ with initial temperature of $\mathbf{T}(t_{q-1})$. The energy consumption of all processors within that interval can be simply formulated as

$$\mathbf{E}(t_{q-1}, t_q) = \int_{t_{q-1}}^{t_q} \mathbf{P}(t)dt \tag{6.17}$$

Based on our system power model, given by equation (6.7), we have

$$\mathbf{E}(t_{q-1}, t_q) = \Delta t_q \mathbf{\Lambda} + \boldsymbol{\zeta} \int_{t_{q-1}}^{t_q} \mathbf{T}(t)dt \tag{6.18}$$

To calculate $\mathbf{E}(t_{q-1}, t_q)$, we only need to solve $\int_{t_{q-1}}^{t_q} \mathbf{T}(t)dt$.

Recall that the analytical solution for $\mathbf{T}(t)$ is given by equation (6.15). One intuitive approach is therefore to solve $\int_{t_{q-1}}^{t_q} \mathbf{T}(t)dt$ as follows:

$$\begin{aligned}
\int_{t_{q-1}}^{t_q} \mathbf{T}(t)dt &= \int_{t_{q-1}}^{t_q} \big(e^{t\mathbf{A}}\mathbf{T}(t_{q-1}) + \mathbf{A}(e^{t\mathbf{A}} - \mathbf{I})\mathbf{B}\big)dt && (6.19) \\
&= \int_{t_{q-1}}^{t_q} e^{t\mathbf{A}}dt\mathbf{T}(t_{q-1}) + \mathbf{A}\Big(\int_{t_{q-1}}^{t_q} e^{t\mathbf{A}}dt - t\mathbf{I}\Big)\mathbf{B} && (6.20)
\end{aligned}$$

Now the problem becomes how to solve $\int_{t_{q-1}}^{t_q} e^{t\mathbf{A}}dt$. However, we are not aware of any existing method or mathematical tools that can be used to solve the problem of exponential matrix integration. However, based on the definition of our state interval, as all the processors are running on a single mode, $\int_{t_{q-1}}^{t_q} \mathbf{T}(t)dt$ is essentially a constant value, that be can calculated.

To calculate $\int_{t_{q-1}}^{t_q} \mathbf{T}(t)dt$, since all processors are running on a single mode, if we integrate on both sides of equation (6.13) with respect to time t and assume that $\mathbf{G}$ is a non-singular matrix, we can easily find the value of this constant with simple substitution. In what follows, we can formulate the over all energy consumption of the system within a given state interval as,

$$\mathbf{E}(t_{q-1}, t_q) = \Delta t_q \mathbf{\Lambda} + \boldsymbol{\zeta}\mathbf{G}^{-1}\mathbf{Y} \tag{6.21}$$

where

$$\mathbf{Y} = \Delta t_q(\mathbf{\Lambda} + \boldsymbol{\eta}) - \mathbf{C}\Delta\mathbf{T}_q \tag{6.22}$$

Note that given a speed schedule and initial temperature, the temperature at the ends of each state interval can be readily determined using equation (6.15). For a speed schedule $\mathbb{S}$ consisting of Q state intervals, the total system energy consumption under $\mathbb{S}$ can be obtained by summing up the energy consumptions of all state intervals and is given by:

$$E_{total}(\mathbb{S}) = \sum_{q=1}^{Q}\sum_{i=1}^{M} E_i(t_{q-1}, t_q) \tag{6.23}$$

where $E_i(t_{q-1}, t_q)$ can be calculated from equation (6.18).

The computational complexity for our energy calculation of each state interval mainly comes from the matrix multiplications and inversions, with a complexity of $O(M^3)$. To calculate the overall energy consumption for a schedule with Q state intervals, the complexity is thus $O(Q \times M^3)$. In what follows, we use experiments to evaluate the performance of our proposed method.

6.4 Experimental validation

In this section, we validate the accuracy and timing efficiency of our analytical energy calculation method. We compared our approach with the traditional *numerical method*. In what follows, we first introduce the settings for our experiments. We then present and discuss the experimental results.

6.4.1 Experimental Set-Up

V_{dd}	α	β	γ
0	0	0	0
0.8	1.4533	0.0760	6.0531
0.9	2.4173	0.0844	5.8008
1.0	4.0533	0.0936	5.8906

(a) Power/thermal parameters

Parameter	Value
Area Per Core	4 mm^2
Die Thickness	0.15 mm
Heat Spreader Side	20 mm
Heat Sink Side	30 mm
Convection Resistance	0.1 K/W
Convection Capacitance	140 J/K

(b) HotSpot Parameters

Figure 6.2: Experimental Parameters

We performed our experimental simulations based on a 3×3 multi-core system. For convenience, the granularity of the floorplan was restricted to core-level. Our processor model was based on $65nm$ technology as presented in [77]. We assumed that each processor supports 3 active modes with the supply voltage ranging from $0.8V$ to $1.0V$ and step size of $0.1V$. We also set one inactive/sleep mode with supply voltage equal to zero.

We adopted the same thermal parameters as used in work [100] (see Table 6.2(a)). We set the power consumption under the peak temperature constraint of 110^0C. The thermal parameters, including thermal conductance, capacitance etc. were taken from HotSpot-5.02 [4]. The thermal nodes in our thermal model included active layer, interface layer, heat spreader and heat sink. The relevant useful parameters were shown in Table 7.1. We set the ambient temperature T_{amb} as well as the initial temperature T_{amb} as 30^oC.

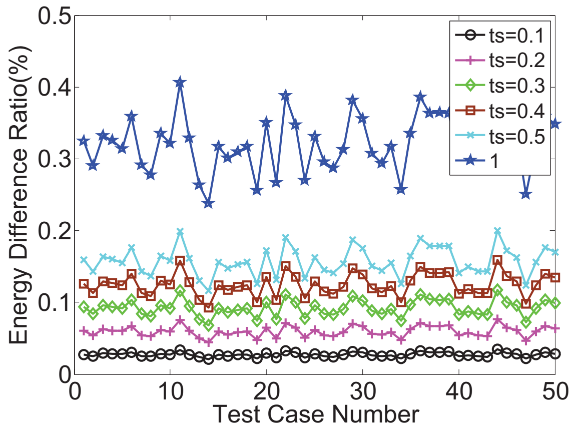
We randomly generated 50 multi-core speed schedules as our test cases. The running mode for each scheduling interval was randomly chosen from $[0, 0.8, 0.9, 1.0]$V (see Table 6.2(a)). The total length of the schedule interval was evenly distributed within $[100, 200]$, and the length of each scheduling interval was evenly distributed within $[50, 80]$. For each test case, our proposed method as well as the traditional numerical method with sampling interval varies from 0.01 second to 1 second were used to calculate the energy consumption. When applying the numerical method, we calculated the leakage power consumption based on the accurate circuit level leakage temperature model, i.e.

$$I_{leak} = I_s \cdot (\mathcal{A} \cdot T^2 \cdot e^{((a \cdot V_{dd} + b)/T)} + \mathcal{B} \cdot e^{(c \cdot V_{dd} + d)}) \tag{6.24}$$

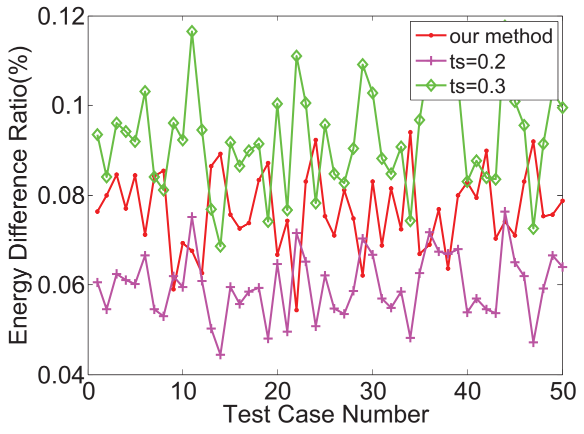
where I_s is the leakage current at certain reference temperature and supply voltage, T is the processor temperature, $\mathcal{A}, \mathcal{B}, a, b, c, d$ are empirically determined constants. All the simulations were conducted on a the *Dell Precision T1500 Desktop Workstation* with CPU type of *Intel i5 750 Quad Core* and memory capacitance of $4GB$.

6.4.2 Accuracy Analysis

In this subsection, we validated the performance of our proposed method in terms of accuracy. To compare the accuracy of different energy estimation approaches, we need to identify the accurate energy consumption for a given speed schedule. We resorted to the numerical method with very short sampling interval to achieve this goal. The question is how short the sampling interval should be.



