System Level Power and Thermal Management on Embedded Processors

by

Sushu Zhang

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2012 by the
Graduate Supervisory Committee:

Karam S. Chatha, Chair
Yu Cao
Goran Konjevod
Sarma Vrudhula
GuoLiang Xue

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

Semiconductor scaling technology has led to a sharp growth in transistor counts. This has resulted in an exponential increase on both power dissipation and heat flux (or power density) in modern microprocessors. These microprocessors are integrated as the major components in many modern embedded devices, which offer richer features and attain higher performance than ever before. Therefore, power and thermal management have become the significant design considerations for modern embedded devices.

Dynamic voltage/frequency scaling (DVFS) and dynamic power management (DPM) are two well-known hardware capabilities offered by modern embedded processors. However, the power or thermal aware performance optimization is not fully explored for the mainstream embedded processors with discrete DVFS and DPM capabilities. Many key problems have not been answered yet. What is the maximum performance that an embedded processor can achieve under power or thermal constraint for a periodic application? Does there exist an efficient algorithm for the power or thermal management problems with guaranteed quality bound? These questions are hard to be answered because the discrete settings of DVFS and DPM enhance the complexity of many power and thermal management problems, which are generally NP-hard. The dissertation presents a comprehensive study on these NP-hard power and thermal management problems for embedded processors with discrete DVFS and DPM capabilities.

In the domain of power management, the dissertation addresses the power minimization problem for real-time schedules, the energy-constrained make-span minimization problem on homogeneous and heterogeneous chip multiprocessors (CMP) architectures, and the battery aware energy management problem with nonlinear battery discharging model. In the domain of thermal management, the work addresses several thermal-constrained performance maximization problems for periodic embedded applications. All the addressed problems are proved to be NP-hard or strongly NP-hard in the study. Then the work focuses on the design of the off-line optimal or polynomial time approximation algorithms as solutions in the problem design space. Several addressed NP-hard problems are tackled by dynamic programming with optimal solutions and pseudo-polynomial run time complexity. Because the optimal algorithms are not efficient in worst case, the fully polynomial time approximation algorithms are provided as more efficient solutions. Some efficient heuristic algorithms are also presented as solutions to several addressed problems.

The comprehensive study answers the key questions in order to fully explore the power and thermal management potentials on embedded processors with discrete DVFS and DPM capabilities. The

provided solutions enable the theoretical analysis of the maximum performance for periodic embedded applications under power or thermal constraints.

To My Dear Family

ACKNOWLEDGEMENTS

This dissertation could not have been made possible without many support and encouragement from a lot of people.

First and foremost, I want to give my sincerest gratitude to Professor Karam S. Chatha, my Ph.D. advisor. His professional guidance, serious attitude to research, full trust of my capability and spiritual encouragement are along all the way in my Ph.D journey. His persistence to the research and supportive trust are treasures to me. I want to have many thanks to my dissertation committee members, Professor Sarma Vrudhula, Dr. Goran Konjevod, Professor GuoLiang Xue and Professor Yu Cao, for their guidance, support and feedback. I have learned a lot from my committees by taking their classes, being their teaching assistance and having discussions on the research topics.

This dissertation would never been finished without my dear family's support. I would like to give my deepest gratitude to them. My dad gives me the motivation to fulfill the work I dream on. My mom gives me the most optimistic attitude toward my every down moment. My little brother gives me the purest love that always makes me cherish everything. I would like to thank many of my relatives for the selfless support and continuous encouragement. I am backed by them whenever I need. I would like to give my special thanks to my husband, Zhizhong Tang, and my son, Stanley Tang. They give me incredible happy moments that I'd cherish for the whole life.

It is a rewarding journey to work as a research associate in the Computing Systems Research Lab at the Arizona State University. I would like to thank my fellow colleagues, Michael Baker, Weijia Che, Glenn Leary, Krishna Mehta, Krishnan Srinivasan, etc., for the friendly research environment and valuable discussions on research projects. It is a busy journey to work as a full-time employee in Microsoft and Intel during my Ph.D life. I would like to thank my industry colleagues, Ping Sager, Scott Stephens, Sanjay Sharma, Jamel Tayeb, Lakshmi Talluru, for their encouragement to finish my dissertation. It is also a happy journey full of many valuable friendships that deserves me to treasure for the whole life. I would like to thank my friends, Jianhui Chen, Huiping Cao, Min Gui, Qian Huang, Ling Liao, Liang Yang, Wei Zhang, Xin Zhang, and many names I cannot enumerate. Although we are diversified in ages, research disciplines, and occupations, we share lots of common interests and personalities. I will treasure our friendship forever.

Last but not least, I want to thank the department of Computer Science and Engineering at the Arizona State University for offering me Teaching Assistant position for 5 years, providing me incredible research environment and giving me the great facilities on future career development.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

Chapter 1

Introduction

Increased device densities and device counts fueled by semiconductor technology scaling have led to a sharp increase in power consumption and heat flux in modern microprocessors. Many modern embedded devices integrate these microprocessors to offer richer features and attain higher performance than ever before. These embedded devices are generally aimed at hand-held applications. The devices are powered by on-board batteries and do not utilize large heat sinks or fans for cooling due to size constraints. One of the important design metrics for these portable devices is the battery lifetime. Power consumption is of direct consequence to a large number of portable embedded system applications that are constrained by battery lifetime. The other important design metrics for the devices is the peak temperature limit for the die, which is the upper limit that the processors should run below. This is because the high die temperatures cause the low device reliability to failure. Further, high temperatures affect device performance as the carrier mobility decreases with increasing temperature. Finally, high temperatures result in increased leakage current which dissipates more power, and thereby potentially lead to thermal runaway.

Consequently, the increased power consumption and heat flux in microprocessors have emerged as principal barriers to the design of such hand-held and high performance embedded devices. Chip designers have incorporated architectural features for addressing the challenges of system-level power ([11, 42]) and thermal management ([12, 75]) in modern microprocessors. These features enable the designer to vary the power consumption profile of applications, and thereby limit the power and heat dissipation in embedded devices. There are two primary system level design techniques that can exploit these architectural features, dynamic voltage/frequency scaling (DVFS) and dynamic power management (DPM) ([68, 92]).

DVFS is a power conservation technique based on the CMOS property that a cubic power reduction can be achieved by a linear decrease in supply voltage with a linear slow-down in processor frequency. It adjusts applications' power or thermal profile by scaling supply voltage and frequency (v/f) of the microprocessor. DPM exploits the various sleep states available on the current day microprocessors to reduce stand-by and leakage power consumption of applications. Therefore, the key problems in system level power and thermal management for embedded applications involve the effective exploitation of DVFS and DPM schemes in order to maximize the application performance under power/thermal constraints.

1.1 Preliminaries

This section presents the preliminaries on power model, thermal model and application specific models for the design time system level power and thermal management problems.

System wide power model based on processor operating mode

In the work, we target on an embedded CMOS circuit system that consists of one/multiple processors and a set of resources that model peripheral components (for example, memory banks, flash drives, FPGA components, buses etc.). In the system, power consumption consists of different system resource portions which depends on the processor operating mode. We define the power modes that processor operates on as follows.

- Active mode, where a circuit is on and performs an operation for executing an application. In our system, the processor/processors operate in an active v/f state for application execution and other resource components such as memory and I/O interfaces need to be active (on) along with the processor. The active power is the system wide power consumption when the system is in active mode. We consider system wide power consumption for applications. The power consumption for executing an application in this work includes three portions per component of the system: dynamic power (P_{AC}), static or leakage power (P_{SC}) and an inherent power cost in keeping the system on (P_{on}). These power portions will scale with technology and architectural improvements. The total power consumption P_{active} in active mode for executing an application is:

  $$P_{active} = P_{AC} + P_{SC} + P_{on}$$

  based on the characteristics of applications. As different supply voltage and frequency is chosen for executing the application, the total power consumption in each v/f state also varies.

- Standby mode, where a circuit is on but idle to ready to execute an operation. In our system, the processor is not executing an application and enters a sleep (or low power) state, and the resources is on but idle. Therefore, the standby power is the total power when the system is in standby mode. We consider system wide power consumption when processor is in sleep state not executing application. It includes static power (P_{SC}) and inherent power cost in keeping processor on P_{on} per component. The total power consumption in standby mode is:

  $$P_{standby} = P_{SC} + P_{on}$$

where the P_{SC} dominates the standby power in many cases.

- Shutdown mode, where a circuit is off. In our system, we assume the processor and resources are turned off and thus reduce a large amount of leakage power. On the other hand, this mode brings a non-negligible energy and overhead of mode transition and requires application meets their system requirements. We omit this mode for our power management work.

In our work, we consider system wide power consumptions when processors are in active or standby mode. In the active mode, we consider the system wide power consumption for the embedded applications when processors are executed in various v/f states. In the standby mode, we consider the embedded processor enters a sleep mode which consumes a small amount of power.

Compact thermal model

The design time techniques require a thermal model of the processor to estimate the temperature. Thermal-electrical phenomenon duality permits the modeling of heat transfer behaviors by compact thermal models (CTM). CTM is an alternative to detailed thermal models which is suitable for predicting package or chip-wide temperatures during system-level thermal aware design [70]. A CTM extracts a behavioral model from an accurate but time-consuming detailed thermal model. It aims to predict the temperatures at only a few critical physical points in the design [48, 82, 93]. In the study on thermal management, we utilize CTM to model the thermal behavior of the embedded processors.

The thermal behavior of the processor in our work in Chapters 5-7 is modeled by a lumped circuit with thermal resistance and capacitance as proposed by Sabry et al. [84]. This model has been widely adopted by current research on system-level thermal aware design [7–9, 19, 22, 55, 77, 79, 98, 99]. The CTM can simulate both heat conduction and convection phenomena as well as capture steady state and transient behaviors of the temperature. In Chapter 8, we consider the effect of package temperature to the die temperature based on a lumped RC network thermal model derived from HotSpot [93]. The CTM can capture both the steady state and transient thermal behaviors of both die and package.

Application specific power and thermal aware design

Application specific power and thermal aware designs address the problem in the context of a task set. We consider the following practical applications in the advanced embedded systems.

- General real time task sets with independent tasks and release times, where the task set includes periodic, preemptive and independent tasks. The relative deadline of each task is equal to its

period. The release time of each task is not known a prior. The tasks need to be scheduled under EDF (or RM) with the deadline constraints.

- Digital signal processing, where a periodic admissible sequential schedule (PASS) can be constructed from a synchronous data flow (SDF) on uniprocessor. The PASS can be scheduled statically in compile time.

- Multimedia processing, where multimedia decoder/encoder includes well-defined loops for data processing.

The above applications can be found on embedded devices, which lacks assemblies for cooling the system. These systems expose an extreme need for power and thermal aware design with advanced semiconductor technology scaling.

In the Chapter 2, we consider the general real time task sets for minimizing the power consumption of the task set in the periodic schedules under EDF (or RM). In the Chapter 3, we consider the power management problem for a set of tasks with a common deadline on homogeneous/heterogeneous CMP architectures. In the chapters 4, 5, 6, we target the digital signal processing or multimedia processing applications. We consider a periodic task set with well defined task execution sequence. The tasks are independent and the order of execution is specified by the sequence. Periodic task set denotes that the sequence of n tasks are executed in an iterative manner. Once one run of the task set is finished, the processor continues to execute the task set for the next run. We study the power management based on a nonlinear battery discharging model and the thermal management for tasks with deterministic or uncertain execution times. In the Chapter 7, we consider a periodic task set with undefined task execution order. We study the effect of task sequencing to the temperature of the processors. In the Chapter 8, we study the thermal aware scheduling of a periodic task set with pre-defined execution order by considering the effect of package temperature to the die temperature.

In the power and thermal aware design on processors with a sleep state, we introduce a new task named sleep task. The sleep task indicates the scenario when processors enter sleep state. The sleep task can have different execution times which indicates the processor sleep time. In the chapters 4, 5, 6, 7, 8, we formulate the problems with the addressed task set and the sleep tasks.

1.2 Scope of the work

The research addresses system-level design techniques for power and thermal management on embedded processors. In recent past there has been considerable amount of research in this domain. The existing

research of DVFS and DPM schemes can be classified into four categories. The below lists the category explanations and discussions.

- Off-line ([6, 20, 28, 65, 79, 100, 107, 110, 113]) versus on-line ([12, 92, 99, 114]) techniques. This classification is based on the provided input information in problem descriptions. Off-line algorithms apply to problems that the entire sequence of inputs on the whole duration of task executions are given in advance and require to output a solution at hand, while on-line algorithms work on inputs piece-by-piece and do not require the entire input available from the start.

- Inter-task ([6,20,28,100,107,110,113]) versus intra-task DVFS ([79,92,114]). This classification is based on DVFS scheme implementation detail. The inter-task DVFS scheme only changes v/f state before or after execution of a task, while the intra-task DVFS scheme can vary v/f state at any time.

- Discrete ([6, 28, 68, 100, 113]) versus continuous DVFS ([65, 79, 92, 107, 110, 114]) schemes. This classification is based on processor model. Some existing work assumes processor is able to select any v/f values in a continuous range for task execution and the solution is a continuous DVFS scheme. Some others assume processor only has limited discrete v/f states to select and the solution is a discrete DVFS scheme.

- Heuristic ([6, 68, 92, 114]) versus optimal/approximation ([20, 28, 65, 79, 110, 113]) approaches. This classification is based on the proposed techniques for problems. Heuristic approaches are algorithms that usually produce good solutions but there is no proof that the solutions could not get arbitrarily bad. Optimal algorithms achieves optimal solutions. Approximation approaches are algorithms that find approximate solutions with provable solution quality bound and provable run time bounds. Both optimal and approximation techniques generate solutions with guaranteed quality.

In contrast to the existing approaches to system-level power and thermal management, our problem instance is characterized by the consideration of a realistic processor model that supports only discrete v/f states (as opposed to the idealistic scenario with continuous speed settings). This consideration is driven by the observations that most main stream processors only support a few discrete v/f states, and the Advanced Configuration and Power Interface (ACPI) standard also specifies only discrete v/f states [3]. Note that our discrete v/f state consideration differs from the existing approaches [79, 99] in that the available v/f states are the inputs to our problem. Further, similar to [7–9, 107] we consider that

each task operates at a single v/f state. This is due to the consideration that inter-task v/f scheduling has less overhead than intra-task techniques, and is easier (more practical) to implement [97]. Moreover, our research focuses on off-line optimal/approximation approaches for several addressed power and thermal management problems. The proposed approaches are able to generate DVFS and DPM schemes with provably solution quality bounds in provable run time bounds.

Th scope of our work is mainly on ***off-line optimal/approximation algorithms for inter-task power and thermal management based on processors with discrete DVFS and DPM states***. We have addressed related problems for periodic applications on embedded platforms.

1.3 Addressed problems and contributions

In the following paragraphs we highlight our research and contributions till date.

Power management of real-time schedules

Problem description

The work addresses the system-level low power design for a set of periodic tasks to be executed under earliest deadline first (EDF) or rate monotonic (RM) scheduling scheme on an embedded processor that only supports discrete DVFS. Each task is specified by its period and known execution times and power consumption at the various v/f states of the target processor. The objective is to assign a v/f state for execution of each task such that the total power consumption of the application is minimized subject to the processor utilization bounds of EDF and RM scheduling schemes.

Contributions

The contributions of this work are listed below.

- We prove that the specified problems are NP hard.
- We present fully polynomial time approximation schemes (FPTAS) for the problems. The FPTAS generates solutions that are guaranteed to be within a designer specified approximation bound (e.g. within 1% of the optimal power consumption) in polynomial time. To the best of our knowledge, the proposed FPTAS generates solutions with the lowest run time comparing the existing FPTAS techniques for the addressed problems.
- We present experimental results that evaluate the proposed techniques with both real and synthetic applications, and the comparisons with optimal and existing [60] approaches. Results show

that our techniques can match optimal solutions when QB is set at 1%, out perform existing approaches [100] even when QB is set at 10%, generate solutions that are quite close to optimal ($< 5\%$) even when QB is set at higher values (25%), and execute in a fraction of a second (with QB $> 5\%$) for large 100 node task sets.

Power management on homogeneous/heterogeneous CMP architectures

Problem description

The work addresses the energy constrained scheduling problem on homogeneous/heterogeneous CMP architectures that support core-level DVFS. The work jointly addresses two key problems for energy efficient application development on CMP architectures: 1) the mapping of tasks to processing elements (PE) and 2) selection of discrete v/f state for execution of each task. The objective of the techniques is to maximize the performance of an application subject to an energy budget. The problem is formulated as a makespan minimization problem for a set of independent tasks to be executed on CMP under energy budget constraints.

Contributions

The contributions of this work are listed below.

- We prove that the energy-efficient mapping and scheduling (EMMS) problem as described is strongly NP-hard.
- We propose polynomial time techniques for homogeneous and heterogeneous CMP architectures that can be shown to generate solutions whose performance (latency or makespan) is no more than twice (2-approximation) of the optimal. To the best of our knowledge the proposed techniques offer the tightest quality bounds for the EMMS problem.
- We evaluate proposed techniques with practical and synthetic applications on various CMP architectures. Results demonstrate that for practical instances of the problem the performance of our solutions is on an average no greater than 1.43 of the optimal.

Near optimal battery-aware energy management

Problem description

This work addresses the battery aware energy management problem with discrete DVFS and DPM techniques for a sequence of tasks with a common deadline. We consider the nonlinear battery discharging

model proposed in [78] and address the problem of maximizing the battery lifetime while meeting a deadline constraint.

Contributions

The contributions of this work are listed below.

- We consider the nonlinear discharging process of the battery and present an optimal algorithm based on dynamic programming. It achieves the minimum charge loss with job deadline constraint. This has not been done in any previous work.
- We also propose a fully polynomial approximation algorithm. The designer gives a specific quality bound δ $(0 < \delta < 1)$. This algorithm guarantees to achieve the minimum charge loss no more than $(1+2\delta)$ times the optimal, when the deadline is relaxed by a factor $(1+\delta+\frac{\delta}{2n+1})$ and the battery capacity is relaxed by a factor $(1+\delta)$. The complexity of the approximation algorithm is polynomial in problem size. This is the first known approximation algorithm for the battery-aware energy management problem based on a nonlinear battery discharging model.
- Our experimental results show that the approximation algorithms widely outperform an existing technique. Further, for a number of realistic and synthetic benchmarks, the qualities of the solutions produced by our approximation techniques are much better than the required quality bounds imposed by the designer.

Thermal aware scheduling for periodic applications

Problem description

The work addresses the thermal management problem for a sequence of periodic tasks executing on a processor subject to a peak temperature constraint. The execution time and power consumption at the various v/f states of the target processor are specified as part of the problem. The heat transfer characteristics of the processor is specified by a network of the thermal resistors and capacitors. The problem is specified as a latency minimization problem for the sequence of periodic tasks subject to a peak temperature constraint.

Contributions

The contributions of this work are listed below.

- We prove that the problem as described is NP-hard.

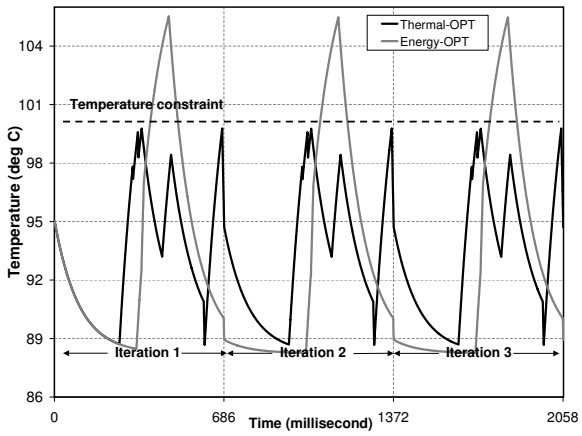


Figure 1.1: Thermal constraint violation due to optimal energy schedule

- We present a pseudo-polynomial time optimal algorithm based on dynamic programming.
- We propose a fully polynomial time approximation schemes (FPTAS) for the problem. The FPTAS can be utilized to generate solutions that are guaranteed to be within a designer specified approximation bound (for example within 1% of the optimal power consumption) in polynomial time. To the best of our knowledge, this is the first work that present both optimal and approximation algorithm for thermal aware scheduling problem.
- We evaluate our techniques by experimentation with multimedia and synthetic benchmarks mapped on a 70nm CMOS technology processor [79]. We demonstrated that energy optimal schedule can result in peak temperature violation, thereby justifying our approach for seeking a thermal aware schedule (see Figure 1.1). Further, our approach is able to match optimal solutions when QB is set at 5%, can generate solutions that are quite close to optimal ($< 5\%$) even when QB is set at higher values (50%), and executes in few seconds (with QB $> 25\%$) for large task sets with 120 nodes (while the optimal solution takes several hundred seconds). We also analyzed the effect of different thermal parameters (such as the initial temperature, the final temperature and the thermal resistance) on the performance of the schedule.

Thermal aware scheduling for applications with uncertain execution times

Problem description

The work addresses the stochastic version of thermal management problem on an embedded processor. The tasks in the sequence are specified with uncertain run times instead of deterministic run time on a v/f

state. The problem is formulated as an expected latency minimization problem for a sequence of tasks executing on an embedded processor subject to statistical thermal constraints. The statistical information of required processor cycle numbers for each task are specified as part of the problem. The statistical thermal constraint is described as the probability that the peak temperature constraint T_m will not be violated during the execution of tasks is not less than a specified value β ($\frac{1}{2} < \beta \leq 1$). The outcome is an off-line v/f schedule for the tasks that satisfy all the design requirements.

Contributions

The contributions of this work are listed below.

- We prove that the system-level thermal design problem for applications with uncertain cycle time is at least NP-hard.
- We present an optimal algorithm SO' for the case $\beta = 1$. $\beta = 1$ specifies that peak temperature limit is never violated for the task sequence even though they demonstrate variable cycle demands.
- We propose an exact optimal algorithm SO for the problem when $\frac{1}{2} < \beta \leq 1$. The computational complexity of SO is exponential in the number of tasks in the application.
- We propose a fully polynomial approximation algorithm called SA for the problem. In the case of SA the designer specifies two quality bounds namely ε (to denote that the expected latency should be within $(1+\varepsilon)$ of the optimal) and peak temperature relaxation bound μ ($0 < \mu < 1$) (to denote that peak temperature constraint can be relaxed to $(1+\mu)T_m$). The SA can generate solutions in polynomial time that are guaranteed to be within $(1+\varepsilon)$ of the optimal when peak temperature T_m is relaxed to $(1+\mu)T_m$[1]. To the best of our knowledge, this is the first work that addresses the stochastic version of the system-level thermal-aware design problem.
- We demonstrate experimental results to show that existing approaches to system-level thermal aware design cause peak temperature violations when the clock cycle demand of the tasks is variable. We also evaluated the effectiveness of our techniques with realistic and synthetic benchmarks.