(a) Numerical method



(b) Our proposed method

Figure 6.3: Accuracy Analysis, Compared with the Numerical Method Under $t_s = 0.01$

In our experiments, we set the sampling intervals from $t_s = 0.01$ second to 0.1 second with step size of 0.01 second and calculated the energy consumption for different schedules. We found that the largest relative energy difference between $t_s = 0.01$ second and $t_s = 0.03$ second is smaller than 0.02%. We thus set the energy estimation results by the numerical method with sampling interval of $t_s = 0.01$ second as our baseline results. We then normalized the energy consumption by other approaches to the baseline results. Figure 6.3(a) shows the relative energy differences of energy consumption estimation results using numerical approach with different sampling intervals, i.e. from $t_s = 0.1$ second to $t_s = 1.0$ second. The relative difference of energy consumption based on our proposed approach and comparable numerical results are presented in Figure 6.3(b).

From Figure 6.3(a), it is not surprising to see that the smaller of the sampling interval, the smaller of the energy difference ratio becomes. For example, when t_s is decreased from 1 to 0.5, the average energy difference ratio is reduced from 0.3% to 0.15%. This is because that the smaller the sampling interval is, the less the temperature can change. Since the numerical method estimates the leakage consumption within an interval assuming temperature within a sampling interval does not change, the estimated leakage energy can be kept small if the sampling interval is small enough.

On the other hand, we can see from Figure 6.3(b) that our proposed method performed very well from the aspect of accuracy. For example, the largest error rate observed in Figure 6.3(b) is no more than 0.1%. As shown in Figure 6.3(b), we can see that our method outperformed the numerical method with $t_s = 0.3$ second for most test cases, with an average energy different ratio 0.73% vs. 0.85% in accuracy. The experimental results clearly show that our proposed approach can achieve very good accuracy in estimating the overall energy consumption for a give speed schedule.

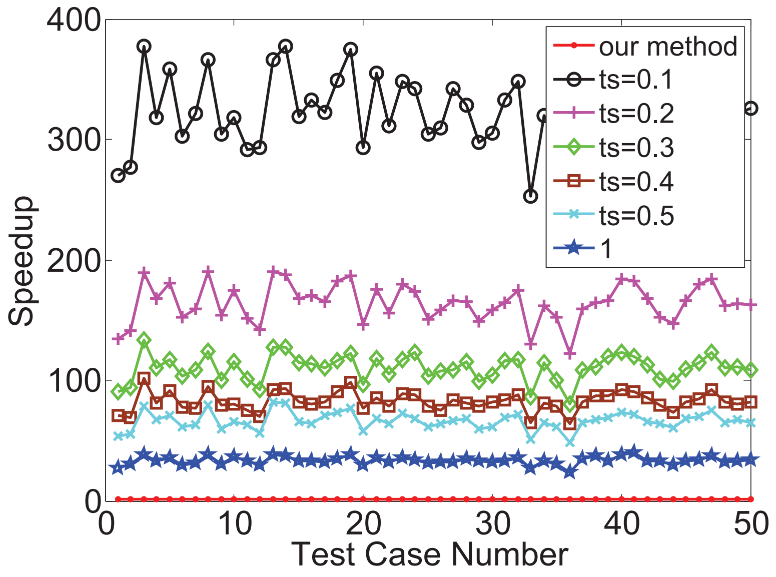


Figure 6.4: Time Efficiency Analysis, Normalized with Our Method

6.4.3 Time Efficiency Analysis

We next want to evaluate the computational efficiency of our proposed method. We collected the CPU time for different approaches for all test cases. We then use the CPU times of our method as the baseline results. The normalized results are shown in Figure 6.4.

From Figure 6.4, we can see that the numerical method with small sampling interval can have a substantially large computational overhead than our approach. For example, as shown in Figure 6.4, our method is more than 300 times (in average) faster than the numerical approach with $t_s = 0.1$, and 25 times (in average) faster than that with $t_s = 1$. Compared with the numerical method with $t_s = 0.03$, which is compatible with our method from the perspective of accuracy, our method can achieve a worst-case speedup no less than 80 times and an average speedup of 110 times. From Figure 6.4, we can conclude that the proposed method was much more time efficient than the numerical approach.

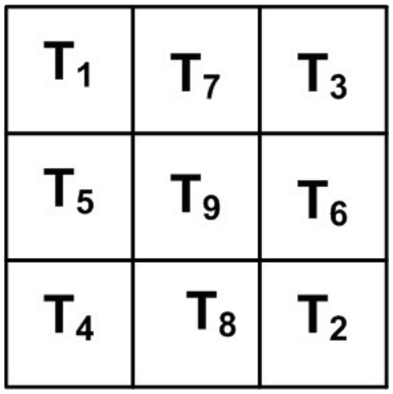


Figure 6.5: *Thermal Aware Task Allocation* on 3X3 Platform

6.5 Energy-Minimization Scheduling Methods

Energy minimization is a critical design challenge for multi-core systems and to address this issue most of the previously published research works, rely upon better task allocation or speed scheduling methods to minimize over all energy consumption of the system [143, 144, 15, 39, 41, 34]. Even though some of these techniques are very effective, but due to the absence of reliable and fast energy calculation methods, the computation complexity limits the efficiency and accuracy of such techniques.

Our energy calculation method offers a fast, accurate and computationally light solution to the above mentioned problem. Following what, we combined our method with two different task allocation techniques and proposed new energy minimization(EM) methods, namely *Exhaustive Task Allocation(ETA-EM)* and *Thermal Aware Task Allocation(TATA-EM)*. For comparison purpose, we also define a third method, *Random Task Allocation(RTA-EM)*. We now present a brief description of these three energy minimization methods.

1. *Exhaustive Task Allocation(ETA-EM)*: Given N processing cores and N real-time tasks to execute, this technique finds the optimal task allocation for energy minimization by exhaustively checking all the possible permutations of task allocation running on the minimum speed that could guarantee real-time

constraints for each task. A strong limitation to this approach is due to total number of permutations which increases with number of processing units N. However, search for close to optimal task allocation can be achieved by using different search optimization or meta-heuristic methods like simulated annealing, genetic algorithms etc. A similar discussion on thermal aware task mapping is presented in [15], where author used genetic algorithm to find optimal or close to optimal task allocation. In our experiments, as our floorplan was no larger than 3×3, we were able to perform exhaustive approach. However, for more dense platforms genetic algorithm can be a useful solution.

2. *Thermal Aware Task Allocation(TATA-EM)*: As temperature plays a critical role in increasing system's energy consumption, we propose a new technique that performs temperature aware task allocation. It assign tasks to cores, such that no two hot tasks are assigned to neighboring cores. This technique uses the knowledge of floor-plan and assign hotter tasks to core with minimum lateral impact to reduce the system temperature and hence energy consumption. For example, given N processing cores and N real-time tasks $(T_1, T_2 ... T_N)$ to execute, this method first calculate the minimum speed that will guarantee the deadline of each task and then based on required power consumption by each task, it assigns hotter tasks to cores at farthest distance from each other to minimize the impact of lateral heat transfer. This is simple static task assignment method with no computation expenses. For instance, in a 3X3 platform, assuming power consumption of each task as $P_1 > P_2 > ... P_9$, the thermal aware task allocation will be done as shown in Figure 6.5.

3. *Random Task Allocation(RTA-EM)*: Given N processing cores and N real-time tasks to execute, this technique randomly assigns tasks to cores and executes task using the minimum speed to guarantee deadlines.

Next, we present our performance evaluation experiments and results. Using the experimental set-up discussed in previous section, we generated 100 random task sets, assuming random execution time for each task ($1 - 100$ seconds). Every task set has a common deadline. Deadline for each set was selected randomly between 1.1 to 2 times of the maximum execution time in that set. Figure 6.6 shows the corresponding experimental results which are normalized with the result of *ETA*.

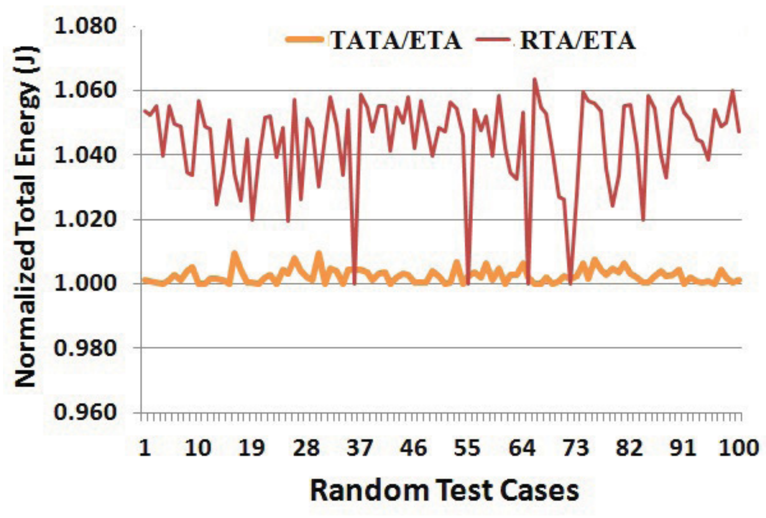


Figure 6.6: Normalized System Energy (J) on 3x3 Multi-Core Platform

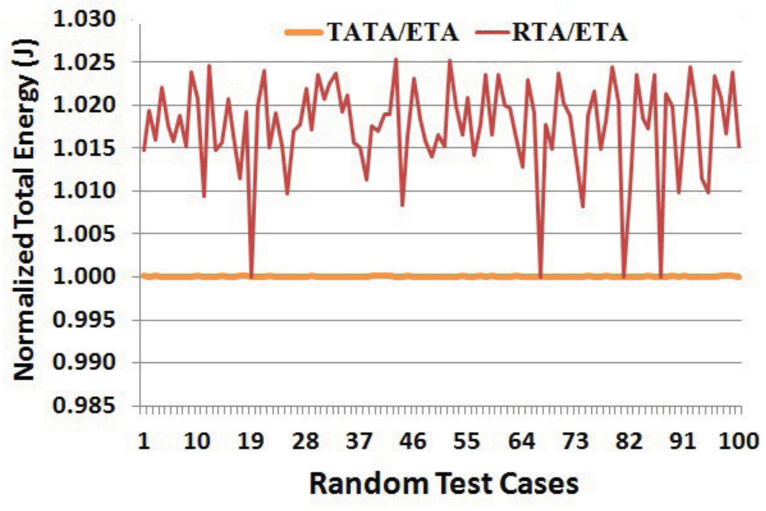


Figure 6.7: Normalized System Energy (J) on 2x3 Multi-Core Platform

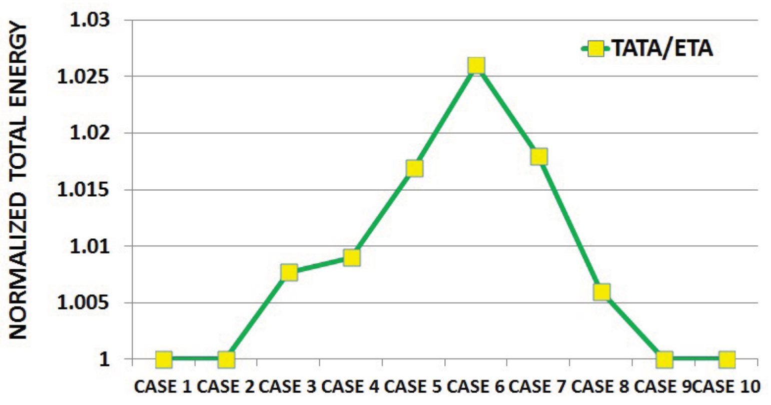


Figure 6.8: *ETA-EM* vs *TATA-EM* on 3x3 Multi-Core Platform

From Figure 6.6, we can clearly see that *ETA-EM* is the optimal solution. In Figure 6.6, *RTA-EM* consumed up to 6% extra energy than that of *ETA-EM*. On the other hand, *TATA-EM* provided a solution matching very close to *ETA-EM* and in much lesser time. The computation time of *ETA-EM* was recorded to be at least 300X more than *TATA-EM*.

We also performed a similar experiment on a different multi-core platform having 6 cores arranged as $2X3$ mesh. The results presented in Figure 6.7, endorses the conclusion of the previous experiment with *TATA-EM* matching very closely to the optimal solution by *ETA-EM*.

In an another set of experiments, we compared only *ETA-EM* and *TATA-EM* on a $3X3$ platform . We assumed only two tasks, i.e. hot task and cool task. The hot task is a task which runs on the peak mode ($V_{dd} = 1.0$) and cool task runs on the minimum mode ($V_{dd} = 0.6$) for equal period of time (50 seconds). We created 10 different test cases, starting with all the cores running only cool task, then replacing cool task by hot task one by one in every new test case. For instance test case 1 has 9 cool tasks to be executed, test case 2 has 1 hot and 8 cool tasks, so on and so forth. The result of this experiment are shown in Figure 6.8. In Figure 6.8, we can see that, when all the tasks are cool or all the tasks are hot, both *ETA-EM* and

TATA-EM are same in energy consumption, which is obvious. In all the other cases, *TATA-EM* results in energy consumption close to energy consumption by *ETA-EM*. The difference in energy consumption increases when the number of hot and cool tasks are equal or close to equal.

Thus, from all our experiments, we can conclude that in multi-core systems, thermal aware task allocation can provide a close to optimal solution for energy minimization with negligible computational expenses compared to optimal *ETA-EM* method. This conclusion is very useful particularly in the futuristic many core platforms and 3D stacked integrated circuits, where energy consumption due to high chip temperature is a critical concern.

6.6 Conclusions

Energy consumption optimization is a critical metric in the design of the multi-core computing systems. It becomes even more challenging in deep sub-micron domain where the leakage power consumption is becoming increasingly significant. In this chapter, we presented a fast and accurate solution for the energy calculation on a multi-core system that take the interdependency of leakage, temperature and supply voltage into consideration. Different from the traditional numerical approach, we developed an analytical formulation of the energy consumption to calculate the overall energy consumption rapidly and accurately. Our system models are rather general and can be easily extended to different platforms and applications. Our experiments showed that the proposed method can achieve a speedup of two orders of magnitude compared with the numerical method, with a relative error no more than 0.1%. Furthermore, employing the analytical energy estimation solution, we presented two new energy minimization methods and compared them for their efficiency in energy reduction.

CHAPTER 7
Conclusions and Future Works

In this chapter, we summarize our contributions presented in this dissertation. We then discuss the possible directions for our future research work.

7.1 Summary

Over the years, the increasing demand for high computing performance and the continuous scaling of semiconductor technology have resulted in the exponentially increased number of transistors integrated on an IC chip. One immediate impact of the increasing transistors on the chip is the exponential increase of power/energy consumption, translating directly into high chip temperature. The dramatically increased power consumption presents grave challenges not only on how to extend battery life for portable devices, but also on how to keep the operational cost in check for power-rich platforms such as data center. The soaring temperature adversely impacts the performance, reliability, life-span and packaging/cooling costs of the computing systems. In addition, due to the diminishing feature size (feature size $< 45nm$), rising chip temperature increases the temperature-dependent leakage power consumption, which is now a serious threat to power and thermal efficiency of modern computing systems. Evidently, power/thermal problems have become the first-class design issues in computer system design, and efforts from every design abstraction level are demanded to address these problems.

In this dissertation, we presented our research work that seeks to address power/thermal issues at the operating system level. Specifically, we presented several novel strategies to use real-time scheduling methods in optimizing the power/thermal efficiency of the real-time computing systems with leakage/temperature dependency taken into considerations. We started our research work by first exploring the fundamental principles on how to employ dynamic voltage scaling (DVS) to reduce the peak operating temperature. We found that, for a specific interval, a real-time schedule using the lowest constant speed is not necessarily the optimal choice in minimizing the peak temperature. Moreover, we identified the scenarios when a schedule using two different speeds can outperform the one using the lowest constant speed. In addition, we also found that, when scheduling a periodic task set, the constant speed schedule is still the optimal solution for minimizing the peak temperature when the temperature is at its *stable status*. We formally proved the validity of our findings using system models that are sensitive to leakage/temperature dependency. These new findings and theorems form the basis for the future study of developing more effective power and thermal aware scheduling techniques for more complicated architectures and real-time systems.