[1] When the ambient temperature is T_{amb}, the peak temperature limit T_m is actually relaxed to $(T_m - T_{amb})\mu + T_m$ and is clearly less than $(1+\mu)T_m$. For example, when T_m is set as $100°C$, T_{amb} is $35°C$ and μ is 0.02, peak temperature is relaxed to $101.5°C$.

Problem description

The work addresses the thermal aware design problem to maximize the throughput for a set of periodic tasks subject to a peak temperature T_m constraint. In particular we study the thermal aware task sequencing (or ordering) problem on embedded processors with or without DVFS capabilities. The problem (denoted as $\mathscr{T}_{\mathscr{S}}$) is motivated by two primary observations (i) task execution order or sequence has a significant impact on thermal profile and consequently the performance of an application, and (ii) arbitrarily long periodic execution of the task set requires the determination of an initial temperature setting T_o^* that enables feasible (T_m is not violated) schedules in all iterations. T_o^* which needs to be determined as part of the problem solution is the optimal initial temperature (at the start of each iteration) of the sequence in steady state that results in highest throughput.

We address the thermal aware task sequencing problem and several subproblems in the work. The addressed problems are shown in Figure 1.2.

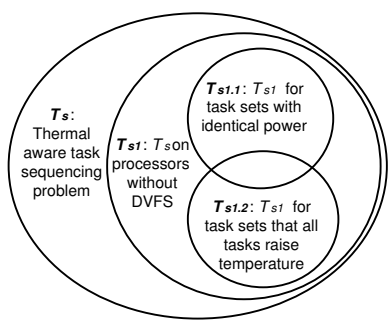


Figure 1.2: The vision of the addressed problems

In Figure 1.2, $\mathscr{T}_{\mathscr{S}}$ is the general instance of the thermal aware task sequencing problem. $\mathscr{T}_{\mathscr{S}1}$ is $\mathscr{T}_{\mathscr{S}}$ on processors without DVFS capability, which is a subproblem of $\mathscr{T}_{\mathscr{S}}$. $\mathscr{T}_{\mathscr{S}1}$ includes subproblems $\mathscr{T}_{\mathscr{S}1.1}$ and $\mathscr{T}_{\mathscr{S}1.2}$. $\mathscr{T}_{\mathscr{S}1.1}$ is $\mathscr{T}_{\mathscr{S}1}$ for task sets with identical power and $\mathscr{T}_{\mathscr{S}1.2}$ is $\mathscr{T}_{\mathscr{S}1}$ for task sets that all tasks raise temperatures. $\mathscr{T}_{\mathscr{S}1.1}$ and $\mathscr{T}_{\mathscr{S}1.2}$ intersect on $\mathscr{T}_{\mathscr{S}1}$ for task sets that all tasks raise temperatures by identical power.

Contributions

The contributions of this work are listed below.

- We derive the optimal initial temperature setting T_o^* (Section 7.3) that can lead to optimum solutions and feasible executions over multiple iterations. To the best our knowledge, this is the first

work that finds such an optimal initial temperature setting for the addressed problem.

- For all problems in Figure 1.2, our specific contributions are listed in Table 1.1. We present an

Table 1.1: Our solutions and contributions

Prob.	Solution	Contributions
$\mathscr{T}_{\mathscr{S}1.1}$	SEQ_f	Optimal algorithm in polynomial time
$\mathscr{T}_{\mathscr{S}1.2}$	SEQ_f	Optimal algorithm in polynomial time
$\mathscr{T}_{\mathscr{S}1}$	SEQ_s	Heuristic algorithm in polynomial time Average 27.0% improve against JM_s [40]
$\mathscr{T}_{\mathscr{S}}$	SEQ_d	Heuristic algorithm in polynomial time Average 9.5% improve against JM_d [40]

optimal algorithm SEQ_f for $\mathscr{T}_{\mathscr{S}1.1}$ and $\mathscr{T}_{\mathscr{S}1.2}$. We derive a sequencing property for task sets executing on processors without DVFS capability and utilize it to develop a novel algorithm SEQ_s for $\mathscr{T}_{\mathscr{S}1}$. Finally, we derive a DVFS property for the general instance of $\mathscr{T}_{\mathscr{S}}$ and present a novel algorithm SEQ_d for the same.

Thermal aware scheduling by considering the impact of package temperature

Problem description

The work addresses system-level thermal aware design problem as the performance optimization of a task set executing on an embedded processor subject to a peak temperature limit. In particular, we consider a temperature-dependent leakage power model with discrete voltage/frequency settings and a sophisticated thermal model derived from HotSpot for an embedded processor with die and package. The heat transfer characteristics of the processor is specified by a compact thermal model (CTM) which captures the inter-dependence of the die temperature with leakage power consumption and package temperature. The execution time and dynamic power consumption at the various v/f states of the target processor for each task are specified as part of the problem.

Contributions

The contributions of this work are listed below.

- We prove that the system-level thermal aware design problem as described is NP-hard.
- We present a pseudo-polynomial time optimal algorithm as solution. We also present a polynomial time algorithm based on a fully polynomial approximation scheme (FPTAS) as a more efficient solution. The solution techniques are based on the solutions to a subproblem with power budget constraint.

- We explore the optimal substructure of the subproblem and present a dynamic programming based optimal algorithm for the addressed subproblem. This algorithm is proved to be optimal and the solution time is pseudo-polynomial.

- We present a bi-criteria FPTAS for the subproblem. The bi-criteria FPTAS can generate solutions that are guaranteed to be within a designer specified approximation bound with relaxation of the power budget constraint (for example within 1% of the optimal latency with relaxation of 2% of the power budget) in polynomial time. We prove the approximation bound and fully polynomial time computational complexity.

To the best of our knowledge, this work is the first one to present both optimal and FPTAS based algorithms for the thermal aware design problem on processors with discrete v/f states based on a CTM that captures both the impact of temperature dependent leakage power and package temperature on the die temperature. We present experimental results that evaluate the proposed techniques for the thermal-aware scheduling problems with both real and synthetic applications. We show with a counter example that ignoring the impact of package temperature on die temperature cannot guarantee thermal constraints, thereby substantiating our contribution. We evaluate the actual quality of the results produced by our FPTAS based techniques for different quality bounds by comparisons with the optimal approach. The proposed FPTAS generates solutions quite close to the optimal even when the quality bound is set to a big value (say 50%) in a few seconds for a large task set with up to 50 tasks. In particular the FPTAS solutions are within 3% of the optimal even when the quality bound is set at 50%.

1.4 Outline

The rest of the dissertation is outlined as follows.

Chapter 2 addresses the power management problem of real-time schedules on embedded processors and presents optimal and approximation algorithms as solutions.

Chapter 3 addresses the power management problem on CMP architectures and presents approximation algorithms for the problem on both homogeneous and heterogeneous CMPs.

Chapter 4 addresses the battery aware energy management problem based on a nonlinear battery discharging model and provides optimal and approximation algorithms as solutions.

Chapter 5 defines the thermal management problem for a sequence of tasks and proposes both optimal and approximation algorithms as solutions.

Chapter 6 introduces uncertainty of task execution times to the thermal management problem for a sequence of tasks, and then present optimal algorithm and a fully polynomial approximation as solutions.

Chapter 7 addresses the task sequencing and scheduling problem under a peak temperature constraint for processors with or without DVFS capabilities. We provide optimal and efficient heuristic algorithms for several subproblems and problem as solutions.

Chapter 8 demonstrates the effect of package temperature to die temperature and addresses the thermal management problem for a periodic task sequence based on a sophisticate thermal model including die and package. We provide the optimal and an FPTAS based technique as solutions.

Chapter 9 concludes the work and presents future potential research directions.

Chapter 2

Power management of real-time schedules

The chapter presents the work on power management of real-time schedules on embedded processors. This problem is described as a power consumption minimization problem for a set of periodic tasks executing on uni-processor with real-time schedules. The work is organized as follows: Section 2.1 defines the problem, Section 2.2 discusses the previous work, Section 2.3 presents the fully polynomial time approximation scheme for the problem, Section 2.4 discusses the experimental results, and finally Section 2.5 concludes the work.

2.1 Problem definition

Problem description

The power management of real-time schedules on uni-processor is described as follows:

Given

- n independent periodic tasks $\mathscr{X}\{x_1,\ldots,x_n\}$ specified in the ascending order of their period d_i,
- a target embedded processor architecture with multiple active voltage and corresponding frequency states[1] $\mathscr{V}\{v_1,\ldots,v_m\}$,
- for each task $x_i \in \mathscr{X}$ and each active state $v_j \in \mathscr{V}$, p_{ij} and t_{ij} that denote the power consumption and execution time of the task, respectively,

The objective is to minimize the power consumption when the tasks are executed by Earliest deadline first (EDF) or rate monotonic (RM) scheduling schemes subject to:

- each task is executed at a unique voltage state of the processor,
- every task is finished before its next request, and
- the utilization bound of valid EDF (or RM) scheduling is satisfied.

EDF and RM are two main real time scheduling algorithms for periodic task sets on uni-processor architectures [52]. EDF is a dynamic priority scheme that assigns the highest priority to the task with the earliest deadline. RM is a static priority scheme that assigns the highest priority to the task

[1] We assume without loss of generality that at a particular voltage the processor operates at a unique frequency. The proposed techniques can also address the more general case of multiple operating frequencies at a particular processor voltage.

with shortest period. Consequently, for large independent task sets EDF has a higher utilization bound of 1 as opposed to RM whose utilization bound is given by 0.69 ($\ln 2$) [52]. If the utilization of system when applying power management is no more than the utilization bound for EDF (or RM), the task set is feasible to schedule and all the deadlines of task instances can be met by EDF (or RM). As described in the problem, each task has an associated period d_i, which represents the minimum inter release time of consecutive instances of the task. And it must finish its execution by the end of its period. In other words the relative deadline of each task is equal to its period. The release time of each task is not known a prior. To satisfy the deadline constraint for each task, the sufficient condition for EDF (or RM) is adopted as a constraint of the problem when applying power management. Examples of the systems include video-on-demand systems and digital signal processing [60].

Then, we formulate the problem as specified above and prove that the problem is NP hard. In the following discussion we address the problem in the context of EDF scheduling and then present modifications for RM scheduling. The switching overhead between frequency states or tasks is assumed to be negligible. We denote the two problems as LP-EDF and LP-RM.

Problem formulation

LP-EDF and LP-RM problems can be proved to be NP-hard. We consider scheduling the tasks over the hyper-period D which is the least common multiple of $d_1, d_2, \ldots, d_n$. For a task x_i there are $\frac{D}{d_i}$ instances to be executed in D. Let e_{ij} be the total energy consumption of the task instances when it is executed at voltage v_j, thus $e_{ij} = p_{ij} \times t_{ij} \times \frac{D}{d_i}$. Let a_{ij} denote a $0/1$ variable that is '1' if task x_i is assigned to execute at voltage/frequency state v_j (otherwise $a_{ij} = 0$). The power consumption of the set of tasks is given by:

$$P = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} \cdot e_{ij}}{D}$$

The numerator represents the total energy consumption (E) due to the execution of the tasks at their assigned voltage/frequency states in the hyper-period D. The voltage/frequency assignment problem for low power EDF (LP-EDF) or RM (LP-RM) schedules can be stated as:

$$\min E = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} \cdot e_{ij}$$

$$s.t. \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} \frac{t_{ij}}{d_i} \leq U_{EDF/RM} \tag{2.1a}$$

$$\sum_{j=1}^{m} a_{ij} = 1, \forall i \in \{1, 2, ..., n\} \tag{2.1b}$$

In the formulation Equation 2.1a denotes that the sufficient condition on the utilization bound of the EDF or RM schedule is satisfied.

Theorem 2.1.1. *The LP-EDF and LP-RM problems as stated above are NP-hard.*

Proof. The problems can be shown to be NP-hard by a reduction from the multiple-choice knapsack problem (MCKP) [43] [60]. The energy minimization objective is replaced by the goal of maximizing energy savings. Let $E_{\max}$ be the maximum energy consumption of any task, that is $E_{\max} = \max(e_{ij}), \forall x_i \in \mathscr{X}, v_j \in \mathscr{V}$. The energy savings due to a task x_i operating at voltage/frequency state v_j is given by the difference between the $E_{\max}$ and e_{ij}. Finding an optimal solution to the ILP formulation with the energy savings maximization objective is equivalent to solving MCKP, which is NP-hard [43]. □

There are known FPTAS for solving the MCKP problem. Chandra et al. [16] present the first FPTAS for MCKP problem. Lawler et al. [49] and Kellerer et al. [43] proposed similar FPTAS with running time much better than that of the scheme in [16]. However, as is often the case, equivalence of finding optimal solutions to these two problems does not imply that an approximation algorithm for MCKP can be directly used with the same approximation guarantee for LP-EDF and LP-RM. In fact, one can easily prove that FPTAS solutions to MCKP in some cases translate into poor solutions to LP-EDF and LP-RM. Thus even though there exist polynomial-time approximation schemes for MCKP, finding good approximation algorithms for LP-EDF and LP-RM are open problems. We first give an exact pseudo-polynomial time algorithm for LP-EDF and LP-RM based on dynamic programming, and then use it to construct a FPTAS.

2.2 Related work

Jha et al. [42] and Benini et al. [11] give a survey of the existing DVFS and DPM techniques, respectively. There exists a significant body of research on efficient algorithms for DVFS in hard real-time systems. These can be classified on the basis of the following categories: i) offline [6, 20, 37, 39, 41, 60, 62, 71, 86, 87, 100, 104, 106, 107, 113] versus online [44, 46, 63, 114, 115] schemes for voltage/frequency state assignment, ii) inter-task DVFS [6, 20, 37, 41, 44, 46, 60, 62, 63, 71, 86, 100, 104, 106, 107, 113, 115] versus intra-task [39, 87, 114] approaches, and iii) continuous voltage/frequency scaling [6, 37, 44, 86, 107] versus discrete active states [20, 39, 41, 46, 60, 62, 63, 71, 87, 100, 104, 113–115].

Yao et al. [107] proposed an optimal offline low power scheduling algorithm that assumed a processor with continuous voltage/ frequency scaling. However, realistic embedded processors only offer discrete voltage/frequency states. Ishihara et al. [39] proposed an optimal low power offline scheduling

algorithm where every task is executed with at most two discrete voltage states. Although intra-task DVFS approach can possibly result in greater power consumption savings than inter-task DVFS technique, it is not easy to implement in the operating system and requires a higher overhead. These drawbacks have also been specifically recognized by others [97], and are substantiated by the much larger body of work on inter-task optimizations versus intra-task approaches. Due to the above mentioned observations in the following discussions we primarily focus on online and offline low power scheduling algorithms for inter-task DVFS with discrete active states.

Pillai et al. [71] and Jejurikar et al. presented [41] offline and online heuristic schemes for integration of DVFS with real time schedulers. The technique presented by Jejurikar et al. also considered the leakage power consumption of the peripheral components that are present in the system. Yan et al. [104] in addition to the traditional DVFS also considered adaptive body biasing (ABB) to minimize the leakage power consumption of an embedded processor. Mochocki et al. presented heuristic algorithms for offline [62] and online [63] DVFS that considered switching overheads between voltage states. Xie et al. [100] presented an exponential time exact algorithm based on branch and bound, and a linear time heuristic for DVFS that considered switching overheads and power consumption of peripheral components. Mejia-Alvarez et al. [60] and Yang et al. [106] proposed a greedy heuristics based on modelling the low power scheduling problem as a variations of the standard knapsack problem. All the above mentioned approaches either propose heuristic algorithms or exponential run time exact approaches for DVFS. In contrast we propose polynomial time algorithms for offline DVFS that generate solutions for EDF and RM schedulers which are guaranteed to be within a designer specified bound from the optimal. Chen et al. [20] and Zhong et al. [113] presented fully polynomial time and pseudo polynomial time approximation algorithms for the low power DVFS problem, respectively. Our approach is a fully polynomial time approximation scheme which has a lower complexity than either of these two approaches.

2.3 Algorithms

We first present the exact algorithm for the problems, then derive fully polynomial time approximation schemes (FPTAS) as sou. Given an NP-hard minimization problem Π with an objective function f_{Π}, an algorithm $\mathscr{A}$ is an approximation scheme for Π if given an instance I of the problem, and an error parameter ε it outputs a solution s such that $f_{\Pi}(I,s) \leq (1+\varepsilon) \cdot OPT$ where OPT is the optimal solution. $\mathscr{A}$ is a FPTAS if its running time is bounded by a polynomial in the size of the instance I and $1/\varepsilon$. FPTAS is the best one can hope for a problem that is NP-hard [96].

Optimal algorithms

The exact algorithms are based on a dynamic programming algorithm for the knapsack problem [96] that runs in pseudo-polynomial time. Given $E_{\max}$ as defined above, $nE_{\max}$ is an upper bound on the energy consumption of any solution. Let $S_{i,E}$ denote an assignment of the i tasks $x_1,\ldots,x_i$ to voltages such that their energy consumption is at most E and the total processor utilization due to these i tasks is minimized. Let $U(i,E)$ be this minimum processor utilization. If $S_{i,E}$ does not exist, define $U(i,E)=\infty$. $U(1,E)$ is known for $E\in[1,\ldots,nE_{\max}]$. The recurrence relation for the dynamic programming algorithm is given by:

$$U(i,E)=\min_{j\in[1,m]}(U(i-1,E-e_{ij})+\frac{t_{ij}}{d_i}).$$

From this recurrence we can find $U(n,E)$ for all $E\in[1,nE_{\max}]$. The optimum solution is then S_{n,E^*}, where

$$E^*=\{\min E|U(n,E)\leq U_{EDF/RM}\}.$$

The recurrence leads to an algorithm that loops over tasks $i\leq n$, energy values $e\leq nE_{\max}$ and m voltage states to construct a two-dimensional table indexed by tasks and energy values, so that entry (i,e) contains $U(i,e)$. The table is constructed in order, so that before considering (i,e), the first $i-1$ rows are filled in. For each (i,e), we compute $U(i,E)$ by looping over different voltages, as indicated in the recurrence above.

The computational complexity of the algorithm is $O(n^2mE_{\max})$, because there are $n(nE_{\max})$ entries in the table, and determining each requires m steps.

$(1+\varepsilon)$-*FPTAS*

The exact algorithms described above are not polynomial because their running time includes a factor $E_{\max}$, which could be exponential in the number of tasks and voltage states. We next describe algorithms parameterized by δ. The approximation guarantee for these algorithms is $(1+2\delta)$. However, these are still FPTAS as we can get a $(1+\varepsilon)$ approximation by invoking the algorithms with parameter $\delta=\varepsilon/2$. To get polynomial algorithms with approximation guarantee $(1+2\delta)$, we first show that if the optimal energy consumption E^* was given as part of the input, we could adapt the knapsack polynomial-time approximation scheme [96] and get a $(1+2\delta)$-approximate solutions to our problems. In fact, we show the following:

Lemma 2.3.1. *Let $E_{ub} \leq \alpha E^*$ for some $\alpha \geq 1$. If probe(E_{ub}) succeeds (where probe is the function defined in lower half of Figure 2.1), then the solution found by the call to the dynamic programming procedure consumes at most $(1+\alpha\delta)E^*$ energy.*

Proof. Given δ and the upper bound E_{ub}, *probe* first scales the energy consumption values: let $K = \delta E_{ub}/n$ and replace e_{ij} by $e'_{ij} = \lceil e_{ij}/K \rceil$ for every i and j. The next step is to invoke the exact dynamic programming algorithm described in the previous section using the modified energy consumption values e'_{ij}. $U'(n,E')$ is identical to $U(n,E)$ except that it operates on scaled values. The program returns an optimal solution A to the scaled instance of the problem. Let A^* denote the optimal solution to the original instance. Let $E'(A)$ denote the cost of A in terms of the modified energy consumption values e'_{ij}. To simplify the notation, we use $ij \in A$ to denote that in the solution A, task x_i is assigned voltage v_j, thus contributing e_{ij} to the energy consumption (or e'_{ij} in the scaled instance). We have

$$\begin{aligned} E(A) \leq KE'(A) &= \sum_{ij \in A} Ke'_{ij} \leq \sum_{ij \in A^*} Ke'_{ij} \\ &\leq \sum_{ij \in A^*} (e_{ij} + K) \leq E^* + nK = E^* + \delta E_{ub} \\ &\leq E^*(1+\alpha\delta). \end{aligned}$$

The first step is true because of rounding after scaling. The second step expands the energy consumption of A term by term. The third step is true because A is the optimal solution for the scaled instance, and so in terms of e' it is cheaper than A^*. The fourth step follows because by the definition of e'_{ij}, we have $e_{ij} \leq Ke'_{ij} \leq e_{ij} + K$. The remaining steps follow from the definitions of K and α. □

We use the function *probe* as a building block for our algorithms. The full algorithms are shown in Figure 2.1. It consists of a binary search on the sequence $(1, 2, 2^2, \ldots, 2^i, \ldots, 2^N)$, where $N = \lceil \lg nE_{\max} \rceil$.

Lemma 2.3.2. *If probe(E) returns failure, then $E^* > E$.*

Proof. Suppose $E \geq E^*$. Then by the definition of scaling, the optimal solution A^* to the original instance has scaled energy consumption at most

$$E'(A^*) \leq \left\lceil \frac{E^*}{K} \right\rceil + n \leq \left\lceil \frac{E}{K} \right\rceil + n.$$

Since *probe*(E) invokes the dynamic program with the upper bound $\lceil \frac{E}{K} \rceil + n$, there exists a feasible solution and *probe*(E) will succeed. □

LP-EDF/LP-RM FPTAS(δ):

```
Initial: N = ⌈lg nE_max⌉, l = 1, r = N;
while (l < r){
    h = ⌊(l+r)/2⌋.
    if (probe(2^h) = success) then r = h;
    else l = h;
}
h = r;
return 2^h;
```

probe(E):

```
K = δE/n;
e'_ij = ⌈e_ij/K⌉, E' = ⌈E/K⌉ + n;
if (U'(n,E') ≤ U_EDF/RM)) {return success;}
else {return failure;} endif;
```

Figure 2.1: LP-EDF and LP-RM:FPTAS

Theorem 2.3.1. *The approximation ratio of LP-EDF/LP-RM FPTAS is* $(1+2\delta)$.