We next developed a novel scheduling method namely '*M-Oscillations*' to reduce the peak operating temperature of a given periodic hard real-time task set on a uniprocessor platform. To minimize the peak temperature, our *M-Oscillations* method uses two-speed schedule similar to our previous work discussed above, and divides the high speed interval and the low speed interval evenly into 'm' sections, and run the processor with the low speed and high speed alternatively. Apparently, an *M-Oscillations* schedule will complete the same workload as the original schedule in one period and thus guarantee the deadline. We formally proved the correctness of the proposed algorithm based on a processor model that can effectively account for the leakage/temperature relationship. The experimental results validated the assumptions of our scheduling method and also demonstrated its effectiveness in terms of feasibility improvement and peak temperature reduction. The proposed *M-Oscillations* scheduling method can reduce peak-temperature of the system up to 14^oC, improving feasibility of the given tasks set by maximum 20%.

Next, we diverted our focus from peak temperature minimization to an equally challenging problem of guaranteeing the feasibility of a given periodic hard real-time tasks set under a given peak temperature constraint. Specifically, we presented three feasibility analysis techniques to determine if a periodic task set can meet deadlines under a given maximal temperature constraint. Our experimental results on the feasibility analysis demonstrated the effectiveness of our methods and clearly highlighted the importance to deal with the impacts of leakage/temperature relationship.

Lastly, we focused our research efforts on energy estimation problem associated with multi-core platforms. In this work, we presented a fast and accurate solution for the energy calculation on a multi-core system that take the interdependency of leakage, temperature and supply voltage into consideration. Different from the traditional numerical approach, we developed an analytical formulation of the energy consumption to calculate the overall energy consumption rapidly and accurately. Our experiments showed that the proposed method can achieve a speedup of two orders of magnitude compared with the numerical method, with a relative error of no more than 0.1%. Our system models are rather general and can be easily extended to different platforms and applications. Moreover, based on our proposed method, we performed a comparative study on different energy minimization techniques for their efficiency in minimizing the overall energy consumption of a multi-core system.

7.2 Future Work

Our existing research is based on a number of simple yet accurate system models (chip-level thermal models, leakage/temperature dependency model, etc), which enable us to conduct a rigorous theoretical research. The effectiveness and efficiency of the proposed techniques were evaluated using architecture/system level simulation platforms and commonly used benchmarks. This approach has been very effective

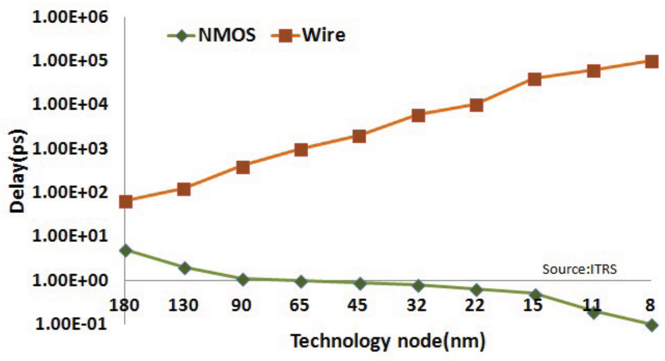


Figure 7.1: source [65]

and can be extended in a number of ways. One most promising future work is to employ this approach for the 3D processor architecture.

The present 2D planar microprocessor architectures integrate numerous functional blocks to achieve increased parallelism and higher performances. This dense integration results in long interconnects that carry on-chip and off-chip communications. Due to the continuous scaling of device feature size, the performance gap between transistor gate delay and interconnect wire delay is growing very fast. From Figure 7.1, we can see that the gap between the transistor gate delay and the delay of a representative 1mm wire has increased from 10X at the 180nm node to 10,000X at the 32nm process node [65]. The gap between transistor and wire delay is expected to increase exponentially with every new technology node generation in the future, as shown in Figure 7.1.

As a result, communication delay penalties will severely affect the performance of microprocessors [19, 130]. Additionally, the interconnect delays result in the unwanted dynamic power consumption, sometimes leading to 30% of the total microprocessor power consumption [19]. Furthermore, the combination of denser functionality combined with degraded interconnect scalability results in larger footprints and inferior packing of integrated circuit chips.

The three dimensional(3D) integrated circuits are foreseen as a promising solution to the long standing problem of scalability and power consumption of interconnects in 2D multi-core platforms [19, 99, 62]. In 3D integration technology multiple active silicon dies are stacked together vertically to reduce on/off chip interconnects' lengths. The reduction in interconnect lengths improved the performance and reduces the power consumption overheads on interconnects. Moreover, 3D integration provides cost effective flexibility in design and better packing of integrated circuits systems. However, as there is no "free lunch", 3D integration technology also faces several severe challenges, on top of which are the thermal challenges.

Thermal issues are the primary concern in 3D ICs. Due to stacking of active layers, the power density of the chip increases linearly with the number of stacked layers [86]. This results in soaring chip temperatures. For instance, with 3D implementation of a 2 layer Alpha-like processor 17-20^oC rise in the peak temperature was recorded compared to its planar design [62]. Furthermore, the impact of increased temperature grows many folds in 3D ICs due to strong thermal relation between vertically aligned neighbors. The vertical heat conductance in 3D stacks is at least 16X times compared to lateral heat flow. This strong thermal correlation elevates the thermal issues by creating more hot spots [147].

The adverse impact of rising chip temperature on the leakage power consumption, reliability, performance and cooling/packging is already well established in 2D planar designs [22]. It will degrade further in 3D implementation. For example, in [110], authors studied the relation between leakage power and temperature and showed that the increase in the leakage power consumption can result up to 18.6% increase in the peak temperature. Moreover, as temperature has a strong relation with reliability, it is shown that 30% variation in process parameters can result up to 20X increased leakage power consumption [28]. In order to gain the first hand experience with the

Table 7.1: HotSpot Parameters and Floorplan

Parameter	*Value*
Total Cores	8 (2x2x2)
Area per Core	4 mm^2
Die Thickness	0.15 mm
Heat Spreader Side	20 mm
Heat Sink Side	30 mm
Convection Resistance	0.1 K/W
Convection Capacitance	140 J/K
Interlayer Material Thickness	0.02mm
Interlayer Material Resistivity	0.25 mK/W
Ambient Temperature	45^oC
Sampling Interval	100 ms

thermal dynamics of 3D ICs, we conducted a series of experiments.

Experimental Set-Up: Similar to the practical simulation set-up discussed in Section 4.4.2, we used a combination of SimpleScalar [2, 13] and Wattch [3, 27] to generate the power trace of a gcc integer benchmark program from SPEC CPU2000. To capture thermal characteristics in a 3D platform, we used a new version of HotSpot simulator i.e. HotSpot-5.02 (HotSpot detailed 3D(default) extension) [40, 4]. This version of HotSpot is capable of modeling multiple active layers stacked together along with the thermal impact of through-silicon-vias(TSV) connecting the layers. We configured model parameters of HotSpot as shown in Table 7.1. For the remaining parameters we used the default values [4]. Our 3D multi-core platform consists of a total of 8 homogenous processing cores distributed on two layers with each active layer having 4 cores as a 2x2 mesh, as shown in Figure 7.2. In our experiments, we use a single task i.e. gcc integer benchmark program.

Experiments and Results: Through our experiments we wanted to study the impact of vertical heat flow and lateral heat flow on the temperature profiles of each core on a practical 3D multi-core platform. We conducted three different experiments: (CASE 1) the task is assigned to core 8 (Figure 7.2) and the rest of the cores are idle,

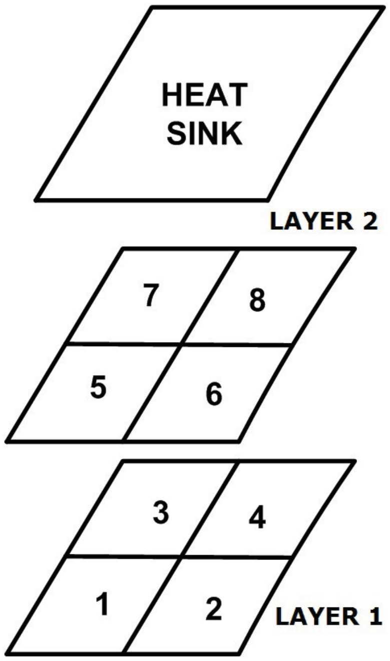


Figure 7.2: A 3D Stacked Multi-core Architecture

(CASE 2) the task oscillates among the four cores on layer 2 for an equal number of cycles, and (CASE 3) the task oscillates between two vertically stacked cores i.e. core 4 and core 8.