Proof. Let 2^h be the value returned by the binary search. Let $E(A)$ be the energy consumption of the solution found by $probe(2^h)$. We have the following two cases:

Case I: $2^h \leq E^*$. Then by Lemma 2.3.1, $E(A) \leq (1+\delta)E^*$.

Case II: $2^h > E^*$. As 2^h is the smallest value for which the probe succeeds we have $2^h < 2E^*$. Now Lemma 2.3.1 implies $E(A) \leq (1+2\delta)E^*$. □

Lemma 2.3.3. *The running time of LP-EDF/LP-RM FPTAS is bounded by* $O(\frac{n^2m}{\delta}\lg\lg(nE_{\max}))$.

Proof. The binary search is applied to the N-element sequence $(1,2,\ldots,2^i,\ldots,2^N)$, where $N = \lceil \lg(nE_{\max})\rceil$. Therefore, *probe* is invoked at most $O(lg(N)) = O(\lg\lg(nE_{\max}))$ times. Each call to *probe* requires $O(\frac{n^2m}{\delta})$ time. Thus the overall running time of the algorithm is $O(\frac{n^2m}{\delta}\lg\lg(nE_{\max}))$. While this expression contains the term $E_{\max}$, the double logarithm ensures that the running time is not only polynomial in the size of the input, but also that the extra term $\lg\lg(nE_{\max})$ is only logarithmic in the input size. □

Theorem 2.3.2. *LP-EDF/LP-RM FPTAS are fully polynomial approximation schemes for LP-EDF and LP-RM problems, respectively.*

Proof. From Theorem 2.3.1 and Lemma 2.3.3, the proof follows. □

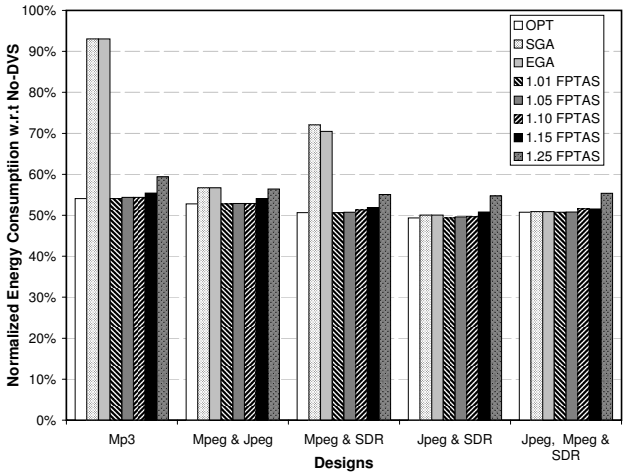


Figure 2.2: LP-EDF FPTAS: Multimedia benchmarks

2.4 Experimental results

We present and analyze the results of experimentation that was performed to evaluate our techniques. We evaluated both LP-EDF and LP-RM FPTAS for multimedia benchmark applications and synthetic task sets. In both the experiments we compared our techniques against a non-DVFS approach, optimal designs, and SGA and EGA algorithms proposed by Mejia-Alvarez et al. [60]. The non-DVFS approach executes the tasks at their highest voltage/ frequency state. The optimal designs were obtained by utilizing the exact algorithm discussed in Section 2.3. The Intel StrongARM 1100 processor was considered as the target embedded processor for the two experiments. The optimization techniques were coded in C++ and the experimentations were performed on a Pentium M/1.6GHz/512MB WindowsXP PC.

Results for multimedia benchmarks

We considered four applications drawn from the multimedia domain namely JPEG decoding, MPEG2 decoding, MP3 encoding and software defined radio (SDR). The JPEG decoding algorithm was modeled by four tasks consisting of: variable length decoding, huffman decoding, inverse-zigzag and quantization, and IDCT. An MPEG2 stream consists of I, P and B pictures. Decoding an I picture consists of the following tasks: preprocessing, variable length decoding, inverse zigzag and de-quantization, and IDCT. P and B picture decoding consist of preprocessing and motion compensation. The MP3 encoding algorithm was modeled by three tasks consisting of pulse code modulation, filtering, and huffman encoding. Finally, the SDR application was obtained from Niyogi et al. [67] and consisted of low pass

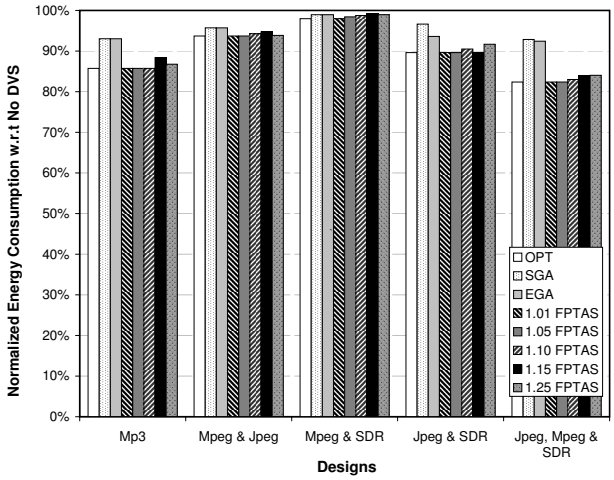


Figure 2.3: LP-RM FPTAS: Multimedia benchmarks

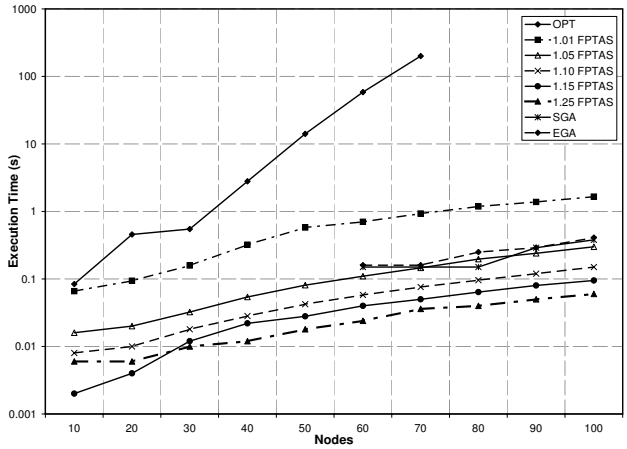


Figure 2.4: LP-EDF FPTAS: execution time

filter, demodulator and equalizer.

The Intel StrongARM 1100 processor was run at the following specifications: 1.5 V - 206 MHz, 1.4 V - 192 MHz, 1.2 V - 162 MHz and 1.1 V - 133 MHz. We obtained execution times (CPU run time) and average power consumption estimates of the tasks in the multimedia applications by utilizing the JouleTrack simulator [90] for the StrongArm processor. We considered the design of application sets with the period constraints for all tasks in a particular application specified as follows: JPEG = $1ms$, MPEG2 = $900\mu s$, MP3 = $45ms$, and SDR = $8ms$. For the integrated JPEG, MPEG2 and MP3 design the periods were specified as $1.5ms$, $1.5ms$ and $12ms$, respectively. We implemented the designs with both

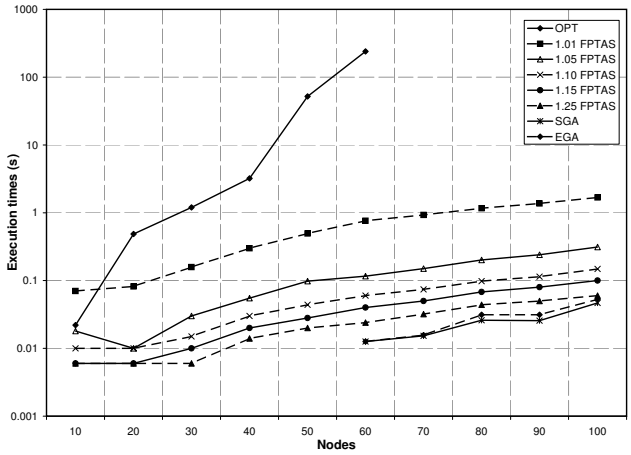


Figure 2.5: LP-RM FPTAS: execution time

LP-EDF FPTAS and LP-RM FPTAS and with quality bounds of 1% ($\varepsilon = 0.01$), 5% ($\varepsilon = 0.05$), 10% ($\varepsilon = 0.10$), 15% ($\varepsilon = 0.15$) and 25% ($\varepsilon = 0.25$). The results are plotted in Figure 2.2 and 2.3 for EDF and RM schedulers, respectively. The plots depict the normalized energy reduction due to a particular approach in comparison to no DVFS technique.

Evaluation of LP-EDF FPTAS Both SGA and EGA give inferior results in comparison to the optimal in all cases. In fact on an average the SGA and EGA are over 1.25 (max = 1.72) and 1.24 (max = 1.72) times the optimal, respectively. In contrast LP-EDF FPTAS with a bound of 1 % is able to match the optimal solution in all cases. Even with a quality bound of 25 % LP-EDF FPTAS is able to out perform SGA and EGA in 3 out of the 5 cases, and is on an average within 1.09 (max = 1.11) of the optimal.

Evaluation of LP-RM FPTAS As the RM scheduler is constrained by much lower utilization bound, the energy consumption is higher in comparison to the EDF scheduler. Similar to the EDF scheduler, both SGA and EGA give inferior results in comparison to the optimal in all cases. On an average the SGA and EGA are 1.06 (max = 1.13) and 1.06 (max = 1.12) times the optimal, respectively. Again, LP-RM FPTAS with a quality bound of 1 % is able to match the optimal solution in all cases. LP-RM FPTAS with a quality bound of 25 % out performs SGA and EGA in all cases, and is on an average within 1.01 (max = 1.02) of the optimal.

Summary We can conclude that for realistic applications both LP-EDF and LP-RM FPTAS are able to match the optimal with a quality bound of 1 %, out perform SGA and EGA in most instances with a quality bound of 25 % and also generate high quality solutions with a quality bound of 25 %.

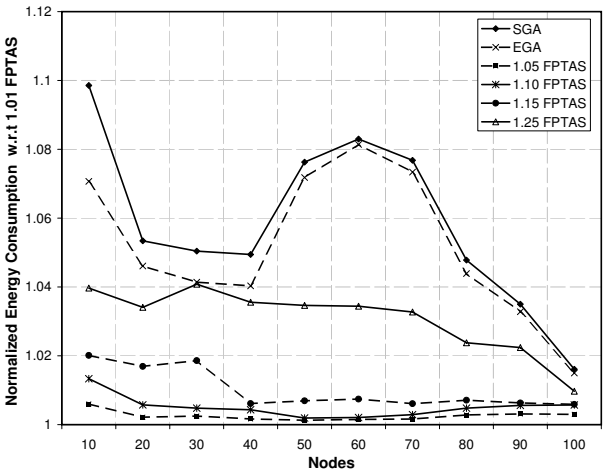


Figure 2.6: LP-EDF FPTAS: actual design quality

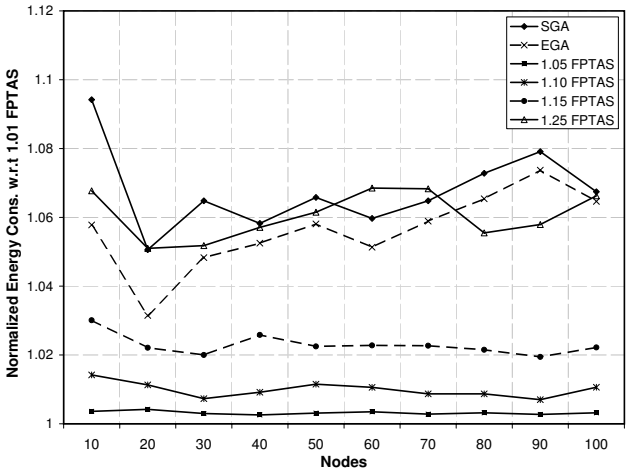


Figure 2.7: LP-RM FPTAS: actual design quality

Results for synthetic task sets

We evaluated the proposed techniques by experimenting with large synthetic task sets with up to 100 nodes. The number of tasks in each set were varied from 10 to 100 in steps of 10 with 10 task sets at each value. Each task was assumed to run on the StrongArm processor with 5 voltages (0.90V, 1.00V, 1.20V, 1.30V, 1.50V). The workload of the tasks was varied uniform randomly from 10^2 to 10^8 clock cycles. In the case of EDF scheduler, for each task set, 10% of the tasks were set to be high utilization tasks at lowest operating frequency. As the RM schedule has lower utilization than EDF scheduler, only

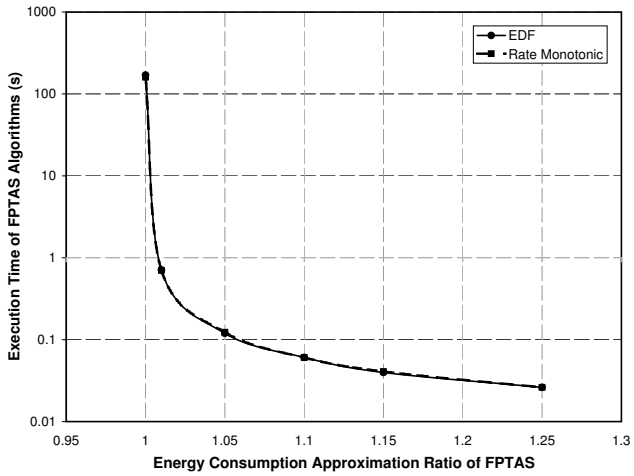


Figure 2.8: Execution time versus design quality

5% of total tasks were assigned as high utilization. The utilization of tasks with high utilization was varied uniform randomly from $\frac{1}{N}$ to $\frac{1.5}{N}$ where N is the number of jobs in the task set. The utilization of low utilization tasks was varied uniform randomly from 0.0001 to $\frac{1}{N}$. We generated designs by executing the optimal algorithm, SGA, EGA and both FPTAS-EDF and FPTAS-RM. We set the quality bounds of our algorithm at 1%, 5%, 10%, 15% and 25%. We recorded the execution times of the various techniques and also compared the actual quality of results with the solution of 1% FPTAS of the respective scheduler. Figures 2.4 and 2.5 depict the execution time of the various approaches, and Figures 2.6 and 2.7 present the comparison of design quality with 1% FPTAS solution.

Evaluation of LP-EDF FPTAS The run time and memory usage of the optimal technique increases exponentially, and we were unable to obtain the results in a reasonable amount of time for task sets greater than 70 nodes. Both SGA and EGA are quite efficient and can generate results for large task sets (100 nodes) in less than a second[2]. The execution time of LP-EDF FPTAS is comparable to SGA and EGA for a quality bound greater than 5%. The LP-EDF FPTAS with a quality bound of 1% takes just over a second for 90 and 100 node task sets. The design quality of SGA and EGA was on an average inferior to LP-EDF FPTAS with a quality bound of 25%. For quality bounds of less than or equal to 15% the LP-EDF FPTAS was much superior to both SGA and EGA techniques.

Evaluation of LP-RM FPTAS The run time of the optimal technique for LP-RM rises much fast than LP-EDF. Both SGA and EGA are very efficient, and are comparable to run time of LP-RM FPTAS with a quality bound of 25%. LP-RM FPTAS with a quality bound of 1% requires just over a second for

[2]We do not plot the execution times for SGA and EGA for less than 60 nodes as they are close to zero.

large 80 to 100 node task sets. The run times of LP-RM FPTAS for all other quality bounds was under a second for large task sets. The design quality of SGA and EGA was comparable to LP-RM FPTAS with a quality bound of 25%. The LP-RM FPTAS was much superior to both SGA and EGA techniques for quality bounds of less than or equal to 15%.

Summary Although SGA and EGA are efficient techniques, the average quality of their solutions is consistently poorer than LP-EDF and LP-RM FPTAS for quality bounds less than or equal to 15%.

Execution time versus quality bounds

Figure 2.8 plots the average execution time of the LP-EDF and LP-RM FPTAS for synthetic task sets versus the approximation ratio or quality bound[3]. We can observe that at a quality bound of 10% the execution time of the approaches is less than 0.001 times the run time of the optimal. Thus, at 10 % quality bound the two approaches offer an excellent trade-off between design quality and solution time.

2.5 Conclusion

We addressed the minimum power consumption assignment of voltage/frequency states for a set of periodic tasks to be executed on an embedded processor by EDF and RM schedulers. We showed that the problem is NP-hard and presented FPTAS as solutions. Experimental results with both multimedia applications and synthetic benchmarks demonstrate that our approaches with a quality bound of 1% are able to get very close to the optimal, and produce high quality solution even when an approximation bound of 25% is considered.

[3]The plots of the two approaches are overlapping. Consequently, the plots appear as only a single curve.

Chapter 3

Power management on homogeneous/heterogeneous CMP architectures

The chapter first addresses an power management problem on multiprocessors in terms of throughput maximization problem with energy budget constraints on CMP. Then approximated algorithms are provided as solutions. The work is organized as follows: Section 3.1 defines the problem, Section 3.2 discusses the previous work, Section 3.3 presents the approximation schemes for the problem on homogeneous and heterogeneous architectures, Section 3.4 discusses the experimental results, and finally Section 3.5 concludes the work.

3.1 Problem definition

Consider a CMP composed of m processing elements (PEs) denoted by the set $\Phi = \{pe_1, \ldots, pe_i, \ldots, pe_m\}$. Each PE consists of a DVFS equipped processor, a local memory and a globally coherent DMA engine. An interconnect bus is provided for the communication between the PEs. On each pe_i, there is an available active voltage/frequency (v/f) state set $\Psi_i = \{s_1, \ldots, s_k, \ldots, s_{l_i}\}(|\Psi_i| = l_i)$. We assume the local memory is large enough to hold all the tasks.

The energy-efficient multiprocessor mapping and scheduling (EMMS) problem is described as:

Given a target multiprocessor chip CMP, n independent non-preemptable tasks $\Gamma = \{\tau_1, \ldots, \tau_j, \ldots, \tau_n\}$ *to be executed on the CMP, the objective is to maximize the chip-level throughput such that each task is scheduled at a unique v/f state on one of the PEs, and the total energy consumption is no more than an energy budget C.*

We assume for each task τ_j, c_{ijk} and t_{ijk} are given as the energy consumption and the worst case execution time (WCET) of the task on $pe_i \in \Phi$ at v/f state $s_k \in \Psi_i$, respectively. And all the tasks arrive the CMP at time zero. The objective to maximize the chip-level throughput can be transformed to minimize the overall completion time (makespan) of the task set. In this work, we focus on off-line provable approximation techniques for the EMMS problem. The Integer Linear Programming (ILP) formulation of the EMMS problem, named $\mathbb{P}1$, is as follows:

$$\min \mathrm{T}$$

$$s.t. \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{l_i} c_{ijk} x_{ijk} \leq C \tag{3.1a}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{l_i} t_{ijk} x_{ijk} \leq T, \forall pe_i \in \Phi; \tag{3.1b}$$

$$\sum_{i=1}^{m} \sum_{k=1}^{l_i} x_{ijk} = 1, \forall \tau_j \in \Gamma; \tag{3.1c}$$

$$x_{ijk} = \{0,1\}, \forall pe_i \in \Phi, \forall \tau_j \in \Gamma, \forall s_k \in \Psi_i. \tag{3.1d}$$

Here x_{ijk} is 1 if and only if τ_j is executed at v/f state s_k of the pe_i, otherwise 0. Constraint (3.1a) specifies that the total energy consumption is no more than C. Constraint (3.1b) describes that the overall throughput is limited by the completion time of tasks on each PE. Constraint (3.1c) ensures that each task is executed on one voltage of some PE.

Theorem 3.1.1. *The EMMS problem is strongly NP-hard.*

Proof. We prove the strongly NP-hardness by showing that a well-known strongly NP-hard problem, the minimum makespan scheduling (MMS) problem [27], is a special case of the EMMS problem. When the processing time of each task is fixed and there is no energy budget constraint, the EMMS problem becomes a MMS problem with an arbitrary m. □

Hochbaum et al. [27] discusses several research results on approximation algorithms for the MMS problem. However, those results are for the classical MMS problems without consideration of energy budget or v/f states. In this work, we focus on approximation techniques for the EMMS problem. Standing on the shoulders of giants, we extend some useful ideas for the MMS problems to address the EMMS problem. In the following section, we present a 2-approximation algorithm for scheduling on homogeneous CMP by extending the LP rounding method for the MMS problem with identical machines [27]. Then, we propose a 2-approximation algorithm for scheduling on heterogeneous CMP based on the solution for the MMS problem with unrelated machines (the generalized assignment problem) [27, 88]. In contrast to the original algorithms [27, 88] for the MMS problems, both of our algorithms can deal with simultaneous v/f state assignment and task to PE mapping. In this work, the WCET of tasks is assumed to be integral as the cycle numbers in cores, and the switching overhead between v/f states is negligible.

3.2 Related work

The existing techniques for energy-efficient scheduling on CMPs can be classified into several categories based on different metrics: i) the laptop problem [14, 20, 28, 38, 72] versus the server problem [6, 51, 95] ii) continuous [14, 20, 72] versus discrete v/f states [28, 38, 51, 95] iii) heuristic [6, 38, 51, 95] versus approximation [14, 20, 28, 72] techniques.

Bunde et al. [14] classified the energy efficient scheduling problems into the laptop problem and the server problem. The former fixed the energy consumption to maximize schedule performance, while the latter fixed the schedule performance to minimize energy consumption. Jha et al. [42] introduced different variations of the both problems with more considerations such as the task models [20, 72],

the communication links [5, 95] and the synthesis costs [28]. Our work belongs to the laptop problem, which asks "given an energy budget, what is the best schedule to maximize performance". We focus on independent non-preemptable task set and assume all the tasks arrive at the same time instance.