Figures 7.3, 7.4 and 7.5 shows the results of the three experiments. From Figure 7.3, we can make two important observations. First, the temperature of core 4 which is vertically connected to core 8 continues to be the maximum temperature. This clearly demonstrates the large impact of high vertical conductance over lateral conductance. Second, even though all the cores other than core 8 are idle temperature continues to increase in all of them. This result suggests that even though vertical conductance is very large compared to lateral conductance, but ignoring lateral conductance in high-level system models is not appropriate. Furthermore, when comparing the results of CASE 2 (Figure 7.4) and CASE 3 (Figure 7.5) with CASE 1(Figure 7.3), we can see that oscillating task among multiple cores result in the higher peak temperature compared to the task assigned to the coolest core.

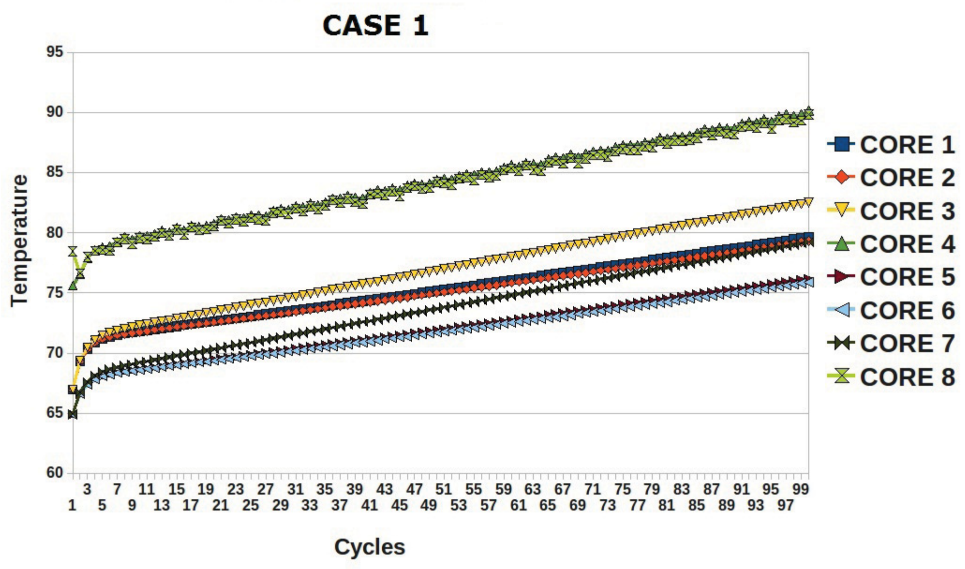


Figure 7.3: Peak Temperature Dynamics on each Core, CASE 1

Based on our background on 2D single core and multi-core platforms, combined with observations we recorded through our empirical study, we can established the corresponding 3D system level power and thermal models and extend our research on power/thermal problems to 3D architecture. Furthermore, the 3D integration method provides a great opportunity for designers to leverage upon the capabilities of heterogeneous integration of active devices. How to explore opportunities in floor-planning, cooling options, architecture variations to form the effective power/thermal solutions for 3D architectures is an interesting problem and needs further study.

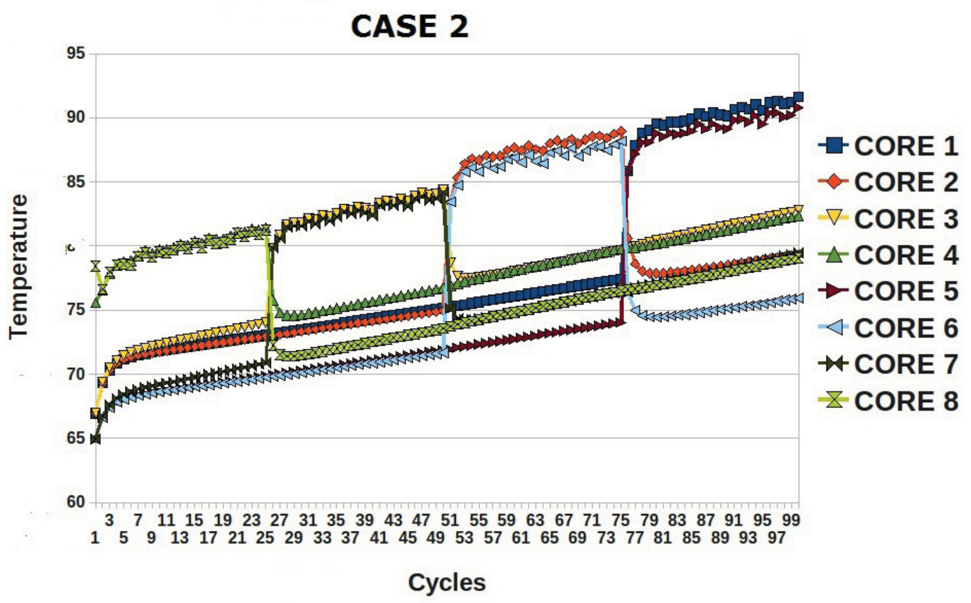


Figure 7.4: Peak Temperature Dynamics on each Core, CASE 2

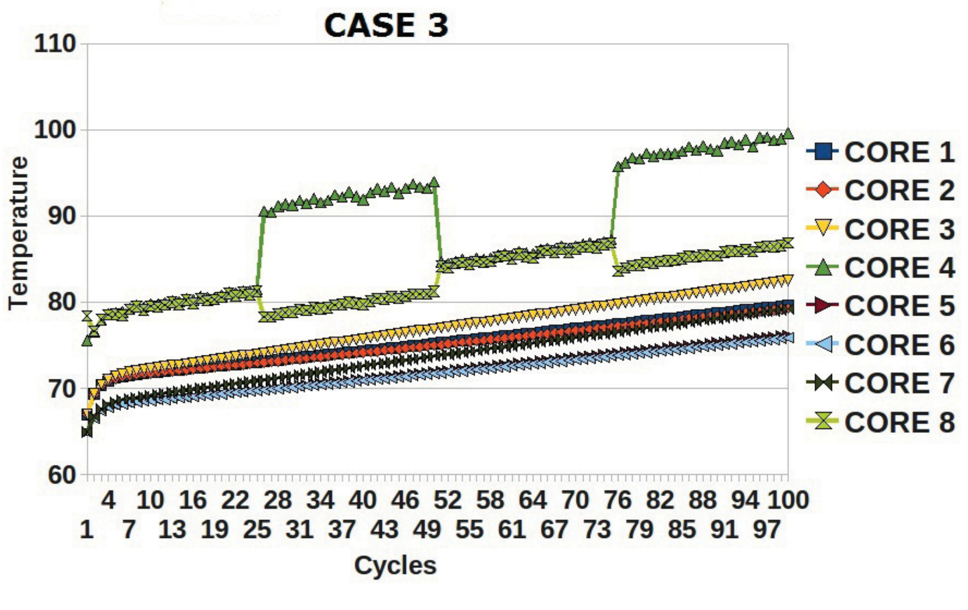


Figure 7.5: Peak Temperature Dynamics on each Core, CASE 3

REFERENCES

[1] Arbitron and edison research media. the infinite dial 2008: Radios digital platforms.

[2] Simplescalar llc. page http://www.simplescalar.com/.

[3] Wattch. pages http://www.eecs.harvard.edu/ dbrooks/wattch–form.html.

[4] Hotspot 5.02 temperature modeling tool. *University of Virgina*, page http://lava.cs.virginia.edu/HotSpot, 2011.

[5] T. F. Abdelzaher, V. Sharma, and C. Lu. A utilization bound for aperiodic tasks and priority driven scheduling. *IEEE Transactions on Computers*.

[6] A. Abdollahi, F. Fallah, and Massoud. Runtime mechanisms for leakage current reduction in cmos vlsi circuits. In *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on*, pages 213 – 218, 2002.

[7] E. P. Agency. Report to congress on server and data center energy efficiency. page http://www.energystar.gov, 2007.

[8] B. Andersson. Global static-priority preemptive multiprocessor scheduling with utilization bound 38%. In T. Baker, A. Bui, and S. Tixeuil, editors, *Principles of Distributed Systems*, volume 5401 of *Lecture Notes in Computer Science*, pages 73–88. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-92221-6_7.

[9] B. Andersson, S. Baruah, and J. Jonsson. Static-priority scheduling on multiprocessors. In *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE*, pages 193 – 202, Dec 2001.