We focus on the approximation techniques for the problem that can generate solutions with guaranteed quality bounds. The existing heuristic techniques [6, 38, 51, 95] cannot satisfy this property. Pruhs et al. [72] proposed a polynomial time approximation scheme based on load balancing for the energy-efficient scheduling problem. Bunde [14] extended the work by Pruhs et al. and gave an exact algorithm for multiprocessor makespan minimization of equal-workload jobs. Chen et al. [20] summarized their approximation techniques on several variants of the energy-efficient scheduling problem. However, all of these techniques assumed that v/f could be scaled continuously. As we know, most commercial processors only support discrete v/f states and the optimal v/f as generated by the previous techniques may not be available. In the discrete v/f domain, Andrei et al. [5] presented a MILP formulation with multiple considerations for the energy-efficient scheduling problem. Hsu et al. [28] considered an independent task set with EDF/RM schedule and provided an (m+2)-approximation algorithm to minimize the allocation cost within an energy budget. In contrast, we propose 2-approximation polynomial time techniques for the EMMS problem on homogeneous and heterogeneous CMPs. To the best of our knowledge, our techniques offer the tightest quality bounds for the EMMS problem.

3.3 Algorithms

In this section, we propose polynomial time approximation algorithms for the EMMS problem. Initially, we utilize a binary search to achieve a tight lower bound of the optimal EMMS. Then the scheduling algorithms for the homogeneous CMP and heterogeneous CMP are proposed based on a fractional schedule result. Both of the scheduling techniques are justified to be 2-approximation algorithms of the optimal EMMS.

Finding a tight lower bound of the optimal

In general, the LP relaxation of an ILP problem is an effective way to obtain the lower bound of the optimal. However, sometimes the LP relaxation result is not a tight lower bound. Consider the LP relaxation of $\mathbb{P}1$ by replacing $x_{ijk} \in \{0,1\}$ with $x_{ijk} \geq 0$, denoted as $\mathbb{P}1\mathbb{LP}$. Suppose that we have two identical PEs, one single task, and each PE is only equipped with one v/f level. Assume the WCET of this task is t on the PE. The optimal makespan of $\mathbb{P}1$, denoted as $T^\star$, would be t. However, the naive LP relaxation gives a solution where the task is split into equal halves on the two PEs. The optimal makespan of the LP relaxation is $\frac{1}{2}t$. The bound on $T^\star$ is not tight because the WCET of the single job

P1LP-OPT:

$l = T_{LB}, r = T_{UB}$
while $(l < r)$
$\{ h = \lfloor \frac{l+r}{2} \rfloor$.
  if $(probe(h) = success)$ then $r = h$;
  else $l = h$;$\}$
return $T^{\star}_{\mathbb{P}1\mathbb{LP}} = r$ and S;

$probe(T)$:

Let $\mathrm{T}_d = \mathrm{T}$, solve $\mathbb{P}2\mathbb{LP}$ by the simplex method;
if $(C_s \leq C))$ return *success* and the solution S;
else return $failure$;

Figure 3.1: An optimal algorithm for $\mathbb{P}1\mathbb{LP}$

is larger than the lower bound. To avoid this case and achieve a tighter lower bound of $\mathrm{T}^{\star}$, we include a property of the optimal solution of the ILP as an extra constraint. Thus, this constraint would not affect $\mathrm{T}^{\star}$.

$$\text{if } t_{ijk} > T, x_{ijk} = 0; \tag{3.2}$$

Since the if-then constraint is not easy to be linearized because of the unknown T, we introduce another problem, $\mathbb{P}2$, which includes this constraint. $\mathbb{P}2$ is described as "given a deadline T_d for the task set, what is the best schedule with minimum energy consumption". The ILP formulation is as follows:

$$\min C_s = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{l_i} c_{ijk} x_{ijk}$$

$s.t.$ Constraint (3.1b)(3.1c)(3.1d)(3.2) and replace T by T_d in (3.1b)(3.2).

In $\mathbb{P}2$, the if-then constraint can be transformed to a preprocessing step by setting values of some x_{ijk}, since T_d is given.

Let the $T^{\star}_{\mathbb{P}1\mathbb{LP}}$ be the optimal makespan of the $\mathbb{P}1\mathbb{LP}$ problem with Constraint (3.2). Based on the linear relaxation of $\mathbb{P}2$, named as $\mathbb{P}2\mathbb{LP}$, $T^{\star}_{\mathbb{P}1\mathbb{LP}}$ is found by the P1LP-OPT algorithm in Figure 3.1. In P1LP-OPT, the T_{LB} is set as $\min\{t_{ijk}\}$ and the T_{UB} is set as $n \cdot \max\{t_{ijk}\}$, $\forall pe_i \in \Phi, \tau_j \in \Gamma, \forall s_k \in \Psi_i$. Then, we have the following lemma. The proof is omitted here since it is similar to that for the MMS problem [88].

Lemma 3.3.1. *The binary search based on $\mathbb{P}2\mathbb{LP}$ in the P1LP-OPT algorithm finds the optimal solution $T^{\star}_{\mathbb{P}1\mathbb{LP}}$ of $\mathbb{P}1\mathbb{LP}$.*

The P1LP-OPT returns an optimal fractional schedule S. For each $x_{ijk} > 0$ in S, $t_{ijk} \leq T^{\star}_{\mathbb{P}1\mathbb{LP}}$, because of Constraint (3.2). In the following subsections, we present the scheduling techniques based on S for the homogeneous CMP and the heterogeneous CMP.

Scheduling on Homogeneous CMP

Homogeneous (or symmetric) CMP consists of m identical PEs. The scheduling problem on homogeneous CMP is easier than that on heterogeneous one, because the active v/f state space is independent of the PEs in the CMP. In other words, a task requires the same amount of WCET and consumes the same energy on a particular active state among all the PEs. Based on this property, we propose a simple 2-approximation technique in Figure 3.2.

Observe that the linear relaxation of $\mathbb{P}2$ after the preprocessing step for Constraint (3.2) includes at most $m+n$ constraints in addition to the non-negativity conditions. Therefore, each basic solution has at most $m+n$ basic variables which may take positive values while the other non-basic variables take the value zero. The simplex method searches among the basic solutions and generates an optimal solution of this form [69]. Thus, if $n > m$, there are at most m tasks that get split. Based on the property, we have

Theorem 3.3.1. *The makespan of the schedule from the P_{sym} algorithm is at most twice of the optimal.*

Proof. In Step 1, the P_{sym} algorithm computes a fractional assignment from P1LP-OPT. After Step 2, the s_k with the smallest c_{ijk} for each τ_j is selected, when the t_{ijk} is associated with a positive x_{ijk}. The energy budget constraint is satisfied. At Step 3, we consider the two cases here.

- Case $n \leq m$: It is clear the n tasks can be assigned to disjoint m PEs. Thus, the overall makespan is determined by the WCET of single task on each PE. Because the schedule S satisfies Constraint (3.2), the makespan is $\leq T^{\star}$. Because $T^{\star}$ is the optimal, the result from P_{sym} is the optimal.

- Case $n > m$: After Part (a) in Step 3 of the P_{sym}, because of constraint (3.1b), the maximum completion time of this part is no more than $T^{\star}_{\mathbb{P}1\mathbb{LP}}$. Thus, it is no more than $T^{\star}$. Starting from this time point on each PE, the task-PE mapping with the fractional x_{ijk} is determined by Part (b). Since at most m tasks get split in S, the task number in Part (b) is no more than m. Thus, similar to the case $n \leq m$, the maximum completion time of Part (b) is no more than $T^{\star}$. Therefore, the overall makespan is no more than twice $T^{\star}$.

Thus, it is proved. □

P_{sym}:

Step 1: Achieve the fractional schedule S from P1LP-OPT;
Step 2: For each τ_j, select the s_k with the associated smallest c_{ijk} with positive x_{ijk} in S;
Step 3: If $n \leq m$, schedule the tasks to disjoint PEs; else:
(a) schedule the tasks with the integral x_{ijk} from S;
(b) schedule the remaining tasks with the determined s_k to arbitrary disjoint PEs.

Figure 3.2: A 2-approximation for EMMS problem with homogeneous CMP

P_{asym}:

Step 1: Achieve S from P1LP-OPT;
Step 2: Construct a bipartite graph $G = (U, V, E)$;
Step 3: Find a minimum cost matching A that exactly matches all the task nodes in G;
Step 4: For each edge in A, assign the task to the corresponding PE and the active voltage state via the associated x_{ijk}.

Figure 3.3: A 2-approximation for EMMS problem with heterogeneous CMP

Scheduling on Heterogeneous CMP

Another kind of practical CMP architecture consists of a diversity of PEs, called heterogeneous (or asymmetric) CMP. Heterogeneous PEs imply different active v/f states with varying power/WCET characteristics. In this subsection, we propose a 2-approximation scheduling algorithm based on the algorithm for the generalized assignment problem (GAP) [88]. We construct a bipartite graph based on the schedule S generated from P1LP-OPT, achieve a minimum cost matching on the graph and then schedule the tasks according to the matching. The algorithm P_{asym} is described in Figure 3.3. Our technique differs from [88] in that our algorithm addresses the scheduling problem with one more dimension namely the active v/f state assignment.

In Step 2 of P_{asym} we construct a bipartite graph G with two disjoint node sets (U,V) and one edge set (E). One side of the graph, U, consists of all the task nodes $U = \{u_j | \tau_j \in \Gamma\}$. The other side of the graph, V, consists of all the PE nodes with $\sum_{j=1}^{n} \sum_{k=1}^{l_i} x_{ijk} > 0$. For each pe_i in V, there are $q_i = \lceil \sum_{j=1}^{n} \sum_{k=1}^{l_i} x_{ijk} \rceil$ nodes in V ($q_i \neq 0$).

Edges of the G are constructed based on the positive $\{x_{ijk}\}$ from the fractional schedule S. In this section, we only consider the items associated with $x_{ijk} > 0$ in S. Let v_{ih} denote the $h^{th} (h = 1, 2, ...q_i)$

node associated with pe_i in V. Let $e = (u_j, v_{ih})$ be an undirected edge connecting node u_j and v_{ih}. For each $pe_i \in M$, construct a list including all the t_{ijk} with positive x_{ijk}, $\forall \tau_j \in \Gamma, s_k \in \Psi_i$. Sort this list in the non-increasing order of t_{ijk}, and name the sorted list as $L_i(t) = \{t_{ijk}\}$. Construct an associated list $L_i(x) = \{x_{ijk}\}$ according to the order of $L_i(t)$. Recall that there are q_i nodes for the pe_i in V. If $q_i = 1$, construct an edge $e = (u_j, v_{i1})$ for every $x_{ijk} > 0$ and assign $x'(e) = x_{ijk}, t(e) = t_{ijk}, c(e) = c_{ijk}$. If $q_i > 1$, for v_{i1} $(h = 1)$ find the smallest splitting index r in $L_i(x)$ such that $\sum_1^r x_{ijk} \geq 1$. Construct $r-1$ edges $e = (u_j, v_{i1})$ for the first $r-1$ x_{ijk} in $L_i(x)$. Assign $x'(e) = x_{ijk}$ as the case of $q_i = 1$. Add an edge for the r_{th} x_{ijk} as $e = (u_j, v_{i1})$ and assign $x'(e) = 1 - \sum_1^{r-1} x_{ijk}$. Delete the first $r-1$ x_{ijk} from $L_i(x)$ and replace the r_{th} x_{ijk} as $\sum_1^r x_{ijk} - 1$. The assignment rules of $t(e)$ and $c(e)$ are always the same as those of $q_i = 1$ case. Similarly, for each $h = 2, 3, ..., q_i$, construct the edges and $x'(e)$ such that the following properties hold true:

i. $\forall v_{ih} \in V, \sum_{e \in E_{ih}} x'(e) = 1$, where E_{ih} denotes all the edges e incident to v_{ih}, $\forall h = 1, 2, ..., q_i - 1$.

ii. $\forall i \in M, \sum_{j=1}^{n} \sum_{k=1}^{l_i} x_{ijk} = \sum_{e \in E_i} x'(e)$, where E_i includes all the edges incident to any node of pe_i in V.

iii. $\forall i \in M, \min(t_{e \in E_{ih}}(e)) \geq \max(t_{e \in E_{ih+1}}(e))$.

Properties i. and ii. follow from the computation of $x'(e)$. Property iii. follows from edge construction based on $L_i(x)$ in non-increasing order of t_{ijk}.

We present an example to show the construction. Suppose that $m = 2, n = 3$. pe_1 only has one voltage, and pe_2 has two voltage states. After the LP relaxation of $\mathbb{P}1$ problem, the schedule S is a 3×3 matrix as follows. The values inside the square embraces [ ] are the related t_{ijk}.

$$\begin{pmatrix} x_{111} = \frac{1}{3}[6] & x_{211} = 0 & x_{212} = \frac{2}{3}[6] \\ x_{121} = 0 & x_{221} = \frac{1}{2}[5] & x_{222} = \frac{1}{2}[4] \\ x_{131} = 0 & x_{231} = 1[1] & x_{232} = 0 \end{pmatrix}$$

The constructed bipartite graph is shown in Figure 3.4. On pe_1, because $h_1 = \lceil x_{111} + x_{121} + x_{131} \rceil = 1$, $x'(u_1, v_{11}) = x_{111}$. On pe_2, because $\sum_{j=1}^{n} \sum_{k=1}^{l_i} x_{2jk} = 2\frac{2}{3}$, $h_2 = 3$. According to the non-increasing order of t_{ijk}, $L_2(t) = \{6, 5, 4, 1\}$. Therefore, $L_2(x) = \{x_{212}, x_{221}, x_{222}, x_{231}\}$. The first splitting item is x_{221} in $L_2(x)$ as $x_{212} + x_{221} > 1$. We add edge $(u1, v21)$ with $x'(e) = x_{212}$. Then, for x_{221}, we add an edge $(u2, v21)$ and assign $x'(e) = 1 - x_{212} = \frac{1}{3}$. We delete x_{212} from $L_2(x)$ and replace x_{221} as $x_{212} + x_{221} - 1 = \frac{1}{6}$. Thus, $L_2(x) = \{\frac{1}{6}, x_{222}, x_{231}\}$. For $v22$, because $\frac{1}{6} + x_{222} + x_{231} > 1$, we add edges $(u2, v22)$ and $(u2, v22)$ with $x'(e) = \frac{1}{6}$ and $x'(e) = x_{222} = \frac{1}{2}$ respectively. We construct one portion of

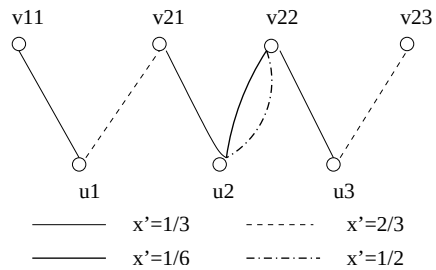


Figure 3.4: The constructed bipartite graph $G(U,V,E)$

x_{231} as edge $(u3,v22)$ with $x'(e) = \frac{1}{3}$ and another portion as edge $(u3,v23)$ with $x'(e) = \frac{2}{3}$. The resulting bipartite graph embeds the mentioned properties.

Observe that a minimum cost matching on the graph G is actually a feasible solution of the scheduling if the total energy cost is no more than C. In Figure 3.3, Step 3 of the P_{asym} algorithm computes the minimum cost matching in the bipartite graph G. Based on the properties of G, a maximum flow matching in G is a schedule for the EMMS problem. For example in Figure 3.4, the corresponding $(x_{111}, x_{221}, x_{231})$ of the matching $\{(u1,v11),(u2,v21),(u3,v22)\}$ is a schedule for the EMMS problem. To satisfy the energy budget constraint, we should find the minimum cost matching (MCM) on G, where the cost on any edge stands for the energy cost $c(e)$. R. Ahuja et al. [4] have shown that any basic feasible solution of the LP relaxation of the MCM problem is integral.

Lemma 3.3.2. *The minimum cost matching exists in $G(U,V,E)$ and the energy consumption of the matching is at most C.*

Proof. Because the fractional vector $x'(e)$ is a feasible solution of the LP relaxation of the MCM problem, and the minimum cost is no more than C, it is an upper bound of the optimal LP relaxation problem. According to [4], the basic feasible solution of the LP relaxation would be integral. Thus, the minimum cost matching exists and the total energy consumption is at most C. □

Theorem 3.3.2. *The makespan of the schedule generated from the P_{asym} algorithm is at most as twice as the optimal.*

Proof. The proof is similar to that in [88]. In the minimum cost matching A, there is at most one task scheduled on each node of the pe_i in V. Therefore, the makespan of the matching $T(A) \leq \sum_{h=1}^{q_i} \max(t_{e \in E_{ih}}(e))$. For the first node $v_{i1}(h=1)$, $\max(t_{e \in E_{i1}}(e)) \leq T^{\star}$, because $t(e) = t_{ijk}$ in G and Constraint (3.2) is satisfied. For the remaining nodes of pe_i,

$$\sum_{h=2}^{q_i} \max\left(t_{e\in E_{ih}}(e)\right) \leq \sum_{h=1}^{q_i-1} \min\left(t_{e\in E_{ih}}(e)\right)$$
$$\leq \sum_{h=1}^{q_i-1} \sum_{e\in E_{ih}} t(e)x'(e) \leq \sum_{h=1}^{q_i} \sum_{e\in E_{ih}} t(e)x'(e)$$
$$= \sum_{j=1}^{n} \sum_{k=1}^{l_i} t_{ijk}x_{ijk} \leq T^{\star}$$

The first inequality follows from Property iii. of the graph G. The second inequality follows from the definition of $\min\left(t_{e\in E_{ih}}(e)\right)$ and Property i. of the graph G. The fourth equality follows from the construction of $x'(e)$ and $t(e)$. The last inequality follows from the Constraint (3.1b) of the $\mathbb{P}2\mathbb{LP}$. Thus, P_{asym} is a 2-approximation algorithm. □

Complexity Analysis

Since P1LP-OPT performs Step 1 in both P_{sym} and P_{asym} algorithms, the computational complexity of P1LP-OPT influences the complexity of the proposed techniques. Simplex method is a well-known polynomial time algorithm to solve linear programming problems. Let C_o denote the computational complexity of the simplex method, which is polynomial. The computation complexity of the P1LP-OPT is $O(log(\frac{T_{UB}}{T_{LB}})C_o)$, because of the binary search.

Let $l = \max_{\forall i\in M}\{l_i\}$. In the P_{sym} algorithm, Step 1 dominates the overall complexity. Step 2 and Step 3 only take at most $O(nml)$. Thus, it is a polynomial time algorithm. In the P_{asym} algorithm, Step 1 is polynomial as discussed above. In Step 2, because the schedule S consists of at most $n+m$ positive x_{ijk}, the the sorting algorithm for the list $L_i(t)$ takes at most $O((n+m)\log(n+m))$ time by merge sorting for each PE. Step 3 is of polynomial complexity as it utilizes the simplex algorithm on the MMS problem. Step 4 is at most $O(nml)$. Thus, the computational complexity of P_{asym} algorithm is polynomial.

3.4 Experimental results

We evaluated the proposed techniques with extensive experiments that are presented in this section. In the case of homogeneous and heterogeneous CMPs, we analyzed the achieved approximation ratio with the effects of two factors : the CMP architecture and the task patterns. We compared the makespan generated from P_{sym}, P_{asym}, the ILP solver from [1] and the tight lower bound of $\mathbb{P}1$ (P1LP-OPT). In some cases, the ILP solver took an unbounded large amount of time to achieve an optimal. We set a timeout of 10000 seconds, after which the ILP solver returned the best suboptimal solution. In all the plots, the makespan values were normalized with respect to the P1LP-OPT (the tight lower bound

of the EMMS problem), which can directly reflect the actual approximation ratio[1]. The runtimes of the proposed techniques was also studied in comparison to the ILP solver with 8 hours timeout configuration.

Experimental Setup

We obtained the PE models from two commercial DVFS-equipped processors: IBM PowerPC [33] and Intel PXA270 [35]. We chose 6 v/f states for PowerPC ranging from 1V/1.0GHz to 1.25V/2.0GHz and 7 v/f states for PXA270 ranging from 0.85V/13MHz to 1.55V/624MHz. For homogeneous CMPs, the PowerPC was set as the PE unit to compose the multiple PE system. For heterogeneous CMPs, four combinations of the PowerPC and the PXA-270 were chosen as the target CMPs. We designed 4 task sets with different workload distributions: *equal, uniform, Gaussian* and *Poisson*. Each task set included 30 task nodes. The cycle number of the tasks were in the range of $[10^6, 10^{10}]$. For the energy budget, a metric named energy budget ratio $r(r \in [0,1])$ from [28] was introduced. With various r values, we set $\mathrm{C} = \sum_j (r \cdot (\max_{i,k}\{c_{ijk}\} - \min_{i,k}\{c_{ijk}\}) + \min_{i,k}\{c_{ijk}\})$, where $pe_i \in \Phi, \tau_j \in \Gamma, s_k \in \Psi_i$. The optimization techniques were coded in C++ and the experimentations were performed on a Pentium 4/2.4GHz/1GB WindowsXP PC.

Effect of CMP architecture and task patterns

We evaluated the proposed techniques by experimenting with the 4 task patterns. For the homogeneous CMP case, the number of PEs were varied from 4 to 16. For the heterogeneous CMP case, we designed four kinds of CMP with combinations of multiple PowerPC and PXA270. For the both cases, we compared the makespan generated from P_{sym}, P_{asym} and the ILP solver which are plotted in Figures 3.5 and 3.6. All the makespan values are normalized to the tight lower bound of $\mathbb{P}1$ generated from P1LP-OPT. Therefore, the actual approximation ratio is no more than the normalized makespan. Each CMP was plotted as a separate category. In each category, the results from 4 task sets were depicted from left to right in the order of equal, uniform, Gaussian and Poisson. The energy budget ratio was set as 0.5.

Homogeneous CMP The four target CMPs were designed as CMPs with 4, 8, 12, and 16 PowerPC PEs. As observed in Figure 3.5, the normalized makespan generated from P_{sym} is no more than 1.36 with all the task sets. With the 16 PEs, the results are better than ILP solver for the task sets with equal/Gaussian workload. In all cases, the average approximation ratio to the P1LP-OPT is 1.13, while the average ratio to the ILP is 1.06. With each task pattern, the average approximation ratio to the P1LP-OPT is below 1.15.

[1]The actual approximation ratio is no more than the normalized makespan w.r.t the P1LP-OPT. Even if there is an integrality gap between the LP relaxation and ILP problem, the normalized makespan is still meaningful. This is because the P1LP-OPT is a tight lower bound of the ILP problem.

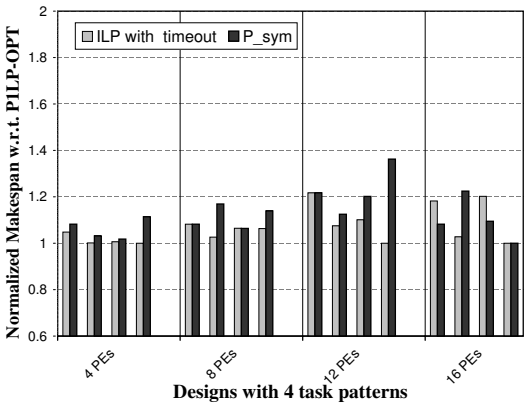


Figure 3.5: Evaluation on Homogeneous CMP architecture

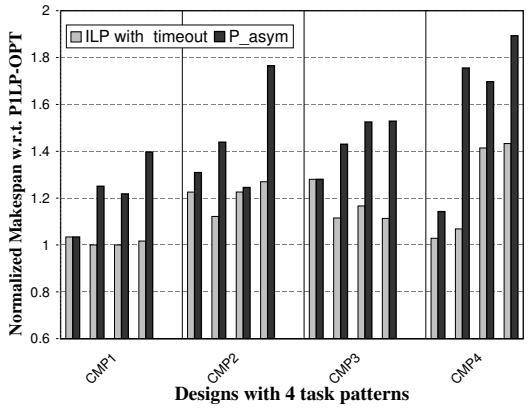


Figure 3.6: Evaluation on Heterogeneous CMP architecture

Heterogeneous CMP We designed 4 types of heterogenous CMPs with PowerPC and PXA270. We denote the PowerPC as *H* and the PXA270 as *L*. The 4 heterogeneous CMPs are plotted in the following order *1H3L, 1H8L, 1H16L, 2H16L*. As shown in Figure 3.6, the normalized makespan generated from P_{asym} (the upper bound of the actual approximation ratio) is within the theoretical bound of 2. In general, the normalized makespan for heterogeneous CMP is larger than that for the homogeneous CMP with all the task patterns. For the normalized makespan generated from the P_{asym}, the maximum ratio with comparison to the ILP is 1.64. In all the cases, the average normalized makespan of the P_{asym} is 1.43, while that of the ILP is 1.24.

Summary The actual approximation ratios of the schedules generated by P_{sym} and P_{asym} are within the theoretical bound. The task patterns have less effect on solution quality for the homogeneous CMP than that for the heterogeneous CMP.

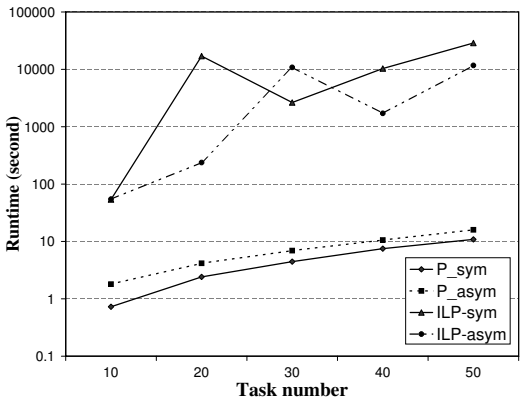


Figure 3.7: Runtime versus task number

Runtime

To evaluate the computational complexity of our techniques, we utilize synthetic task sets with up to 50 nodes to do experiment on the run time evaluation of the technique. These tasks have uniformly distributed execution time. The CMP with 4 PowerPC PEs was targeted for the homogeneous case and the *1H3L* CMP was targeted for the heterogeneous case. The energy budget ratio was set as 0.5. We compared the average runtime of P_{sym}, P_{asym} with the ILP solver (8 hours timeout setting) in Figure 3.7. The number of tasks in the task sets was varied from 10 to 50 nodes in steps of 10. Note that the y axis is in logarithmic scale in Figure 3.7. With up to 50 nodes, the P_{sym} and P_{asym} algorithms were completed within half a minute. The figure shows that the runtime of our techniques is linearly increasing with the increase of task numbers. As predicted, the P_{sym} is slightly faster than the P_{asym} algorithm because of the simplicity of the former. In comparison, the runtime of the ILP solver is exponentially large in some cases. Even with 10 nodes, the average runtime of the ILP solver is around 10 times of the P_{sym} and P_{asym}. With 50 nodes, the average runtime is beyond 8 hours and is actually more than 1000 times of that with our techniques. Therefore, the results demonstrate that the proposed techniques are efficient and applicable in practice.

3.5 Conclusion

In this work, we addressed the energy-efficient scheduling problem on CMP architectures with core-level DVFS. We proved that the EMMS problem is strongly NP-hard. We then proposed 2-approximation polynomial time techniques for both homogeneous and heterogeneous CMP. Our extensive experimentation with multiple workloads and CMP architectures demonstrate that our techniques can efficiently

generate solutions whose makespan is much lower than the factor of 2 in comparison to the optimal that is guaranteed by the approximation bound.

Chapter 4

Near Optimal Battery-Aware Energy Management

This chapter addresses the battery-aware energy management problem for a sequence of tasks with a deadline constraint. The objective is to maximize the battery lifetime while meeting a deadline constraint. We consider the nonlinear battery model proposed in [78] and propose optimal and approximation algorithms for the solution.

The work is organized as follows: Section 4.1 defines the problem, Section 4.2 presents the optimal and approximation algorithms as solutions, Section 4.3 discusses the experimental results and finally Section 4.4 concludes the work.

4.1 Problem Definition

Preliminaries

System model

Without loss of generality, we consider a battery-powered processor equipped with a set of discrete voltage/frequency (v/f) states and an idle state. A sequence of jobs is to be executed on the processor. The jobs have a deadline constraint. The duration and power consumption of each application in each v/f state are specified as inputs. In the battery-powered system, when the processor executes each job in a v/f state, the battery is discharged by a load current, which is proportional to the cube of supply voltage of the processor. When the processor stays idle, the system is shut down and thus no load current discharges the battery.

Battery model

Rakhmatov et al. [78] proposed a high-level model of the battery discharge process based on the electrochemical reaction of battery. A charged battery consists of symmetric positive and negative electrodes connected by an electrolyte with electroactive species. When a load is attached to the battery, the electrochemical reaction causes electroactive specie surface in the electrolyte to have a nonzero gradient. The apparent charge loss of the battery at time t, denoted by $\sigma(t)$, is the sum of two terms $a(t)$ (actual charge) and $u(t)$ (recoverable charge). $a(t)$ is the amount of actual charge consumed by the external system. Once $a(t)$ is consumed, it is lost permanently. $u(t)$ denotes the amount of charge that cannot be used at time t, because the gradient of electroactive specie surface makes some species unavailable at that moment. However, $u(t)$ can be recovered after enough idle time because the electrolyte diffusion causes the surface to eventually become flat. When the apparent charge loss $\sigma(t)$ exceeds the battery capacity,

the electroactive specie surface drops below a threshold and the battery reaction cannot be sustained. A real battery dies at this time t, which is the lifetime of the battery.

Based on the one-dimensional electrolyte diffusion behavior, the derived battery model in [78] takes load current profiles $I(\tau)$ as inputs and can predict the lifetime of battery, denoted as L, based on $I(\tau)$. The prediction error of the battery model has been validated to be within 5% on a physical lithium-ion battery [78]. The lifetime prediction is based on the following equation, for a given battery with capacity α and technical parameter β.

$$\alpha = \int_0^L I(\tau)d\tau + \int_0^L I(\tau)(2\sum_{c=1}^{\infty} \frac{e^{-\beta^2 c^2 (L-\tau)}}{\beta^2 c^2})d\tau$$

Therefore, different load profiles can lead to various lifetimes of battery. In order to maximize battery lifetime, Rakhmatov et al. [78] also defined the cost function $\sigma(t)$, given by

$$\sigma(t) = a(t) + u(t) = \sum_{k=1}^{n} I_k d_k + \sum_{k=1}^{n} I_k F(\beta, t_k^o, t_k^f, t) \tag{4.1}$$

where the two terms after the last equality sign are $a(t)$ and $u(t)$, respectively. The sequence $\{I_1, I_2, ..., I_n\}$ gives discrete load current values for a profile ending at time t, with d_k the duration of the load I_k. Further, t_k^o is the start time of load I_k and t_k^f the end time. The function F in the definition of $u(t)$ is given by

$$F(\beta, t_k^o, t_k^f, t) = 2\sum_{c=1}^{\infty} \frac{e^{-\beta^2 c^2 (t-t_k^f)} - e^{-\beta^2 c^2 (t-t_k^o)}}{\beta^2 m^2}$$

From the above, $a(t)$ is linear in the load current value and durations, while $u(t)$ is not and is affected by start time, end time and final time of the profile. The battery lifetime maximization problem reduces to finding a load profile $I(\tau)$ from time 0 to time t such that $\sigma(t)$ given by the battery model is minimized.

Problem description

The battery-aware energy management problem for a battery po- wered embedded system, denoted as $\mathscr{B}$, is described as follows.

Given

- an embedded processor equipped with a set of active voltage/frequency(v/f) states $\{s_1, s_2, ...s_l\}$ with $s_k = \langle v_k, f_k \rangle$;
- a sequence of n independent jobs $\mathscr{J} = \{J_1, J_2, ...J_n\}$, job J_i requiring power ρ_j at s_j and execution time d_{ij} at s_j;

- a deadline D for the job sequence;
- battery parameters α and β;

the objective is to obtain a v/f schedule $\mathscr{A}$ and idle time selection for recovery such that the apparent battery charge loss $\sigma(D)$ at time D is minimized and the deadline is met. We assume that the v/f state cannot be changed during the execution of a job.

Problem formulation

We formulate the idle time selection for recovery by introducing a recovery job J_s. J_s specifies that the processor stays in idle state and thus recovers (some) recoverable charge. J_s can be executed for various times. The upper bound on the execution time of J_s, as analyzed in [80], is $-\frac{\log(1.5\varepsilon)}{\beta^2}$, where ε is the recoverable ratio, empirically $0 < \varepsilon \leq 0.4$. Beyond this upper bound, the battery cannot recover any more even if the processor stays idle. Since we can only change the v/f state between job executions, we insert a J_s with variable idle time between each pair of jobs. Therefore, the new job sequence becomes

$$\mathscr{J}' = \{J_s, J_1, J_s, J_2, J_s, ..., J_n, J_s\} = \{J'_1, J'_2, ..., J'_{2n+1}\}$$

We discretize the interval $[0, -\frac{\log(1.5\varepsilon)}{\beta^2}]$ into l' steps [1]. Let x_{ij} be a binary variable indicating that J'_i chooses the j_{th} choice for execution. For each job with even index i (an active job), $x_{ij} = 1$ means the active job $J_{\frac{i+1}{2}}$ in $\mathscr{J}$ is executed in state s_j. For each job with odd index i (a recovery job) J_s, $x_{ij} = 1$ denotes that the processor stays idle with the j_{th} execution time in l' choices. We combine the choices for both active and recovery jobs into $\mathscr{V}|\mathscr{V} = r|$. When i is even (active job), $r = l$, which is the number of processor v/f states; when i is odd (recovery job), $r = l'$, which is the number of idle time choices for J_s. Denote the duration of J'_i with j_{th} choice by d_{ij}. Now the problem can be formulated as

$$\min \sigma(D) = (\sum_{i=1}^{2n+1} \sum_{j=1}^{r} a_{ij} x_{ij}) + u(D)$$

$$s.t. \sum_{i=1}^{2n+1} \sum_{j=1}^{r} d_{ij} x_{ij} \leq D \tag{4.2a}$$

$$\sum_{j=1}^{r} x_{ij} = 1, \ \forall J'_i \in \mathscr{J}' \tag{4.2b}$$

$$\sigma(t) < \alpha, \ \forall t \in [0, D); \sigma(D) \leq \alpha \tag{4.2c}$$

$$x_{ij} = \{0, 1\}$$

a_{ij} is the actual charge when J'_i is executed at s_j. For a recovery job, this is zero because system is shut down when idle. For an active job, it is proportional to the duration of the job at s_j times the cube of v_j. Constraint 4.2a makes sure all the jobs meet the deadline. Constraint 4.2b describes that one and

[1] The recovery interval can be discretized into steps with unequal lengths. For example, the first step with recovery time 0 means no idle time for recovery. The second step with recovery time 3 means that system sleeps and then wakes up, and the total time cost is 3. Thus the designer is able to specify the discretized steps based on system requirements.

only one choice is selected for execution of each job. Constraints 4.2c guarantees that no job will fail because the apparent charge loss never exceeds battery capacity. The two constraints can be combined in the single constraint $\sigma(t) \leq \alpha, \forall t \in [0, D]$ by designer's specification. The problem formulation aims at task sets with determined execution order and a common deadline. We assume the load currents of each job in each v/f states can be profiled statically. We also assume the load current in idle state is a constant (zero or non-zero).

Problem $\mathscr{B}$ can be proved to be NP-hard by reduction from Multiple Choice Knapsack Problem (MCKP) [78]. There are known dynamic programming techniques and fully polynomial approximation schemes (FPTAS) for solving MCKP [96]. However, as is often the case, a reduction from MCKP to a special case of $\mathscr{B}$ does not imply that the techniques can be directly applied to $\mathscr{B}$. We first present an optimal algorithm executed in pseudo-polynomial time as a solution. Next, we propose a fully polynomial approximation algorithm for the problem.

4.2 Algorithms

Optimal algorithm

The optimal algorithm for the $\mathscr{B}$ problem is a dynamic programming method that runs in pseudo-polynomial time. Dynamic programming is a method for a problem with optimal substructures. The optimal substructure generally exhibits the property that the optimal solution for a subproblem dominates other solutions and leads to the final optimal solution [66].

The proposed dynamic programming algorithm is based on that for the knapsack problem. However, there exist two critical differences. First, the battery model in the $\mathscr{B}$ involves a nonlinear factor (the recoverable charge) so that the problem formulation cannot be linearized as the knapsack problem. Next, the objective function (the apparent charge loss $\sigma(t)$) of $\mathscr{B}$ is a sum of a linear term (the actual charge $a(t)$) and a nonlinear term (the recoverable charge $u(t)$). This implies that a dynamic programming method based on two-dimensional table (job id and apparent charge loss σ) does not have optimal substructure. This is because an exact σ value could be any combination of a and u. Therefore, we propose a dynamic programming with a three-dimensional table (job id, deadline and actual charge) to solve the problem.

Consider a battery-aware energy management subproblem $\mathscr{B}(i,d,a)$ for the first i jobs in $\mathscr{J}'$ with exact execution time d and actual charge a. Let $S_{(i,d,a)}$ be the schedule for $J'_1, J'_2, ..., J'_i$ such that the overall execution time is no more than d, the actual charge is no more than a and the recoverable charge at time d is minimized. Let $u(i,d,a)$ be the minimum recoverable charge at time d. Here, i is in the

range of $\{1,2,...,N\}$. D and α are the respective upper bounds on d and a. If $S_{(i,d,a)}$ does not exist, let $u(i,d,a)$ be ∞. Initially, $u(1,d,a)$ is 0 for all possible d and a, because the battery is fully charged at the beginning. Thus, the recursive relationship for the dynamic programming is presented as

$$u(i,d,a) = min_{j \in V}\{u(i-1,d-d_{ij},a-a_{ij}) + \Delta u_{ij} | a+u < \alpha\} \tag{4.3}$$

Δu_{ij} is determined by the battery model for a given status of the battery and current choice for J_i'. It can be derived from [78,80]. To accurately calculate $u(t)$ when task J_i' is executed in s_j, we use the schedule associated with $u(i-1,d-d_{ij},a-a_{ij})$ and s_j by Equation 4.1. After the recursive step, we do a linear search for minimum $a+u$ when we find all $u(N,D,a)$ for all possible $a \in [1,\alpha]$. The optimum solution is then S^*, where

$$\sigma^*(D) = \{min(a+u(N,D,a)) | a+u(N,D,a) \leq \alpha\} \tag{4.4}$$

The dynamic programming algorithm based on the recursive relation is illustrated in Figure 4.1 and $BO(N,D,\alpha)$ is invoked for the optimal solution. The main idea of the algorithm is to construct a three-dimensional table $u(i,d,a)$. In each cell the minimum $u(i,d,a)$ is filled by the solution of the subproblem $\mathscr{B}(i,d,a)$. Each $u(i,d,a)$ is associated with a solution schedule S_h for the subproblem. Once the table is completed according to (4.3), linear search is used to find the minimum $\sigma(D)$ according to (4.4). In Figure 4.1, Line 1 describes the initialization of the table. Lines 2–14 illustrate the filling in of the table, following Equation (4.3). In Line 9, u_h is the accurate recoverable charge at time d based on the schedule S_h for the subproblem $\mathscr{B}(i-1,d-d_{ij},a-a_{ij})$ and s_j for current job J_i' from battery model. Lines 15–17 find the minimum $\sigma(D)$ and the associated schedule S^* when it exists.

Let us analyze the complexity of the algorithm called by $BO(N,D,\alpha)$. Line 1 takes at most $O(nD\alpha)$. Lines 2–14 constitute a loop with $O(nD\alpha)$ iterations, each calculating Equation (4.3). Since each calculation enumerates r choices for task J_i' and for each choice a trace back to J_1' is used to find the schedule S_h, the complexity of Equation (4.3) is $O(nr)$. Therefore, the complexity for Lines 2–14 is $O(n^2rD\alpha)$. Lines 15–17 take $O(\alpha)$ or $O(n)$. Thus, the overall computational complexity of the optimal algorithm is $O(n^2rD\alpha)$, which is pseudo-polynomial.

Approximation algorithm

In our definition, $\mathscr{B}$ is a *tricriteria problem* with the objective to minimize $\sigma(D)$ and two constraints (battery capacity α and deadline D). Consequently, *tricriteria approximation algorithms* [47] will be considered for the problem. Let $\mathscr{A}$ be a tricriteria approximation algorithm with quality bounds (c_1,c_2,c_3) for the problem, where c_1,c_2,c_3 are constants. If there exists a feasible solution for the problem, $\mathscr{A}$ must

$BO(n, d_{ub}, a_{ub})$/*$BO_m(n, d_{ub}, a_{ub}, \delta, K_a)$*/:

```
set u(1,d,a) = 0 and the others to be ∞;
for a = 1 : a_ub
  for i = 1 : n
   for d = 1 : d_ub
    u_min = ∞;
    for j = 1 : r
     find the cell (i−1, d−d_ij, a−a_ij);
     /* find the cell (i−1, d−d'_ij, a−a'_ij) */;
     trace back to J'_1 and get the schedule S_h for previous jobs;
     calculate u_h on schedule S_h for {J'_1, ...J'_{i−1}} and s_j for J'_i;
      if (u_h + a < α) and (u_h < u_min),
      /* if (u_h + K_a a < (1+δ)α) and (u_h < u_min), */;
        u_min = u_h and record the index j as j_h; end if;
     end for;
     fill in u_min as u(i,d,a) and record the choice j_h;
  end for; end for; end for;
  find the smallest σ = a + u(n, d_ub, a), ∀a = 1 : a_ub;
  /* find the smallest σ = K_a a + u(n, d_ub, a), ∀a = 1 : a_ub; */
  if σ ≤ α, trace back and return schedule S*;
  /* if σ ≤ (1+δ)α, trace back and return schedule S^A; */
  else return null;
```

Figure 4.1: Optimal algorithm for the $\mathscr{B}$ problem (Comments are the modification for BO_m procedure invoked by the approximation algorithm)

find a schedule with the final apparent charge loss σ no more than $c_1\sigma^*(D)$, modified deadline c_2D and relaxed battery capacity $c_3\alpha$. If there is no solution for the problem, $\mathscr{A}$ should report this or provide a feasible schedule with modified deadline c_2D and relaxed battery capacity $c_3\alpha$. If the complexity of $\mathscr{A}$ is fully polynomial in problem size, we call it a fully polynomial (c_1, c_2, c_3) approximation algorithm. We next describe a tricriteria approximation algorithm with a designer-specified parameter δ $(0 < \delta < 1)$ and prove the proposed algorithm is a fully polynomial $(1+2\delta, 1+\delta+\frac{\delta}{N}, 1+\delta)$ approximation algorithm.

The algorithm is described in Figure 4.2. The main idea of the algorithm is similar to the FPTAS for the knapsack problem [96]. The main algorithm in Figure 4.2 does a binary search for the smallest a value on which a *test* procedure succeeds. Thus, the schedule S^A generated from the smallest a is a provably approximate solution. The procedure *test* invokes a modified dynamic programming procedure BO_m for the scaled and rounded problem. BO_m is similar to the optimal algorithm BO of Figure 4.1. The only difference is that BO_m works for the scaled problem with relaxed battery capacity. Note that only the delay d and actual charge a are scaled and rounded, while the recoverable charge u remains non-scaled and is calculated from non-scaled data. Next, we show $BA(\alpha, D, \delta)$ is a tricriteria approximation algorithm with quality bounds $(1+2\delta, 1+\delta+\frac{\delta}{N}, 1+\delta)$. Then, we argue the computational complexity

$BA(\alpha, D, \delta)$:

1 $g = \lceil \lg \alpha \rceil$,
2 $l = 1, r = g$;
3 binary search for the smallest 2^b s.t. $test(2^b, \delta)$ returns success;
(if $2^b > \alpha$, use $test(\alpha, \delta)$ to do binary search;)
4 $\mathscr{S}^A = test(2^b, \delta)$;
5 return $\mathscr{S}^A$;

$test(a, \delta)$:

6 $K_a = \frac{\delta a}{N}, a'_{ij} = \lceil \frac{a_{ij}}{K_a} \rceil, a'_{ub} = \lceil \frac{a}{K_a} \rceil + N = \lceil \frac{N}{\delta} \rceil + N$;
7 $K_d = \frac{\delta D}{N}, d'_{ij} = \lceil \frac{d_{ij}}{K_d} \rceil, d'_{ub} = \lceil \frac{D}{K_d} \rceil + N = \lceil \frac{N}{\delta} \rceil + N$;
8 $\mathscr{S}^A = BO_m(N, d'_{ub}, a'_{ub}, \delta, K_a)$;
9 if ($\mathscr{S}^A != null$) return $\mathscr{S}^A$ and *success*;
10 else return $failure$; endif;

Figure 4.2: Tricriteria approximation algorithm for $\mathscr{BA}$ problem

of the algorithm is fully polynomial.