[10] B. Andersson and J. Jonsson. Fixed-priority preemptive multiprocessor scheduling: to partition or not to partition. In *RTCSA '00: Proceedings of the Seventh International Conference on Real-Time Systems and Applications (RTCSA'00)*, page 337, 2000.

[11] A. Andrei, P. Eles, O. Jovanovic, M. Schmitz, J. Ogniewski, and Z. Peng. Quasi-static voltage scaling for energy minimization with time constraints. *IEEE Transaction on VLSI systems*, 19(1):10–23, 2011.

[12] F. Assaderaghi, D. Sinitsky, S. A. Parke, J. Bokor, P. Ko, and C. Hu. Dynamic threshold-voltage mosfet (dtmos) for ultra-low voltage vlsi. *IEEE Trans. on Elec. Dev.*, 44(3):414–422, Mar 1997.

[13] T. Austin, E. Larson, and D. Ernst. Simple scalar: an infrastructure for computer system modeling. *IEEE computer society*, 35, 2002.

[14] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1):1–39, 2007.

[15] M. Bao, A. Andrei, P. Eles, and Z. Peng. Temperature-aware voltage selection for energy optimization. In *Design Automation Conference, 2008. DAC '08. 45th ACM/IEEE*, pages 1083 –1086, 2008.

[16] 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 *Design Automation Conference*, pages 490–495, 2009.

[17] M. Bao, A. Andrei, P. Eles, and Z. Peng. Temperature-aware idle time distribution for energy optimization with dynamic voltage scaling. *Date*, pages 21–26, 2010.

[18] C. Belady. In the data center, power and cooling costs more than the it equipment it supports. *Electronics Cooling*, 23(1), 2007.

[19] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. Loh, D. McCauley, P. Morrow, D. Nelson, D. Pantuso, P. reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die stacking (3d) microarchitecture. In *International symposium on microarchitecture*, pages 201–206, 2006.

[20] J. R. Black. Electromigrationa brief survey and some recent results. *IEEE Trans. Electron Devices*, 16(4):338347, 1969.

[21] J. R. Black. Failure mechanisms and models for semiconductor devices. *Joint Electron Device Engineering Council, Tech. Rep.*, 2003.

[22] S. Borkar. Thousand core chips: a technology perspective. In *DAC*, pages 746–749, 2007.

[23] S. Borkar. Gigascale integration challenges and opportunities. *Intel Software Network*, 2008.

[24] S. Borkar and A. A. Chen. The future of microprocessors. *Communications of the ACM*, 54(5):67–77, 2011.

[25] K. G. Brill. The invisible crisis in the data center: The economic meltdown of moore's law. *Uptime Institute*, 2007.

[26] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA '01: Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, pages 17–28, 2001.

[27] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimization. *ISCA*, 6, 2000.

[28] K. Chakraborty and S. Roy. Rethinking threshold voltage assignment in 3d multicore designs. In *International conference on VLSI design 2010*, pages 375–380, 2010.

[29] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. In *DATE*, pages 288–293, 2008.

[30] T. Chantem, X. S. Hu, and R. Dick. Online work maximization under a peak temperature constraint. *ISLPED*, pages 105–110, 2009.

[31] V. Chaturvedi, H. Huang, and G. Quan. Leakage aware scheduling on maximal temperature minimization for periodic hard real-time systems. *ICESS*, pages 1802–1809, 2010.

[32] J. Chen, C. Hung, and T. Kuo. On the minimization fo the instantaneous temperature for periodic real-time tasks. *RTAS*, pages 236–248, 2007.

[33] J.-J. Chen, H.-R. Hsu, K. Chuang, C. Yang, A. Pang, and T.-W. Kuo. Multiprocessor energy-efficient scheduling with task migration considerations. In *ECRTS*, pages 101 – 108, 2004.

[34] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo. Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems. In *RTAS*, pages 408–417, 2006.

[35] J.-J. Chen, S. Wang, and L. Thiele. Proactive speed scheduling for real-time tasks under thermal constraints. *RTAS*, pages 141–150, 2009.

[36] J. Choi, C. Chen, H. Franke, H. Hamann, A. Weger, and P. Bose. Thermal-aware task scheduling at the system software level. In *ISLPED*, pages 213–218, 2007.

[37] M. A. Cirit. Transistor sizing for cmos circuits. In *Design Automation Conference.*

[38] A. Cohen, F. Finkelstein, A. Mendelson, R. Ronen, and D. Rudoy. On estimating optimal performance of cpu dynamic thermal management. *IEEE Computer Architecture Letter*, 2(1):6–9, 2003.

[39] A. Coskun, D. Atienza, T. Rosing, T. Brunschwiler, and B. Michel. Energy-efficient variable-flow liquid cooling in 3d stacked architectures. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pages 111 –116, march 2010.

[40] A. Coskun, J. Ayala, D. Atienza, T. Rosing, and Y. Leblebici. Dynamic thermal management in 3d multicore architectures. In *Design, Automation, and Test in Europe (DATE)*, pages 1410–1415, 2009.

[41] A. Coskun, T. Rosing, K. Whisnant, and K. Gross. Temperature-aware mpsoc scheduling for reducing hot spots and gradients. In *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*, pages 49 –54, march 2008.

[42] P. de Langen and B. Juurlink. Leakage-aware multiprocessor scheduling. *Journal of Signal Processing Systems*, 2007.

[43] R. H. Dennard, F. Gaensslen, H. Yu, V. L. Rideout, E. Bassous, and A. R. Leblanc. Design of ion-implanted mosfets with very small physical dimensions. *IEEE Journal of solid state circuits*, 9(5):256–268, october 1974.

[44] S. Devadas and S. Malik. A survey of optimization techniques targeting low power vlsi circuits. In *Design Automation Conference*.

[45] N. Eagle and A. Pentland. Social serendipity: Mobilizing social software. *IEEE Pervasive computing*, 4(2):28–34, 2005.

[46] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Power challenges may end the multicore era. *Communications of the ACM*, 56(2):93–102, 2013.

[47] M. Fan and G. Quan. Harmonic semi-partitioned scheduling for fixed-priority real-time tasks on multi-core platform. In *Design Automation and Test in Europe, DATE'12*.

[48] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele. Thermal-aware global real-time scheduling on multicore systems. In *RTAS*, pages 131–140, 2009.

[49] F. Gao and J. P. Hayes. Exact and heuristic approaches to input vector control for leakage power reduction. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 527–532, 2004.

[50] F. Gruian. System-level design methods for low-energy architectures containing variable voltage processors. In *First International Workshop on Power-Aware Computer Systems*, pages 1 – 12, 2000.

[51] N. Guan, M. Stigge, W. Yi, and G. Yu. Fixed-priority multiprocessor scheduling with liu and layland's utilization bound. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, pages 165 –174, April 2010.

[52] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. *Intel Technology Journal*, 5(1), 2001.

[53] V. Hanumaiah and S. Vrudhula. Energy-efficient operation of multi-core processors by dvfs, task migration and active cooling. *Computers(TC), IEEE Transactions on*, pages 1–14, 2012.

[54] V. Hanumaiah, S. Vrudhula, and K. Chatha. Maximizing performance of thermally constrained multi-core processors by dynamic voltage and frequency control. pages 310–313, 2009.

[55] V. Hanumaiah, S. Vrudhula, and K. Chatha. Performance optimal online dvfs and task migration techniques for thermally constrained multi-core processors. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(11):1677 –1690, nov. 2011.

[56] L. He, W. Liao, and M. R. Stan. System level leakage reduction considering the interdependence of temperature and leakage. *DAC*, pages 12–17, 2004.

[57] H. Huang and G. Quan. Leakage aware energy minimization for real-time systems under the maximum temperature constraint. *DATE*, pages 1–6, 2011.

[58] H. Huang and G. Quan. Throughput maximization for periodic real-time systems under the maximal temperature constraint. In *DAC (accepted)*, 2011.

[59] H. Huang, G. Quan, and J. Fan. Leakage temperature dependency modeling in system level analysis. *ISQED*, pages 447–452, 2010.

[60] W. Huang, M. Allen-Ware, J. Carter, E. Cheng, K. Skadron, and M. Stan. Temperature-aware architecture: Lessons and opportunities. *IEEE Micro*, 31(3):82–86, 2011.

[61] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam. Compact thermal modeling for temperature-aware design. In *DAC*, pages 878–883, 2004.

[62] W. Hung. Interconnect and thermal-aware floorplanning for 3d microporcessors. In *ISQED*, pages 98–104, 2006.

[63] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. *SODA*, 2003.