Let S^* be the optimal schedule and a_{S^*} the actual charge loss based on S^*. Let $u_{S^*}(t)$ and $\sigma_{S^*}(t)$ be the recoverable and apparent charge loss at time t. Thus, we have $\sigma_{S^*}(t) = a_{S^*} + u_{S^*}(t)$. Note that, when $t \leq D$, S^* in the equation is an initial part of the schedule. The optimal charge loss at time D is $\sigma_{S^*}(D)$. Then, we find two properties for $test$ procedure.

Lemma 4.2.1. *Suppose $a_{S^*} \leq a_h \leq \gamma a_{S^*}$ for some $\gamma \geq 1$. If $test(a_h, \delta)$ returns success and S^A, let t_f be the latency of the N jobs based on S^A. The charge loss generated by S^A at t_f, denoted by $\sigma_{S^A}(t_f)$, is no more than $(1+\gamma\delta)\sigma_{S^*}(D)$.*

Proof. We derive the following equations.

$$\begin{aligned}
\sigma_{S^A}(t_f) &= a_{S^A} + u_{S^A}(t_f) \leq K_a a'_{S^A} + u_{S^A}(t_f) \\
&= \sum_{S^A} K_a a'_{ij} + u_{S^A}(t_f) \leq \sum_{S^*} K_a a'_{ij} + u_{S^*}(D) \\
&\leq \sum_{S^*} (K_a + a_{ij}) + u_{S^*}(D) \leq K_a N + a_{S^*} + u_{S^*}(D) \\
&= \delta a_h + \sigma_{S^*}(D) \leq (1+\gamma\delta)\sigma_{S^*}(D) \qquad (4.5a)
\end{aligned}$$

Step 1–3 follow from definitions of σ and a'_{ij}. The fourth step is true because S^A is optimal for the scaled problem. The fifth step follows because $a_{ij} \leq K_a a'_{ij} \leq K_a + a_{ij}$. Steps 6–7 follow from the definitions of a_{S^*}, $\sigma_{S^*}(D)$ and K_a (note that $K_a = \frac{\delta a_h}{N}$ here). The last step follows from the assumption. □

Similarly, we have the following lemma.

Lemma 4.2.2. *If $test(a,\delta)$ returns failure, the actual charge loss a_{S^*} for the optimal schedule S^* is bigger than a.*

Proof. We prove the lemma by contradiction. Suppose $a_{S^*} \leq a$. Let a'_{S^*} and d'_{S^*} be the actual charge loss based on S^* and the latency for all the jobs after scaling and rounding at Line 6 and 7 in Figure 4.2. First we show a'_{S^*} and d'_{S^*} are no more than the upper bounds (a'_{ub} and d'_{ub}) of search in BO_m procedure.

$$\begin{aligned} a'_{S^*} &= \sum\nolimits_{S^*} a'_{ij} = \sum\nolimits_{S^*} \lceil a_{ij}/K_a \rceil \leq \sum\nolimits_{S^*} a_{ij}/K_a + N \\ &= a_{S^*}/K_a + N \leq a/K_a + N \leq \lceil a/K_a \rceil + N = a'_{ub} \end{aligned}$$

The first step expands the actual charge loss job by job. The second step follows from the definition of a'_{ij} at Line 6 in Figure 4.2. The third step is true because $\lceil \frac{a_{ij}}{K_a} \rceil \leq \frac{a_{ij}}{K_a} + 1$. The fourth step follows from the definition of a_{S^*}. The fifth step is true because of the assumption $a_{S^*} \leq a$. The remaining steps follow from the definition of a'_{ub}. Similar to the above, we have the following for d'_{S^*}.

$$\begin{aligned} d'_{S^*} &= \sum\nolimits_{S^*} d'_{ij} = \sum\nolimits_{S^*} \lceil d_{ij}/K_d \rceil \leq \sum\nolimits_{S^*} d_{ij}/K_d + N \\ &= D/K_d + N \leq \lceil N/\delta \rceil + N = d'_{ub} \end{aligned}$$

Next we show S^* in the scaled problem does not violate the $(1+\delta)$-relaxed battery capacity constraint. Let $\sigma'_{S^*}(t)$ be the charge loss at time t based on S^* after scaling (a'_{ij} is first scaled down by K_a, then scaled up by K_a when calculating $\sigma'_{S^*}(t)$ at Lines 10 and 15 in BO_m). We first show $\sigma'_{S^*}(D) \leq (1+\delta)\alpha$, then show $\sigma'_{S^*}(t) < (1+\delta)\alpha$, $\forall t \in [0,D)$.

$$\begin{aligned} \sigma'_{S^*}(D) &= K_a \sum_{S^*} a'_{ij} + u_{S^*}(D) = K_a \sum_{S^*} \lceil \frac{a_{ij}}{K_a} \rceil + u_{S^*}(D) \\ &\leq K_a (\sum_{S^*} \frac{a_{ij}}{K_a} + N) + u_{S^*}(D) \\ &= a_{S^*} + K_a N + u_{S^*}(D) = \sigma_{S^*}(D) + \delta a \leq (1+\delta)\alpha \end{aligned} \tag{4.8a}$$

Steps 1–2 and 4–5 follow from the definitions of $\sigma'_{S^*}(D)$, a'_{ij}, a_{S^*} and $\sigma^*(D)$. The third step follows because $\lceil \frac{a_{ij}}{K_a} \rceil \leq \frac{a_{ij}}{K_a} + 1$. The last step is true because $a \leq \alpha$ and $\sigma_{S^*}(D) \leq \alpha$.

When $t \in [0,D)$, S^* and D are replaced by an initial part of S^* and t in Equation (4.8a). Therefore, because $\sigma_{S^*}(t) < \alpha$ in the last step, we have $\sigma_{S^*}(t) + \delta a < (1+\delta)\alpha$.

So far, we have showed i) a'_{ub} and d'_{ub} are the upper bounds of search in BO_m and ii) battery capacity constraint is not violated for S^* in the scaled problem. We can conclude that procedure $test(a,\delta)$ returns success. Therefore, it is a contradiction that $test(a,\delta)$ returns failure. □

Then, we show BA is a tri-criteria approximation algorithm.

Lemma 4.2.3. *$BA(\alpha,D,\delta)$ generates a schedule S^A with charge loss no more than $(1+2\delta)$ times the optimal charge loss σ^* when the deadline D is relaxed to $(1+\delta+\frac{\delta}{N})D$ and the battery capacity α is relaxed to $(1+\delta)\alpha$.*

Proof. We first show the approximation ratio of charge loss based on S^A, then show the relaxation ratios of deadline and battery capacity constraints.

Let 2^b be the value returned by binary search at Line 3 in Figure 4.2. Thus S^A is returned as solution in Line 4 by $test(2^b,\delta)$. Let the latency of all the jobs based on S^A be d_{S^A}. Denote by $\sigma_{S^A}(t)$ the charge loss at time t based on S^A. We have two cases:
Case I: $2^b \le a_{S^*}$. By Lemma 4.2.1, $\sigma_{S^A}(d_{S^A}) \le (1+\delta)\sigma_{S^*}(D)$.
Case II: $2^b > a_{S^*}$. We have $2^b < 2a_{S^*}$ because 2^b is the smallest value for which $test$ returns success. Again by Lemma 4.2.1, $\sigma_{S^A}(d_{S^A}$
$) \le (1+2\delta)\sigma_{S^*}(D)$.

Next, we derive the upper bound on d_{S^A}.

$$\begin{aligned} d_{S^A} &= \sum\nolimits_{S^A} d_{ij} \le \sum\nolimits_{S^A} K_d d'_{ij} \le K_d(\lceil N/\delta \rceil + N) \\ &\le \delta D/N(N/\delta + N + 1) = (1+\delta+\delta/N)D \end{aligned}$$

The third step is true because the deadline upper bound of BO_m is $\lceil \frac{N}{\delta} \rceil + N$. The other steps are similar to previous arguments. For $t \in [0,D]$, we have

$$\sigma_{S^A}(t) \le K_a N + a_{S^*} + u_{S^*}(t) \le \alpha\delta + \sigma_{S^*}(t) \le (1+\delta)\alpha$$

The first step follows from proofs of Lemma 4.2.1. The second step is true because the testing value for $test$ procedure is no more than α as described at Line 3 in Figure 4.2. Thus $K_a N \le \alpha\delta$. The last step is true because $\sigma_{S^*}(t) \le 0$. The inequality in the last step becomes $<$ when $t \in [0,D)$. □

Lemma 4.2.4. *The computational complexity of $BA(\alpha,D,\delta)$ is $O(\frac{n^4 r}{\delta^2}\lg\lg\alpha)$.*

Proof. The *test* procedure takes $O(n^2 r a'_{ub} d'_{ub}) = O(\frac{n^4 r}{\delta^2})$ since both a'_{ub} and d'_{ub} are $\lceil \frac{N}{\delta} \rceil + N$ by definition. The *BA* procedure invokes *test* procedure $\lg\lg\alpha$ times. Therefore, the overall complexity is $O(\frac{n^4 r}{\delta^2} \lg\lg\alpha)$. □

Thus, we have the following theorem.

Theorem 4.2.1. *$BA(\alpha, D, \delta)$ is a fully polynomial $(1+2\delta, 1+\delta+\frac{\delta}{N}, 1+\delta)$ approximation algorithm.*

4.3 Results

Experimental setup

We consider an experimental setup similar to that in [78]. The processor is equipped with four supply voltage states (v_0, v_1, v_2, v_3), whose scaling factors are $(1.0, \frac{1}{0.8}, \frac{1}{0.6}, \frac{1}{0.4})$. The frequency in each state is proportional to the supply voltage. The battery capacity α and technical parameter β are set as 4037 (unit:10^{-2}Amin) and 0.273 (unit:min$^{-1/2}$) respectively. Initially the battery is fully charged. The load (or battery) current in each state is proportional to the cube of scaling factors of processor supply voltages.

We apply the proposed techniques for 3 realistic applications and 5 synthetic task sequences. The realistic robot arm controller application includes 9 jobs and is taken from [78]. The load current is the same as that in [78]. The duration of jobs at lowest voltage v_0 is proportional to the supply frequency and execution cycle numbers. The multimedia benchmark I includes a sequence of 7 multimedia jobs with load currents and durations at lowest voltage v_0 taken from [21]. The multimedia benchmark II includes a sequence of 8 multimedia jobs taken from Mediabench [59]. The cycle numbers of these jobs are achieved by SimpleScalar simulations [89] with inputs in Mediabench [59]. The load current of each job at v_0 is set as $50mA$. The durations of jobs are calculated from the frequency and cycle numbers. The 5 synthetic job sequences include 15 or 20 jobs whose cycle numbers are randomly generated. The load current and job durations are calculated similar to those of multimedia benchmark II.

We define an emergency ratio, e_r, as the ratio between the deadline and the summation of execution times of active jobs at v_0 (lowest v/f state). Smaller the e_r is, tighter the deadline setting is. The emergency ratio of all job sequences is set as 0.8 in Section 4.3 and 4.3.

We evaluate the proposed techniques (*BO* and *BA*) with comparison to the existing heuristic algorithm on single processor in [21] (named *C&C* algorithm). *C&C* algorithm starts from an initial solution with highest supply voltage for each job and then repairs battery failures by scaling down v/f

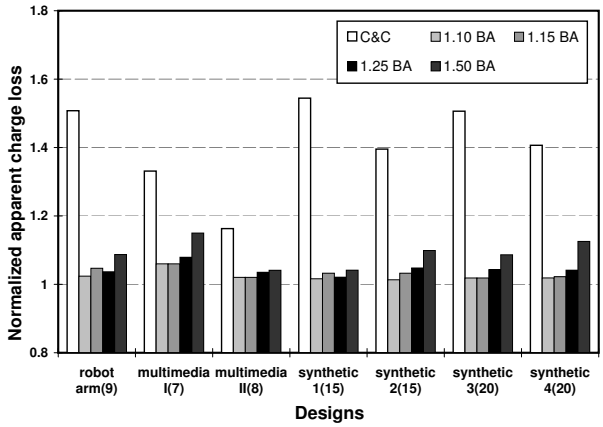


Figure 4.3: Normalized apparent charge loss with comparison to existing technique [21]

state of tasks. After the repairing step and a further scaling-down of v/f states of tasks, *C&C* generates a heuristic solution. For a fair comparison to *C&C* algorithm, the inputs to our *BA* algorithms in experiments addressed in Section 4.3 are modified to $(\frac{\alpha}{1+\delta}, \frac{D}{1+\delta+\frac{\delta}{N}}, \delta)$ because *BA* is a tricriteria approximation algorithm. Thus, the solutions generated by *BA* with the modified input setting satisfy the battery capacity α and deadline D constraints.

We also verify the theoretical bounds of *BA* by experiments. We set the inputs to our *BA* algorithms as (α, D, δ) in experiments addressed in Section 4.3. Then, we compare the results from *BO* and *BA*, and verify the approximation quality bounds. The final apparent charge loss, the actual latency and the peak apparent charge loss are recorded for each solution schedule.

To illustrate the effect of deadline on apparent charge loss, we evaluate the proposed techniques with the robot arm controller application with different emergency ratios. We slide the emergency ratio from 0.5 to 1.4 and plot the results generated by *BO* and *BA* with input setting (α, D, δ) in Section 4.3.

The techniques were coded in C++ and simulations were performed on an Intel Core 2 Quad / 2.66GHz / 3GB Windows XP PC.

Comparison to existing technique

We use 3 realistic and 4 synthetic job sequences to evaluate algorithms *BO* and *BA* with modified input settings and compare with *C&C*. The final apparent charge losses by *C&C* and *BA* with δ values $(\delta = 0.05, 0.075, 0.125, 0.25)$ are depicted in Figure 4.3 and are normalized with the optimal results by *BO*. The *BA* with δ value is simply denoted as $(1+2\delta)BA$ in the figure, for example, *BA* with $\delta = 0.05$ is denoted as $1.10BA$. The x axis lists the job sequence name and numbers.

Table 4.1: Apparent charge loss and latency approximation with respect to *BO*

Jobs	1.10BA/BO		1.15BA/BO		1.25BA/BO		1.50BA/BO	
	σ	latency	σ	latency	σ	latency	σ	latency
robot arm (9)	93.3%	1.03	89.5%	1.05	84.1%	1.08	74.3%	1.16
multimedia I(7)	92.6%	1.04	88.4%	1.05	81.5%	1.10	65.3%	1.20
multimedia II(8)	89.9%	1.04	86.8%	1.06	82.5%	1.10	70.7%	1.21
synthetic 1(15)	93.8%	1.04	90.7%	1.05	83.6%	1.10	70.6%	1.21
synthetic 2(15)	93.4%	1.04	89.4%	1.06	83.2%	1.10	74.3%	1.21
synthetic 3(20)	93.3%	1.03	89.9%	1.06	85.0%	1.09	73.3%	1.21
synthetic 4(20)	90.9%	1.04	89.4%	1.05	84.7%	1.09	75.0%	1.20
synthetic 5(15)	n/a	n/a	103.5%	1.06	99.1%	1.09	86.0%	1.19
on average	92.4%	1.04	91.0%	1.06	85.5%	1.09	73.7%	1.20

As is clear from the figure, *C&C* gives inferior results in comparison to the optimal and *BA* algorithms. In fact on average, the *C&C* is over 1.40 (max=1.54) times the optimal. In contrast, 1.10*BA* and 1.15*BA* are very close to the optimal in all cases and on average outperform *C&C* by 27.2% and 26.6%, respectively. Even with a quality bound setting as 0.50, on average, *BA* is is within 1.09 (max=1.15) of the optimal and outperform *C&C* by 22.5%.

On average for 3 realistic applications, the run time of *C&C* is within one second, the run time of optimal algorithm is 372 seconds and the run time of 1.50BA is 3.4 seconds.

In summary, *BO* and *BA* with various δs and modified input settings outperform *C&C* in all cases. And the solutions by *BA* with modified input settings are quite close to the optimal.

Verification of approximation approaches

We use 3 realistic and 5 synthetic job sequences to evaluate the proposed *BA*. The results generated by *BA* with δ values $(\delta = 0.05, 0.075, 0.125, 0.25)$ are depicted in Table 4.1. In the left-most column of table, the job sequence name and job numbers are listed. Then, the next columns show the apparent charge losses and deadlines of schedules from *BA*. All the results generated from *BA* are normalized by results from *BO*. As we can see, 1.10*BA* generates a schedule that consumes 93.2% of the optimal charge loss with a 1.03 relaxation of deadline. When $\delta = 0.05$, the theoretical bounds on apparent charge loss, deadline and battery capacity are $(1.10, 1.05 + \frac{0.05}{N}, 1.05)$. The table shows the performance of schedules generated by *BA* are all within theoretical bounds for the first 7 job sequences. For the last job sequence (synthetic 5), there exists no feasible solution. 1.10*BA* reports no solution found, and 1.15*BA*, 1.25*BA* and 1.50*BA* generate solutions with relaxed deadline and battery capacity.

On average, 1.10*BA* can generate a schedule with 1.04 relaxation of deadline, which only consumes 92.4% of charge loss for the optimal schedule. The actual charge loss bound is far less than

the theoretical bound, which is 1.10 in this case. Similarly, the actual charge loss bounds of schedules generated from 1.15*BA*, 1.25*BA* and 1.50*BA* are also much less than theoretical bounds. However, the actual deadline relaxation bounds are close to the theoretical bounds. For example, the actual latency of 1.10*BA* is 1.04 comparing to the theoretical bound 1.05. We also record the peak apparent charge loss of schedules. Results show that none of the first 7 applications has peak apparent charge loss more than battery capacity. For the last job sequence, the 1.15*BA*, 1.25*BA* and 1.50*BA* find solutions and the peak apparent charge loss is within $1+\delta$ of battery capacity. We also notice that *BO* and *BA*s with various δ values achieve a trade-off between quality bounds and run time. While *BO* takes about half an hour for optimal solution on multimedia benchmark II, *BA* with a quality bound of 50% only needs about ten seconds for an approximated solution.

In summary, *BA* with different δ values generate schedules with- in quality bounds. On average, the actual bound on apparent charge loss is far less than the theoretical quality bound.

Effect of deadline settings

We demonstrate the effect of deadline settings by the optimal and approximation algorithms by simulating the robot arm job sequence with various emergency ratios from 0.5 to 1.4. Figure 4.4 plots the apparent charge loss by *BO* and *BA* algorithms with emergency ratio settings. The x axis is the actual latency of schedules normalized to the summation of execution times of active jobs at v_0 (lowest v/f state). The y axis is the apparent charge loss. For *BO* algorithm, the normalized latency is the emergency ratio setting. For *BA* algorithms, the normalized latency is calculated by the actual delay of schedules generated by *BA*s, because *BA* with a designer-defined δ generates a schedule with a $1+\delta$ relaxation of deadline.

The plots in the figure show the trade-off between deadline setting and battery charge loss is convex. Note that the peak apparent charge losses of all the *BA*s are no more than battery capacity 4037. Thus, the designer can choose an appropriate deadline setting for a given battery-powered system by analyzing the convex curves. Interestingly, we also found that the curves from *BO* and *BA*s are overlapped together. *BA* algorithms can find a schedule with charge loss close to a point in the optimal curve. This implies that the overlapped curves from *BA*s could provide a rough prediction of optimal charge loss.

Since the plots are overlapped together in Figure 4.4, we re-plot the apparent charge loss $-$ latency data when e_r settings are 0.8 and 0.9 in Figure 4.5. The data points inside dash eclipses are generated by *BA* algorithm with $e_r = 0.8$. The figure shows that *BA* with increased δ generate schedules

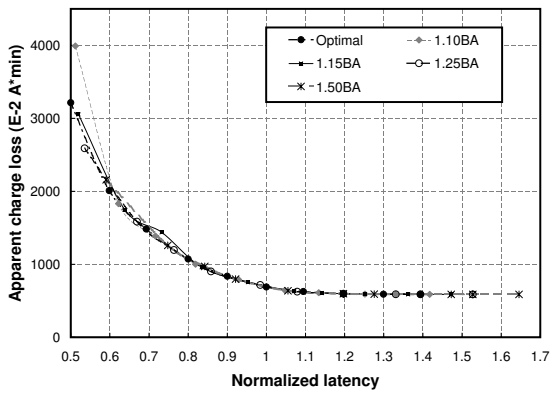


Figure 4.4: Effect of deadline settings

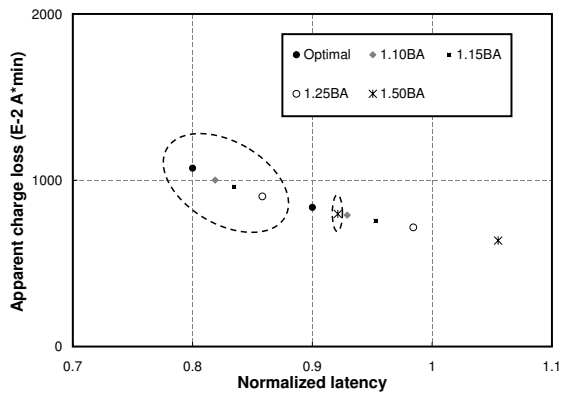


Figure 4.5: Illustration figure for effect of deadline settings

with decreased apparent charge losses and larger delays. For example, when $e_r = 0.8$, the apparent charge loss generated by $1.50BA$ is less than that by $1.25BA$, but with larger latency. We also notice that the apparent charge loss by $1.50BA$ with $e_r = 0.8$ is very close to the optimal with $e_r = 0.9$. This is also true for most of the un-plotted data. Smaller the e_r setting is, the tighter the deadline setting is. Therefore, we can predict the optimal by *BA*s with large δ but with tight deadline settings.

In summary, the $\sigma - latency$ curves generated by *BA*s can roughly predict the optimal solution with appropriate deadline setting.

4.4 Conclusions

We consider a battery model with nonlinear dependency on load current profiles and address a battery-aware energy management problem. We target a sequence of jobs with a deadline constraint executing on a battery-powered processor equipped with discrete v/f states. In order to maximize battery lifetime,

the problem is formulated by introducing recovery jobs and the objective becomes to minimize the final apparent charge loss of battery. Since the problem is NP-hard, we propose a pseudo-polynomial time optimal algorithm and a fully polynomial time approximation algorithm as solutions. Experimental results show that the proposed algorithms outperform existing technique [21] and the approximation algorithms are able to predict the optimal with appropriate deadline settings.

Chapter 5

Thermal aware scheduling for periodic applications

The chapter addresses a thermal-aware performance optimization problem on embedded processors for periodic applications with deterministic execution times. The optimal and fully polynomial approximation algorithms are provided as solutions. The work is organized as follows: Section 5.1 describes system level power and thermal model, Section 5.2 defines the problem, Section 5.3 discusses the previous work, Section 5.4 presents the optimal and a fully polynomial approximation scheme for the problem, Section 5.5 discusses the experimental results, and finally Section 5.6 concludes the work.

5.1 Preliminaries

Current modern embedded processors are usually equipped with a set of discrete voltage/frequency states. Voltage and frequency are respectively the supply voltage and operating frequency to execute applications. Examples of embedded processor architectures include the Intel StrongARM 1100, Intel PXA 270, Intel IXP 2400, Freescale MPC8641D, TI OMAP, Nvidia MediaQ Katana and so on. We present system-level power consumption model and thermal model for this work.

System level power consumption model

The power consumption and frequency of single processor in a particular active state is function of the operating voltage and frequency. The function can be specified by the designer's characterization of system power. For example, the power consumption can be characterized by dynamic and static power consumption of devices required for executing a task. Our technique is independent of a specific power model, and is applicable when the power consumption is function of the task characteristics in addition to operating voltage and frequency. The switching overhead between various active states and from active to sleep state is considered to be negligible [68]. The wake-up overhead from the sleep state is assumed to be a processor dependent constant.

System-level thermal model

We utilize a simple first-order lumped RC model proposed by Sabry et el. [84] and frequently used in recent system-level thermal aware design techniques [7–9,20,22,55,79,98,99] to capture the heat transfer phenomena. It models the steady and transient heat transfer behaviors of many advanced embedded processors. These processors are generally deployed in an environment with limited cooling assemblies. They has well-defined hotspots and it is acceptable to assume a uniform temperature distribution across package [48, 98]. Further, hardware OEMs often incorporate one thermal diode on one processor, for

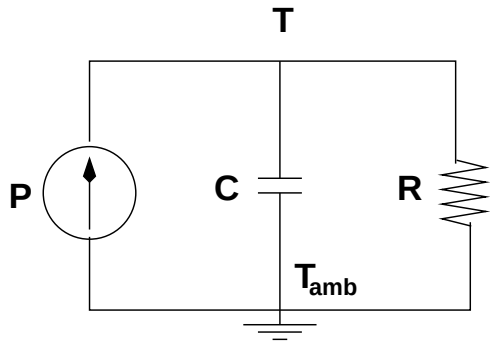


Figure 5.1: Processor heat transfer model

example, the Intel Core Duo [25] and Intel Pentium 4 [34]. Based on these considerations, the model is applicable for system level design.

Our thermal model is showed in Figure 8.2. In the figure P (unit W) denotes the power consumption of the processor at current time t, T (unit $^\circ C$) denotes the die temperature, C (unit $J/^\circ C$) denotes the thermal capacitance of the system, R (unit $^\circ C/W$) denotes the thermal resistance, and T_{amb} (unit $^\circ C$) denotes the ambient temperature. The thermal parameters (R and C) can be achieved by optimization techniques for thermal modeling [82–84]. The relationship between die temperature and processor power dissipation can be modeled by:

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

Assuming an initial die temperature of T_o at time 0 and P remains unchanged during time period $[0,t]$, the final temperature after time t is given by:

$$T = P \cdot R + T_{amb} + (T_o - P \cdot R - T_{amb}) \cdot e^{-\frac{t}{RC}} \tag{5.2}$$

The temperature change during the time t is denoted by $\Delta T = T - T_o$. As $t \to \infty$, T approaches a steady state temperature of $RP + T_{amb}$. In the sleep state the temperature gradually approaches T_{amb}. Typical transition time to steady state temperatures is of the order of several hundreds of milliseconds [92].

5.2 Problem definition

The thermal-aware performance optimization problem TA_{min} can be described as follows.

Given:

- a processor with one sleep state s_{sleep} with power consumption ρ_{sleep}, a set of active voltage/frequency states $M(|M| = m)$ with power consumption ρ_j $(1 \leq j \leq m)$ in state $s_j \in M$, thermal resistance as R and thermal capacitance as C,

- a periodic sequence of n jobs $J = \{J_1, J_2, ..., J_n\}$ with t_{ij} denoting the run time of job J_i at voltage/frequency state s_j,
- an initial temperature (at time $t = 0$) T_o and peak temperature constraint T_{max},

obtain an assignment of one active voltage state for each job, and select the processor sleep times such that the total execution time of the n jobs is minimized subject to the temperature constraint.

The problem as described is a discrete optimization problem with nonlinear continuous feedback constraint. In the remainder of the work we use jobs and tasks to refer to the same entity. In our problem definition each task executes in a single active state of the processor. The final temperature on job completion is determined by the thermal model described in the previous section. The processor can go into sleep mode on completion of a job, and before the start of the next job. Our problem definition considers that the task set executes in a periodic manner. The designer specifies an initial temperature T_o, and the feasible schedule for the problem should guarantee that the job set is executed by the schedule in multiple runs (periodic manner) without peak temperature violation. Therefore, we impose an additional constraint for the periodic characteristic of the job set that the final temperature T_f after one complete execution of the task set must be less than or equal to T_o ($T_f \leq T_o$). This constraint is essential to ensure that the periodic execution of the task set does not violate the temperature constraint.

We incorporate the sleep modes in the problem formulation by considering a sequence of $N = 2n+1$ jobs $J' = \{J'_1, J'_2, ..., J'_{2n+1}\}$. Each J'_i when i is an even number refers to the job $J_{i/2}$ from the original set, and when i is odd refers to a job J_s that denotes that the processor is in sleep state. Assume the maximum cooling transient time is t_{ms}, estimated by cooling the processor from T_{max} to T_{amb} in sleep mode. The execution time of J_s is in the range of $[0, t_{ms}]$. A sleep time of more than t_{ms} lowers the performance in terms of more execution time with no reduction in temperature. We consider the range $[0, t_{ms}]$ as q distinct values $\{t_1, t_2, \ldots t_q\}$ in increments of $t_{ms}/(q-1)$. Thus, if $t_{ms} = 100$ and $q = 11$ we consider the following values $\{0, 10, 20 \ldots 100\}$. We assume that length of the sleep interval is selected from one of the distinct values in the range. Note that 0 belongs to the distinct set of values and it implies that the processor does not go into the sleep mode.

We integrate the decision problem associated with sleep and active state jobs by considering that each job J'_i has $r = (m \text{ or } q)$ different choices ($r = m$ if i is even, else $r = q$), and each choice has an associated execution time given by t_{ij} ($1 \leq i \leq 2n+1, 1 \leq j \leq r$). Thus, TA_{min} is formulated as follows:

$$
\begin{aligned}
TA_{min}: \quad & \min Z = \sum_{i=1}^{2n+1} \sum_{j=1}^{r} t_{ij} x_{ij} \\
s.t. \quad & RC \cdot T' + T - RP = T_{amb}; && (5.3) \\
& \sum_{j=1}^{r} x_{ij} = 1, \forall i \in [1, 2n+1]; && (5.4) \\
& x_{ij} = \{0,1\}; T \leq T_{max}; && (5.5) \\
& T(t=0) = T_o; T(t=Z) \leq T_o;
\end{aligned}
$$

If i is even and $x_{ij} = 1$ the solution to the above formulation denotes that job $J_{i/2}$ executes in active state s_j for time t_{ij}. Similarly, when i is odd and $x_{ij} = 1$ the processor enters the sleep state for time t_{ij}. We assume the various time values in the problem formulation are integral (for example they could be specified in clock cycles). The above formulation includes a non-linear thermal constraint, where P is a variable of time, determined by the state choice. However, even if the thermal model were linear, the problem can be shown to be NP-hard.

Theorem 5.2.1. *TA_{min} is NP-hard.*

Proof. Consider a special case of the TA_{min}. We assume that processor sleeps only toward the end for t_{ms} time. Further, we assume that the initial temperature is close to T_{amb} and the maximum run time of each job is small enough such that the thermal curve is linear. The special case implies that the thermal curve of a feasible schedule would be monotonically increasing. The maximum temperature is achieved on the completion of all jobs. Thus, the objective function can be specified in terms of the execution time of actual jobs (without the sleep jobs) $\min Z = \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} x_{ij}$. As we consider that the thermal curve is linear, thermal constraint (5.3) and (8.9d) can be replaced by $T_o + \sum_{i=1}^{n} \sum_{j=1}^{m} \Delta T_{ij} x_{ij} \leq T_{max}$ where ΔT_{ij} denotes the temperature increase due to the execution of job J_i in active state s_j.

The special case of TA_{min} can be shown to be NP-hard by a polynomial reduction from the well known multiple-choice knapsack problem (MCKP), which is NP-hard. Let t_{max} be the upper bound on the execution time of any job, that is $t_{max} = \max(t_{ij}), \forall J_i \in J, j \in m$. The saving in execution time due to a job J_i operating in active state s_j is given by $t_{max} - t_{ij}$. Finding an optimal solution to the problem with an objective of maximizing the execution savings is equivalent to solving the MCKP. Thus, the TA_{min} is NP-hard. □

5.3 Related work

Recently, various DVFS and DPM policies have been proposed as solutions of dynamic thermal management (DTM). To avoid the thermal crisis in high performance processor, most of them have been addressed as a performance optimization problem under an emergency temperature limit. These work

can be classified into two categories: i) the online DTM techniques [12, 31, 50, 58, 91, 92, 94] and ii) the offline DTM techniques [55, 65, 79, 109].

Brooks and Martonosi [12] present a comprehensive summary and comparison of different DTM techniques aimed at general purpose processors. The techniques are reactive in nature and they throttle the processor activity by either reducing the frequency (or both voltage and frequency) or restricting the operation of a unit (decode throttling, I-cache toggling) and so on. They are invoked only when the temperature crosses a trigger temperature as in Huang's technique [31]. Skadron et al. [92] present a feedback control DVFS mechanism by adaptively varying the voltage to maintain the temperature under a thermal limit. In [91], they also provide a DTM technique by combining the instruction-level parallelism and DVFS technique to manage the temperature. McGowen et al. [58] address a DTM technique by tuning the voltages when a thermal monitor observes an over-power event. Srinivasan et al. [94] proposes a predictive DTM algorithm targeted at multimedia application. Lee et al. [50] present a similar DTM mechanism for MPEG decoding by observing the profiled temperature information to predict future thermal crisis risk. Recently, Intel has reported about the thermal aware design for some of their high performance processors [36]. Most of them control the fan speed when the temperature crosses a threshold. All of these approaches are online, reactive and heuristic techniques, and cannot achieve performance guarantee within a fixed ratio to the optimal.

Liu et al. [55] formulate the dynamic thermal management as a nonlinear programming problem. Bansal et al. [65] theoretically show that the power management techniques that are effective for energy saving may not be effective for managing temperature. Yuan et al. [109] present an optimal offline temperature-aware leakage minimization technique with a single active state voltage by a dynamic programming approach. Rao et al. [79] provide an off-line optimal speed profiling algorithm by the calculus of variation technique with continuous speed. They also present a two-speed approximation solution for the discrete voltage case. All of these are static offline DTM techniques. Most of them assume that the voltage/frequency is continuous and cannot deal with multiple discrete voltages.

Two fundamental problems of the practical dynamic thermal management have not been addressed yet. What is the tight upper bound of the performance under a thermal limit with discrete DVFS? How can we efficiently achieve a good schedule within a quality bound of the optimal? The solutions are still not clear. The answers to these problems would provide a good basis to evaluate the online techniques. Moreover, an efficient approximation algorithm with guaranteed quality bound would be applicable in practice.

In this work, we focus on these two essential problems. Our objective is to achieve maximum performance in terms of minimizing the execution time under thermal constraints. At first, we present an optimal offline DVFS technique with discrete voltage/frequency (v/f) settings for the dynamic thermal management problem. Next, we provide an $(1+\varepsilon)$ FPTAS to efficiently obtain a feasible solution within a quality bound of the optimal. To the best of our knowledge, our technique is the first one that generates both the optimal and approximation schedules for a sequence of tasks with discrete voltage/speed states under thermal constraints. Our technique targets the performance optimization problem with a periodic task set.

5.4 Algorithms

Optimal algorithm

The optimal algorithm is based on a dynamic programming (DP) approach that runs in pseudo-polynomial time similar to the knapsack problem [96] . However, TA_{min} is differentiated from the knapsack problem because of the non-linear thermal constraint. The central idea of the DP originates from the following property of the problem: given time Z to execute i jobs ($i \leq 2n+1$), lower the final temperature on completion of the i jobs, greater the possibility that the thermal constraint will not be violated on completion of the remaining jobs. Let $T(i,Z)$ be the minimum final temperature, when i jobs are executed in exactly Z time. In the DP algorithm, $T(i,Z)$ is minimized subject to T_{max} for $i \in \{1,2,3,\ldots,2n+1\}$ and $Z \in [1, Z_{UB}]$ where Z_{UB} is an upper bound on the optimal value of Z. Let Z^* denote the optimal value. Z^* is determined by the smallest value of Z such that $T(2n+1,Z) \leq T_o$.

Z_{UB} can be calculated by considering a schedule S_{init} as follows. Given an initial temperature T_o we first sleep such that $T = T_{amb}$. Then we execute the first job at the highest voltage (fastest time) that does not violate the temperature constraint T_{max}. Let T_1 denote the temperature at the end of execution of job J_1. Next we again sleep for some time such that the temperature reduces to T_{amb} from T_1. We then execute the second job at the highest voltage that does not violate the temperature constraint and again sleep till temperature is equal to T_{amb}. We repeat for all jobs. Clearly, such a schedule is a feasible schedule and therefore is a valid upper bound on Z^*.

Let $S_{i,Z}$ be the schedule with $T(i,Z)$. If $S_{i,Z}$ does not exist, define $T(i,Z) = \infty$. Set $T(0,Z) = T_o$ for $Z \in [1,...,Z_{UB}]$. Set $T(1,0) = T_o$, because the first one is a sleep job and can be zero sleep time. The recurrence relation for the DP algorithm is given by:

$$T(i,Z) = \min_{j \in [1,r]} \{T(i-1, Z-t_{ij}) + \Delta T(s_j) | T \leq T_{max}\} \tag{5.6}$$

The non-linear thermal equation (Equation 5.2) is utilized to achieve $\Delta T(s_j)$ in a particular sleep or active state. From the recurrence, we can find $T(N,Z)$, for all $Z \in [1, Z_{UB}]$. The optimal solution is then S_{N,Z^*} (denoted by S^* in the remainder of the work), where

$$Z^* = \min\{Z | T(N,Z) \leq T_o\} \tag{5.7}$$

The recurrence relation leads to an algorithm that constructs a 2-dimension DP table. The row represents the objective $Z \in [1, Z_{UB}]$, and the columns represent the $2n+1$ jobs. Each cell has an entry of the minimum final temperature when Z time is spent and i jobs are finished. Further, each cell also has an entry for the time t_{ij} associated with sleep or active state s_j that generates the minimum temperature value. The t_{ij} value will be essential for tracing back the final solution. The table is constructed in the order of row by row. Thus, after the algorithm enters the cell (i,Z), the cells for all the row indices smaller than Z are filled in. And, the previous $i-1$ cells in the Z_{th} row are also filled in. The algorithm need not re-calculate the optimal solution for a given subproblem. For each cell, r calculations are needed to find the minimum final temperature. Once the algorithm finds the Z^*, the optimal schedule, denoted by S^*, is achieved by tracing back in the solution table from (2n+1) to 1. This can be easily implemented by $2n+1$ table lookups.

The computation complexity of the DP algorithm is pseudo- polynomial. For each cell, it needs $O(r)$ computations. The algorithm has $O((2n+1) \cdot Z_{UB})$ iterations to fill in the cells. Thus, the computation complexity is $O(rn \cdot Z_{UB})$.

$(1+\varepsilon)$ *FPTAS for* TA_{min}

The DP algorithm for the optimal solution is not polynomial due to the factor Z_{UB} in the computation complexity which could be exponential in the size of the problem. We now develop a fully polynomial time approximation scheme (FPTAS) for TA_{min}. A FPTAS is an approximation algorithm whose run time complexity is bounded by a polynomial in the size of the problem and $(1/\varepsilon)$. A FPTAS is the best one can hope for a NP-hard optimization problem [96]. The proposed algorithm generates schedules whose execution time is guaranteed to be no more than $(1+\varepsilon)Z^*$ where ε (typically $0 < \varepsilon \leq 1$) is a designer specified quality bound.

Our approximation scheme parallels the FPTAS for the restricted shortest path problem [57, 103]. However, there are several key differences in the TA_{min} as opposed to the restricted shortest path due to the non-linear thermal constraints. The approximation algorithm works by scaling and reducing the search space for Z. It is described in Figure 5.2. The main algorithm is the TA_{min}-Approx(ε). Ini-

tially, the algorithm finds the search space $[Z_{LB}, Z_{UB}]$ for Z^*. As described earlier Z_{UB} can be calculated from S_{init}. Z_{LB} can also be estimated from S_{init} by summation of the execution time of the jobs in the active state. Let $t_{i,init}$ denote the execution time for a job J'_i (i is even) in the active state for the schedule S_{init}. Thus, $Z_{LB} = \sum_{J'_i \in J} t_{i,init}$. The algorithm then narrows down the search space by probing the scaled problem in lines 2 to 5. Here, probe(Z, ε) acts as a test procedure that returns success if the scaled problem has a feasible schedule, otherwise returns failure. The search procedure continues until the solution space is narrowed down to $[Z_{LB}, 6Z_{LB}]$. Finally, TAapprox(UB, LB, ε) is invoked that returns an $(1+\varepsilon)$ approximated result. In both probe and TAapprox, the $T'(N, Z') \leq T_o$ procedure is utilized, which is similar to the recurrence equation 8.12. The only difference to 8.12 is that the scaled values (Z' and t'_{ij}) are utilized when searching for $T(i-1, Z-t_{ij})$. However, non-scaled values of t_{ij} (that is, the original values of t_{ij}) are utilized for calculation of ΔT. Thus, the feasible solution for the scaled problem is also feasible for the non-scaled problem and vice versa, as the temperature calculation is made with the non-scaled time values. Next, we prove TA_{min}-Approx is an $(1+\varepsilon)$ FPTAS.

Let the $t_{min} = \min_{\forall J_i \in J, s_j \in M}\{t_{ij}\}$ denote the minimum execution time of any job in an active state. Let $\beta = t_{ms}/t_{min}$. In the schedule S_{init} let β_i denote the ratio between the sleep time preceding the active job J'_i ($J'_i \in J$) and the execution time of the job $t_{i,init}$. It is clear that $\beta \geq \beta_i$. Thus, we have

$$\begin{aligned} Z_{UB} &\leq t_{ms} + \sum_{J'_i \in J} (\beta_i + 1) t_{i,init} \\ &\leq (\beta+1) \sum_{J'_i \in J} t_{i,init} + \beta t_{min} \leq (2\beta+1) Z_{LB} \end{aligned} \tag{5.8}$$

The first inequality follows by modifying S_{init} such that the last sleep state is for time t_{ms}. The second inequality follows from $\beta \geq \beta_i$. The last inequality follows from $t_{min} \leq Z_{LB}$. Thus, initially $Z_{UB}/Z_{LB} \leq 2\beta + 1$ and $Z^* \in [Z_{LB}, Z_{UB}]$.