[64] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. *ISLPED*, pages 197–202, 1998.

[65] ITRS. *International Technology Roadmap for Semiconductors (2011 Edition)*. International SEMATECH, Austin, TX., http://public.itrs.net/.

[66] R. Jayaseelan and T. Mitra. Temperature aware task sequencing and voltage scaling. *ICCAD*, pages 618–623, 2008.

[67] R. Jejurikar and R. Gupta. Procrastination scheduling in fixed priority real-time systems. *LCTES*, 2004.

[68] R. Jejurikar, C. Pereira, and R. Gupta. Dynamic slack reclamation with procrastination scheduling in real-time embedded systems. *DAC*, 2005.

[69] S. Kato and N. Yamasaki. Semi-partitioned fixed-priority scheduling on multiprocessors. In *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, pages 23 –32, April 2009.

[70] C. Kim and K. Roy. Dynamic vth scaling scheme for active leakage power reduction. *DATE*, pages 163–167, 2002.

[71] W. Kim, J. Kim, and S. Min. Dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis. *DATE*, pages 788–794, 2002.

[72] P. Kumar and L. Thiele. Cool shapers: Shaping real-time tasks for improved thermal guarantees. *DAC*, pages 468–473, 2011.

[73] P. Kumar and L. Thiele. Thermally optimal stop-go scheduling of task graphs with real-time constraints. *ASPDAC*, pages 123–128, 2011.

[74] K. Lakshmanan, R. Rajkumar, and J. Lehoczky. Partitioned fixed-priority preemptive scheduling for multi-core processors. In *Real-Time Systems, 2009. ECRTS '09. 21st Euromicro Conference on*, pages 239 –248, July 2009.

[75] C.-H. Lee and K. Shin. On-line dynamic voltage scaling for hard real-time systems using the edf algorithm. In *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International*, pages 319 – 335, 2004.

[76] J. Li, M. Qiu, J. Niu, Y. Zhu, and T. Chen. Real-time constrained task scheduling in 3d chip multi-processor to reduce peak temperature. In *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pages 170–176, 2010.

[77] W. Liao, L. He, and K. Lepak. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(7):1042 – 1053, 2005.

[78] C. H. Lim, W. Daasch, and G. Cai. A thermal-aware superscalar microprocessor. In *Quality Electronic Design, 2002. Proceedings. International Symposium on*, pages 517 – 522, 2002.

[79] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, January 1973.

[80] G. Liu, M. Fan, and G. Quan. Neighbor-aware dynamic thermal management for multi-core platform. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 187 –192, march 2012.

[81] J. Liu. *Real-Time Systems*. Prentice Hall, 2000.

[82] S. Liu and M. Qiu. Thermal-aware scheduling for peak temperature reduction with stochastic workloads. *RTAS WiP*, 2010.

[83] S. Liu, J. Zhang, Q. Wu, and Q. Qiu. Thermal-aware job allocation and scheduling for three dimensional chip multiprocessor. In *International Symposium on Quality Electronic Design (ISQED), 2010*, pages 390–398, 2010.

[84] Y. Liu, R. P. Dick, L. Shang, and H. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. *DATE*, pages 1526–1531, 2007.

[85] Y. Liu, H. Yang, R. P. Dick, H. Wang, and L. Shang. Thermal vs energy optimization for dvfs-enabled processors in embedded systems. In *ISQED*, pages 204–209, 2007.

[86] G. Loh, Y. Xie, and B. Black. Processor design in 3d die-stacking technologies. volume 27, pages 31–48, 2007.

[87] C. Lung, Y. Ho, D. Kwai, and S. Chang. Thermal-aware online task allocation for 3d multi-core processor throughput optimization. In *Design, Automation, and Test in Europe (DATE)*, pages 1–6, Grenoble, France, 2011.

[88] C. A. Mack. Fifty years of moore's law. *IEEE transactions on semiconductor manufacturing*, 24(2):202–207, may 2011.

[89] H. Mahmoodi, V. Tirumalashetty, M. Cooke, and K. Roy. Ultra low-power clocking scheme using energy recovery and clock gating. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(1):33 –44, jan. 2009.

[90] J. Markoff. Intel's big shift after hitting technical wall. *New York Times*, 2004.

[91] G. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, 38(8):114–117, may 1965.

[92] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. D. Micheli. Temperature-aware processor frequency assignment for mpsocs using convex optimization. In *CODES+ISSS*, pages 111–116, 2007.

[93] P. J. Nahin. *The Story of* $\sqrt{-1}$. Princeton University Press, Boston, 1998.

[94] L. Niu and G. Quan. Reducing both dynamic and leakage energy consumption for hard real-time systems. *CASES'04*, Sep 2004.

[95] M. Pedram and J. Rabaey. *Power Aware Design Methodologies.* Kluwer acdemic, 2002.

[96] S. Perathoner, K. Lampka, N. Stoimenov, L. Thiele, and J. J. Chen. Combining optimistic and pessimistic dvs scheduling: An adaptive scheme and analysis. *ICCAD*, pages 131–139, 2010.

[97] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *SOSP*, 2001.

[98] R. Prasher, J. Chang, I. Sauciuc, S. Narasimhan, D. chau, G. Chrysler, A. Myers, S. Prstic, and C. Hu. Nano and micro technology-based next-generation package-level cooling solutions. *Intel Technology Journal*, 9(4), 2005.

[99] K. Puttaswamy and G. H. Loh. Thermal herding: Microarchitecture techniques for controlling hotspots in high-performance 3d-integrated processors. In *HPCA '07: Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture*, pages 193–204, 2007.

[100] G. Quan and V. Chaturvedi. Feasibility analysis for temperature- constraint hard real-time periodic tasks. *IEEE Transaction on Industrial Informatics*, 6(3):329–339, 2010.

[101] G. Quan and X. Hu. Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 828–833, 2001.

[102] G. Quan and X. Hu. Minimum energy fixed-priority scheduling for variable voltage processor. In *DATE '02: Proceedings of the conference on Design, automation and test in Europe*, page 782, 2002.

[103] G. Quan, L. Niu, B. Mochocki, and X. Hu. Fixed priority scheduling for reducing overall energy on variable voltage processors. *RTSS'04*, pages 309–318, Dec 2004.

[104] G. Quan and Y. Zhang. Leakage aware feasibility analysis for temperature-constrained hard real-time periodic tasks. *ECRTS*, pages 207–216, 2009.

[105] G. Quan, Y. Zhang, W. Wiles, and P. Pei. Guaranteed scheduling for repetitive hard real-time tasks under the maximal temperature constraint. *ISSS+CODES*, pages 267–272, 2008.

[106] V. C. G. Quan. Leakage conscious dvs scheduling for peak temperature minimization. In *Asia and South Pacific Design Automation Conference*, pages 135–140.

[107] J. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits: A Design Perspective.* Prentice Hall, 2003.

[108] T. Raja, V. D. Agrawal, and M. L. Bushnell. Transistor sizing of logic gates to maximize input delay variability. *Journal of Low Power Electronics*, 2(1):121–128, january 2006.

[109] R. Rao, S. Vrudhula, C. Chakrabarti, and N. Chang. An optimal analytical solution for processor speed control with thermal constraints. In *ISLPED*, pages 292–297, 2006.

[110] S. Roy and K. Chakraborty. A convex optimization framework for leakage aware thermal provisioning in 3d multicore architectures. In *International Symposium on Quality Electronic Design (ISQED), 2010*, pages 804–811, 2010.

[111] M. Santarini. Thermal integrity: A must for low-power ic digital design. *EDN*, pages 37–42, 2005.

[112] K. Seta, H. Ham, T. Kurcda, M. Kakumu, and T. Sakurai. 50% active-power saving without speed degradation using standby power reduction (spr) circuit. In *IEEE International Solid-State Circuits Conference*.

[113] S. Sharifi, R. Ayoub, and T. Rosing. Tempomp: Integrated prediction and management of temperature in heterogeneous mpsocs. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 593–598, march 2012.

[114] K. Shin and P. Ramanathan. Real-time computing: a new discipline of computer science and engineering. *Proceedings of the IEEE*, 82(1):6 –24, jan 1994.

[115] Y. Shin and K. Choi. Power conscious fixed priority scheduling for hard real-time systems. In *DAC*, pages 134 – 139, 1999.

[116] A. Shye, B. Scholbrock, and G. Memik. Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. pages 168–178, 2009.

[117] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. *HPCA*, pages 17–28, 2002.

[118] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware computer systems: opportunities and challenges. *IEEE Micro*, 23(6):52–61, 2003.

[119] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. *ICSA*, pages 2–13, 2003.

[120] A. Sridhar, D. Atienza, Y. Temiz, Y. Leblebici, J. Thome, T. Brunschwiler, and B. Michel. Towards thermally-aware design of 3d mpsocs with inter-tier cooling. In *DATE*, pages 1–6, 2011.

[121] A. Tanenbaum. *Modern operating systems*. Prentice Hall, 2001.

[122] T.-H. Tsai and Y.-S. Chen. Thermal-aware real-time task scheduling for three dimensional multicore chip. In *ACM Symposium on Applied Computing (SAC-2012)*, pages 1618–1624, 2012.

[123] I. Ukhov, M. Bao, P. Eles, and Z. Peng. Steady-state dynamic temperature analysis and reliability optimization for embedded multiprocessor systems. In *Design Automation Conference, 2012. DAC '12.*, number 197-204, june 2012.

[124] V. S. V. Hanumaiah. Temperature-aware dvfs for hard real-time applications on multi-core processors. *IEEE Transactions on Computers*, 99, 2011.

[125] V. venkatchalam and M. Franz. Power reduction techniques for microprocessor systems. *ACM Computing Surveys*, 37(3):195 –237, september 2005.

[126] S. Wang and R. Bettati. Delay analysis in temperature-constrained hard real-time systems with general task arrivals. *RTSS*, pages 323–334, 2006.

[127] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. *ECRTS*, pages 161–170, 2006.

[128] Wikepdeia. Cubic function. *http://en.wikipedia.org/wiki/Cubic_equation*, 2008.

[129] Q. Wu, M. Pedram, and X. Wu. Clock-gating and its application to low power design of sequential circuits. volume 47.

[130] Y. Xie, G. Loh, B. Black, and K. Bernstein. Design space exploration for 3d architectures. volume 2, pages 65–103, 2006.

[131] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Mosse. Energy-efficient policies for embedded clusters. *ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*, 40(7):1–10, 2005.

[132] C. Yang, J. Chen, L. Thiele, and T. Kuo. Energy-efficient real-time task scheduling with temperature-dependent leakage. *DATE*, pages 9–14, 2010.

[133] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin. Dynamic thermal management through task scheduling. In *International Symposium on Performance Analysis of Systems and Software*, pages 191–201, 2008.

[134] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. *FOCS*, pages 374–382, 1995.

[135] L.-T. Yeh and R. C. Chu. *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices.* ASME Press, New York, NY, 2002.

[136] I. Yeo, C. C. Liu, and E. J. Kim. Predictive dynamic thermal management for multicore systems. In *DAC*, pages 734–739, 2008.

[137] L. Yuan, S. Leventhal, and G. Qu. Temperature-aware leakage minimization technique for real-time systems. *ICCAD*, pages 761–764, 2006.

[138] L. Yuan and G. Qu. Alt-dvs: Dynamic voltage scaling with awareness of leakage and temperature for real-time systems. *Adaptive Hardware and Systems, NASA/ESA Conference on*, 0:660–670, 2007.

[139] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. *ICCAD*, pages 281–288, 2007.

[140] S. Zhang and K. S. Chatha. Thermal aware task sequencing on embedded processors. *DAC*, pages 585–590, 2010.

[141] S. Zhang, K. S. Chatha, and G. Konjevod. Near optimal battery-aware energy management. In *ISLPED*, pages 249–254, 2009.

[142] Y. Zhang, M. L. Dunn, K. Gall, J. W. Elam, and S. M. George. Suppression of inelastic deformation of nanocoated thin film microstructures. *Journal of Applied Physics*, 95(12):82168225, 2004.

[143] Y. Zhang, X. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *Design Automation Conference (DAC), 2002 39th ACM/EDAC/IEEE*, pages 183–188, 2002.

[144] Y. Zhang, X. Hu, and D. Z. Chen. Thermal-aware job allocation and scheduling for three dimensional chip multiprocessor. In *International symposium on quality electronic design (ISQED)*, pages 390–398, 2010.

[145] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotleakage: a temperature-aware model of subthreshold and gate leakage for architects. *University of Virginia Dept. of Computer Science Technical Report*, 2003.

[146] X. Zhou, Y. Xu, Y. Du, Y. Zhang, and J. Yang. Thermal management for 3d processors via task scheduling. In *ICPP*, pages 115–122, 2008.

[147] C. Zhu, Z. Gu, L. Shang, R. Dick, and R. Joseph. Three-dimensional chip-multiprocessor run-time thermal management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and SystCems*, 27(8):1479–1492, 2008.

[148] D. Zhu, R. Melhem, , and B. R. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. In *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, volume 14, pages 686 – 700, July 2003.

[149] Y. Zhu and F. Mueller. Dvsleak: combining leakage reduction and voltage scaling in feedback edf scheduling. *SIGPLAN Not.*, 42(7):31–40, 2007.

[150] S. Zhuravlev, J. Saez, S. Blagodurov, A. Fedorova, and M. Prieto. Survey of energy-cognizant scheduling techniques. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1, 2012.

VITA

VIVEK CHATURVEDI

EDUCATION

Ph.D. Candidate in Electrical Engineering Florida International University, Miami, FL Advisor: Dr. Gang Quan	08/2009-Present
Master of Science, Electrical Engineering (M.S.E.E.) Syracuse University, Syracuse, NY, USA	August 2008
Bachelor of Engineering, Electronics & Communication Engineering Rajiv Gandhi Technical University, Bhopal, MP, India	June 2006

TEACHING EXPERIENCE

Teaching Assistant Syracuse University, NY	Fall 2007

INTERNSHIP EXPERIENCE

Post-Silicon Validation Intern SUN Microsystems, Burlington, MA	Summer 2007

PUBLICATIONS

Journals

J1. V. Chaturvedi, H. Huang, S. Ren, G. Quan, "On the Fundamentals of Leakage Aware Real-Time DVS Scheduling for Peak Temperature Minimization" , Journal of Systems Architecture, Vol. 58, No. 10, 387-397, 2012

J2. H. Huang, V. Chatervedi, G. Quan, J. Fan, "Throughput Maximization for Periodic Real-Time Systems under the Maximal Temperature Constraint", ACM Transactions on Embedded Computing Systems (under second review)

J3. H. Huang, V. Chaturvedi, G. Quan, "Leakage Aware Scheduling On Maximum Temperature Minimization For Periodic Hard Real-Time Systems", Journal of Low Power Electronics, Vol. 8, No. 4, 378-393, 2012

J4. G. Quan, V. Chaturvedi "Feasibility Analysis for Temperature-Constraint Hard Real-time Periodic Tasks", IEEE Transactions on Industrial Informatics, Vol. 6, No. 3, 329-339, 2010

Refereed Conferences

C1. V. Chaturvedi, H. Huang, G. Quan, "Leakage Aware Scheduling On Maximal Temperature Minimization For Periodic Hard Real-Time Systems", Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on, pp.1802-1809, June 29 2010-July 1, 2010

C2. V. Chaturvedi, G. Quan, "Leakage Conscious DVS Scheduling for Peak Temperature Minimization", IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC), Yokohama, Japan, 2011 pg. 135-140

C3. V. Chaturvedi, P. Thanarungroj, C. Liu, G. Quan, "Validation of Scheduling Techniques to Reduce Peak Temperature on an Architectural Level Platform Set-up", IEEE SoutheastCon, 2011, Nashville, TN, 111-116

C4. M. Fan, V. Chaturvedi, S. Sha, G. Quan, "Thermal-Aware Energy Minimization for Real-Time Scheduling on Multi-core Systems", IEEE Real-Time Systems Symposium (WiP-RTSS), 2012, San Juan, PR, USA

C5. M. Fan, V. Chaturvedi, S. Sha, G. Quan, "Feasibility Analysis for Temperature Constrained Real-Time Scheduling on Multi-Core Platforms", IEEE/ACM Design Automation Conference (WiP-DAC), 2013

C6. M. Fan, V. Chaturvedi, S. Sha, G. Quan, "Energy Calculation for Multi-core Systems with Leakage and Temperature Consideration", (to be submitted)

C7. M. Fan, V. Chaturvedi, S. Sha, G. Quan, "Feasibility Analysis for Temperature Constrained Real-Time Scheduling on Multi-Core Platforms", (to be submitted)