Lemma 5.4.1. *If probe(Z, ε) returns failure, $Z^* > Z$.*

Proof. Suppose $Z^* \leq Z$ and *probe*(Z, ε) returns failure.

$$Z'(S^*) = \sum_{S^*} \lfloor \frac{t_{ij}}{K} \rfloor \leq \sum_{S^*} \frac{t_{ij}}{K} \leq \frac{Z^*}{K} \leq \frac{Z}{K} \leq \lfloor \frac{Z}{K} \rfloor + N \tag{5.9}$$

$Z'(S^*)$ is the objective value for the scaled version of the problem with the optimal schedule S^* that executes in Z^*. Recall that a feasible schedule of the original problem is also a feasible schedule in the scaled problem. Since the upper bound of the search in probe(Z, ε) is $\lfloor \frac{Z}{K} \rfloor + N$, it would succeed with S^*. Thus, it is a contradiction. □

Lemma 5.4.2. *If probe(Z, ε) returns success, $Z^* \leq Z(1+2\varepsilon)$.*

TA_{min}-Approx(ε):

```
initially get Z_LB and Z_UB;
Z_UB = Z_UB/3;
while (Z_UB ≥ 2·Z_LB)
   { let Z = √(Z_LB · Z_UB);
   if probe(Z,1) = failure, Z_LB = Z;
   else Z_UB = Z; /* probe(Z,1) = success */}
Z_f = TAapprox(3·Z_UB, Z_LB, ε);
return Z_f;
```

probe(Z,ε):

```
set K = ε·Z/N; t'_ij = ⌊t_ij/K⌋; Z' = ⌊Z/K⌋ + N;
if T'(N,Z') ≤ T_o, return success;
else return failure;
```

TAapprox(UB,LB,ε):

```
set K = ε·LB/N; t'_ij = ⌈t_ij/K⌉; Z' = ⌈UB/K⌉ + N;
return Z_f = min{Z | T'(N,Z') ≤ T_o};
```

Figure 5.2: A FPTAS for TA_{min}

Proof. Because the probe succeeds, there is at least one feasible schedule S with the scaled problem such that

$$Z'(S) \leq \lfloor \frac{Z}{K} \rfloor + N \leq \frac{Z}{K} + N \tag{5.10}$$

Also,

$$Z'(S) = \sum_S \lfloor \frac{t_{ij}}{K} \rfloor \geq \sum_S \frac{t_{ij}}{K} - N \geq \frac{Z^*}{K} - N \tag{5.11}$$

The first inequality follows from $\lfloor \frac{t_{ij}}{K} \rfloor \geq \frac{t_{ij}}{K} - 1$. The second inequality follows from that Z^* is the optimal. The following inequality follows from Equations 8.14 and 8.15, and the definition of K:

$$Z^* - NK \leq Z + NK \Rightarrow Z^* \leq (1+2\varepsilon)Z \tag{5.12}$$

□

Lemma 5.4.3. *If $LB \leq Z^* \leq UB$, TAapprox(UB,LB,ε) succeeds and returns $Z_f \leq (1+\varepsilon)Z^*$.*

Proof. Since $Z^* \leq UB$ and $\lceil \frac{UB}{K} \rceil + N$ is the upper bound of the DP, the TAapprox(UB,LB,ε) would succeed. Let S be the optimal schedule in the scaled problem. Note that S^* is a feasible schedule in the scaled problem, because the search upper bound in TAapprox is bigger than Z^*. Then, we have

$$\begin{aligned} Z_f &= K\sum_S t'_{ij} \leq K\sum_{S^*} t'_{ij} \\ &\leq \sum_{S^*} t_{ij} + NK = Z^* + \varepsilon LB \leq Z^*(1+\varepsilon) \end{aligned} \tag{5.13}$$

The first inequality follows from the fact that optimal schedule S^* is a feasible solution for the scaled version of the problem, and the optimal schedule S in the scaled problem would achieve execution time no more than that with S^* . The second inequality follows from $Kt'_{ij} \leq t_{ij} + K$, when t_{ij} is rounded up. The third inequality follows from $LB \leq Z^*$. □

Lemma 5.4.4. *TA_{min}-Approx generates a $(1+\varepsilon)$ approximation schedule.*

Proof. By the Lemma 8.5.3, 8.5.4 and the algorithm, we know that, in the kth iteration of the while loop, we have

$$Z_{LB}^{[k]} \leq Z^* \leq 3 \cdot Z_{UB}^{[k]} \tag{5.14}$$

In the line 6 of TA_{min}-Approx, $Z_{UB} < 2 \cdot Z_{LB}$. In the input of TAapprox, $Z_{LB} \leq Z^* \leq 3 \cdot Z_{UB} < 6 \cdot Z_{LB}$. By the Lemma 8.5.5, the $TA_{min} - Approx$ is an $(1+\varepsilon)$ approximation schedule. □

Lemma 5.4.5. *The complexity of TA_{min}-Approx(ε) is $O(\frac{n^2 r}{\varepsilon} + n^2 r \log\log\beta)$.*

Proof. In the line 0 of the TA_{min}-Approx, the complexity is $O(nr)$. In the *probe*, the complexity is $O(\frac{n^2 r}{\varepsilon})$, because the Z is scaled by $K = \frac{\varepsilon Z}{N}$. In the line 6, the complexity is also $O(\frac{n^2 r}{\varepsilon})$ because $Z_{LB} \leq 3 \cdot Z_{UB} < 6 \cdot Z_{LB}$. Now the complexity from line 2 to line 5 is critical for the whole complexity.

In the $(k+1)^{th}$ iteration of the while loop, we always have $\frac{Z_{UB}^{[k+1]}}{Z_{LB}^{[k+1]}} = (\frac{Z_{UB}^{[k]}}{Z_{LB}^{[k]}})^{\frac{1}{2}}$. Recall that the while loop works only when $Z_{UB} \geq 2 \cdot Z_{LB}$. Let the number of iterations be p. We obtain an upper bound on p with the following equation:

$$\frac{Z_{UB}^{[p]}}{Z_{LB}^{[p]}} = (\frac{Z_{UB}^{[0]}}{Z_{LB}^{[0]}})^{(\frac{1}{2})^p} \geq 2 \tag{5.15}$$

As due to line 1 in TA_{min}-Approx(ε) we initially have $\frac{Z_{UB}^{[0]}}{Z_{LB}^{[0]}} = \frac{2\beta+1}{3}$, p is no more than $O(\log\log\beta)$. So, the complexity of line 2 to 5 is $O(n^2 r \log\log\beta)$. Thus, the overall complexity is $O(\frac{n^2 r}{\varepsilon} + n^2 r \log\log\beta)$, which is polynomial. □

Theorem 5.4.1. *The TA_{min}-Approx algorithm is a $(1+\varepsilon)$ FPTAS.*

Proof. The result directly follows from Lemma 8.5.6 and 8.5.7. □

5.5 Experimental results

Experiment Setup

We obtained the power consumption model from [79] which is based on the data of a 70nm CMOS processor from [41]. We choose 6 voltage levels ranging from 0.6V to 1.1V (0.1V per step). The associated

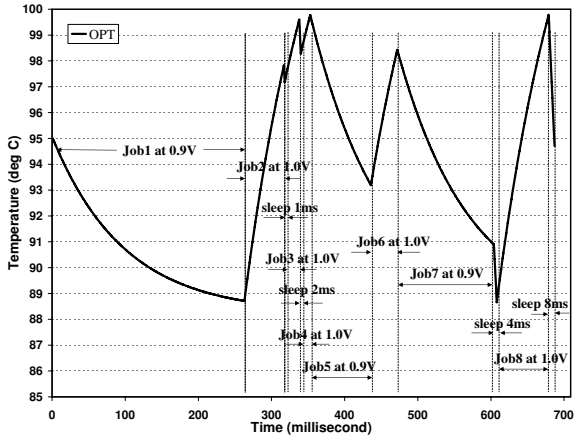


Figure 5.3: Thermal aware schedule

frequencies were between 0.78GHz and 3.8GHz. The thermal capacitance is chosen as $140.3J/°C$ from HotSpot [92]. The thermal resistance is dependent on the cooling technology and the package process. It is stated in [75] that the value is in the range of 0.3 to 1.5 with conventional air cooling. We set the thermal resistance as $0.7°C/W$. We set the maximum temperature constraint as $100°C$ corresponding to a typical thermal constraint on current day processor. The ambient temperature is set as $35°C$. Since $0.1°C$ rise/fall may take 10^5 cycles [92], the granularity of the time is set as milliseconds. The optimization techniques were coded in C++ and the experimentations were performed on a Pentium 4/ 2.4GHz/ 1GB WindowsXP PC.

Results for Multimedia Benchmarks

We combined four kinds of multimedia applications from MediaBench [59] to obtain a task set with 8 jobs: image compression (jpeg), speech compression (adpcm), encryption/decryption (pegwit) and video compression (mpeg2). Each category included encoder and decoder. We obtained the workload (worst case cycle numbers) of each job from SimpleScalar [89]. The workload of these jobs were in the range of $10^7 - 10^9$ cycles. The initial temperature of the processor was set as $T_o = 95°C$. We implemented schedules with both our thermal-aware optimal algorithm and energy-aware optimal algorithm from [100] with the same execution time. The energy-aware optimal technique in [100] performs an exhaustive search on the optimal energy savings for task sets operating at discrete v/f levels under a deadline constraint. We depict the thermal curves with both optimal schedules in Figure 1.1. We generated thermal aware schedules with our technique with quality bounds of 5% ($\varepsilon = 0.05$), 10% ($\varepsilon = 0.10$), 15% ($\varepsilon = 0.15$), 25% ($\varepsilon = 0.25$) and 50% ($\varepsilon = 0.50$). The thermal curves of the results are plotted in Figure 5.4.

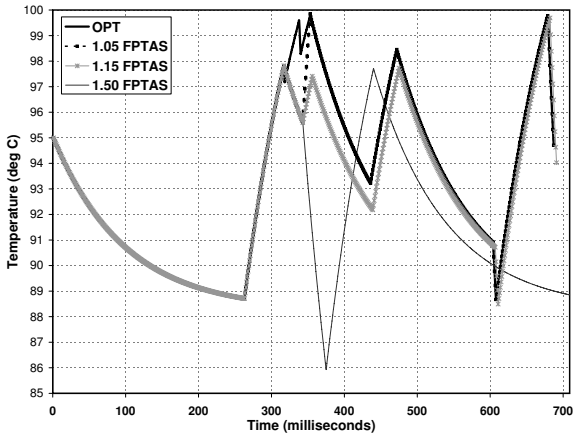


Figure 5.4: OPT vs $(1+\varepsilon)$ FPTASs

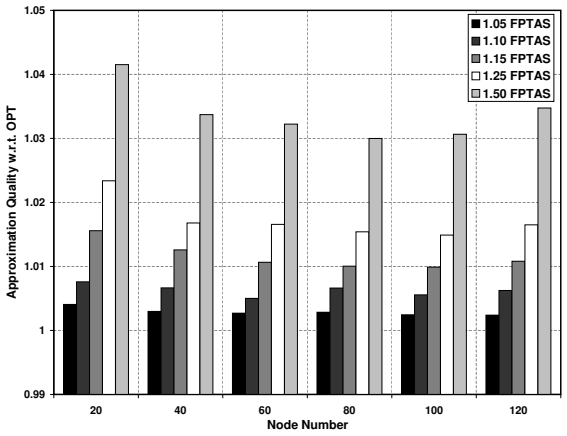


Figure 5.5: FPTAS: Real Approximation Ratio

Thermal-aware OPT vs Energy-aware OPT In Figure 1.1, we compared the thermal-aware OPT schedule generated by our technique with the energy-aware OPT schedule generated by the technique from [100] for 3 iterations ($period = 686ms$). The two schedules have the same period and execute identical jobs. The energy-aware OPT schedule executes all the jobs in the beginning and only goes to sleep toward the end of the iteration. Consequently, the energy optimal schedule generates thermal constraint violations (up to $106°C$). Figure 5.3 depicts the thermal-aware schedule with the job execution times and sleep times specified. Compared to the energy optimal schedule the thermal-aware schedule sleeps more frequently. These observations demonstrate that energy optimal schedules are unsuitable for satisfying the thermal constraints, and justify the need for addressing the thermal aware scheduling problem.

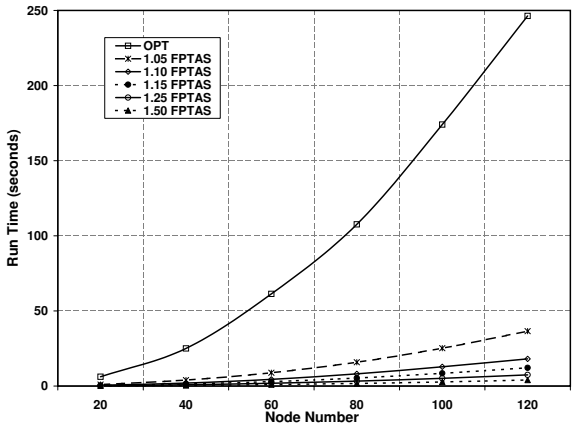


Figure 5.6: FPTAS: Run time vs N

Thermal-aware OPT vs FPTASs We compared the thermal curve and execution time of the optimal schedule with schedules for FPTAS with different values of $(1+\varepsilon)$ $(1.05, 1.15, 1.5)$. The thermal curves are shown in Figure 5.4. As can be seen from the deviation of the curves from the optimal increases as ε is increased. However, the overall execution time for the schedule is very close to the optimal. The actual approximation ratios of the schedules were found to be 1.004, 1.016 and 1.032 from bounds of 5%, 15% and 50%, respectively. In other words for the tested benchmark applications our technique is able to generate schedules within 3.2% of the optimal even with a quality constraint of 50%.

Summary The energy-aware optimal schedule cannot address the thermal aware scheduling problem. For the benchmark applications our techniques can generate very close to optimal results even with a quality bound of 50%.

Results for Synthetic Task Set

We evaluated our technique by experimenting with large synthetic task sets with up to 120 nodes. The number of jobs in each set were varied from 20 to 120 in steps of 20. At each task set number, we generated 10 sets of tasks. The workload of each job was uniform randomly generated, and varied in the range of $10^6 - 10^9$ cycles. Then, we calculated the execution time of each job at each active state by the processor model from [79]. The initial temperature was set at $65°C$. We evaluated the approximation quality of the results (Figure 5.5) and run times of our technique (Figure 5.6) for different values of ε.

Evaluation of the approximation quality of FPTASs Figure 5.5 illustrates the worst approximation ratio with respect to OPT for each node number from 20 to 120. The FPTAS with approximation bound from

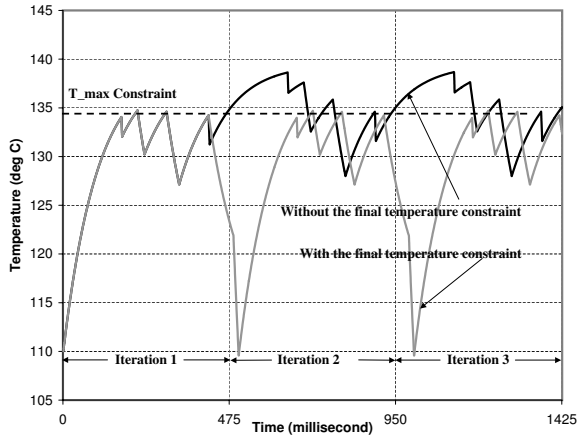


Figure 5.7: Effect of final temperature constraint

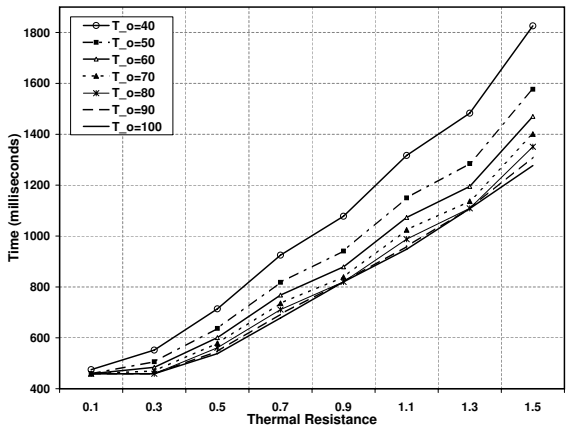


Figure 5.8: Effect of initial temperature and thermal resistance

5% to 25% matches the OPT, since the actual approximation ratios of those are no more than 1.025. Even with the 50% quality bound, the real approximation ratio is no more than 1.05 and the standard deviation of these ratios is no more than 0.007.

Evaluation of the run time of FPTASs Figure 5.6 depicts the average running times (in seconds) of the FPTAS with different values of ε. As expected, the run time of the OPT algorithm is the slowest, while the 1.50 FPTAS is the fastest. The runtime by the OPT algorithm is increasing much faster than the FPTAS. The figure also infers that the runtime by the OPT algorithm is exponential to the increase of the node number, while those of the FPTAS are near-linear. The run time of the FPTAS algorithm with 120 tasks for a 50% quality bound is under 5 seconds.

Summary The actual approximation ratio of the schedules generated by FPTAS for different values of ε were much better than the theoretical bounds. We can obtain a good trade-off between the design quality and technique execution time by varying ε.

Effect of final temperature constraint

As the jobs run in a periodic manner the final temperature of the schedule is the initial temperature of the next iteration. In our problem formulation, we set a constraint $T_f \leq T_o$. Figure 5.7 depicts two schedules with and without the final temperature constraint. As can be observed from the schedule, absence of the final temperature constraint causes thermal violations in subsequent iterations.

Effect of initial temperature and thermal resistance

We evaluated the effect of the initial temperature and thermal resistances on the performance of the job set. We varied the initial temperature from $40°C$ to $100°C$ ($10°C$ per step), and the thermal resistance from 0.1 to 1.5 with 0.2 per step. The results are plotted in Figure 5.8 for a temperature constraint of $100°C$. We observe that as the thermal resistance is reduced the performance of the schedule improves. This observation points towards incorporating better cooling techniques and may be integrating thermal aware scheduling with cooling operation. Sometimes increasing the initial temperature improves the performance. This is primarily due to the final temperature constraint. For example, if $T_o = 40°C$, the final temperature constraint limits the feasible schedule such that the final temperature is at most $40°C$, which takes a large amount of sleep time. If $T_o = 80°C$, the schedule can select some v/f levels such that the temperature is steady around $80°C$. Then the final temperature can be no more than $80°C$ with little sleep time. Therefore, the performance is increased. Although, it seems that an initial temperature $T_o = T_{max}$ generates the fastest schedule, it may not always be the case. Based on the job set, if $T_o = T_{max}$ the processor may require to sleep for a finite amount of time before it can begin executing the first job. Thus, an alternative value of T_o may generate a shorter schedule.

5.6 Conclusion

We introduced the thermal aware performance maximization problem. We justified the problem by demonstrating the inability of the energy optimal schedule to satisfy the temperature constraints. We defined problem and proved that it is NP-hard. We next presented the optimal algorithm and FPTAS for the problem. Experimental results demonstrate that FPTAS can generate very high quality results even with an approximation bound of 50%.

Chapter 6

Thermal aware scheduling for applications with uncertain execution times

The chapter addresses a stochastic version of the thermal-aware performance optimization problem on embedded processors. In the problem, the required execution cycles of each task follow a randomized distribution and hereby the execution time varies uncertainly. We give the definition and formulation of the stochastic problem, and then present optimal and approximated solution. The work is organized as follows: Section 6.1 presents the preliminaries of system model and defines several important parameters of the problem, Section 6.2 defines the problem, Section 6.3 discusses the previous work, Section 6.4 presents the optimal and a fully polynomial approximation scheme for the problem, Section 6.5 discusses the experimental results, and finally Section 6.6 concludes the work.

6.1 Preliminaries

Consider a processor equipped with a finite set of discrete v/f active states $\mathscr{M} = \{s_1, s_2, ...s_m\}$. Each state s_j has a voltage v_j and frequency f_j. The processor executes a job (say J_i) at a particular state s_j and consumes power ρ_{ij}. *Latency* is the time spent to execute a job. It is given by the ratio of the job cycle number w to the scheduled frequency f_j, and specified as $t = \frac{w}{f_j}$.

Similar to the thermal model in Chapter 5, the temperature of a processor is modeled by a lumped RC circuit due to the duality between heat transfer and electronic phenomena. The thermal resistance R and thermal capacitance C capture heat transfer phenomena and are specified as part of the model. Assume the power consumption of the processor is P during a time period t. The current processor temperature T after processor works for time t with power consumption P can be computed from the following equation.

$$T = P \cdot R + T_{amb} + (T_o - P \cdot R - T_{amb}) \cdot e^{-\frac{t}{RC}} \tag{6.1}$$

T_o is the initial temperature when $t = 0$. T_{amb} is the ambient temperature. The final temperature after the execution of a job J_i with cycle number w_i in state s_j is denoted by $T_f(i)$ and calculates as:

$$T_f(i) = T_o + (T_{ij}^s - T_o) \cdot (1 - e^{-\frac{w_i}{RCf_j}}) = T_o + \Delta T(s_j, w_i) \tag{6.2}$$

where T_{ij}^s is the steady temperature with power consumption ρ_{ij} and given by $T_{ij}^s = \rho_{ij}R + T_{amb}$. In this equation, $\Delta T(s_j, w_i)$ is the temperature change from T_o when processor executes the job with w_i cycles at state s_j.

In our system, we consider a sequence of independent jobs $\mathscr{J} = \{J_1, J_2, ...J_n\}$ to be executed on the processor. A DVFS algorithm is one that decides which active state should be chosen for the

execution of jobs. Each job J_i has some workload (w_i), specified as the number of CPU cycles to complete. Each w_i varies randomly and follows a probability distribution $\mathscr{P}_i$ with the best case cycle number BCC_i and the worst case cycle number WCC_i. We assume that the workloads of the various jobs are uncorrelated with each other. We determine at design time the unique active state for execution of each job. The resulting v/f schedule for $\mathscr{J}$ is denoted by $\mathscr{A}$.

The designer specifies a peak temperature limit T_m for the execution of $\mathscr{J}$. When CPU cycles of a job vary randomly, we can think of dynamic thermal management (DTM) as consisting of two DVFS mechanisms, the *non-violation (or off-line) mechanism* and the *violation (or on-line) mechanism*. The non-violation mechanism is the off-line v/f schedule $\mathscr{A}$, in the case that T_m won't be violated. The violation mechanism is on-line response, in the case that T_m is/might be violated. The on-line response mechanism is triggered once the scheduler determines that peak temperature is/might be violated. For example, the online scheduler may decide to abort the remaining jobs or execute the jobs in the run in lowest v/f state.

The work focuses on design time or off-line DVFS scheme as a system-level thermal management approach. In the case of the off-line scheme with uncertain job cycle time, the designer can specify the thermal constraint as – the probability that the schedule $\mathscr{A}$ will not violate T_m is no less than β ($\frac{1}{2} < \beta \leq 1$). For example if the designer specifies β as 0.80, the system should complete tasks based on $\mathscr{A}$ without violating the thermal constraint with a probability of 80%. We denote β as the ***survival*** probability. The actual statistical requirement on the system is that the survival probability for all the jobs is no less than β based on the schedule $\mathscr{A}$.

The optimization goal is to minimize the expected latency L when the statistical thermal constraint is satisfied.[1] The expected latency L is given by the summation of expected latency for each job based on $\mathscr{A}$. The expected latency for each job J_i depends on the expected cycle number of J_i and the scheduled execution state s_j.

6.2 Problem definition

The thermal aware performance optimization problem for the tasks with stochastic CPU demands, denoted as STA_{min}, can be described as follows.

[1]Considering the trade-off between latency and the survival probability under thermal constraint, the goal of some real time embedded systems could be maximizing the survival probability subject to an expected deadline constraint. This is a dual problem to STA_{min}.

- Given
- a processor with a set of active voltage/frequency(v/f) states $\mathscr{M} = \{s_1, s_2, ...s_m\}$, where $s_j =< v_j, f_j >$;
- a sequence of n independent jobs $\mathscr{J} = \{J_1, J_2, ...J_n\}$, each job J_i consumes power ρ_{ij} in s_j;
- for each J_i in $\mathscr{J}$, w_i is specified by a random distribution tuple $< \mathscr{P}_i, BCC_i, WCC_i >$;
- an initial temperature T_o and a peak temperature limit T_m;

The objective is to obtain a v/f schedule $\mathscr{A}$ such that the expected latency L for all the jobs is minimized when the survival probability of $\mathscr{A}$ is no less than a specified constant value β ($\frac{1}{2} < \beta \leq 1$).

Let y_i denote the event that the final temperature after executing J_i is no more than T_m. The probability of y_i depends on the final temperature after the execution of J_{i-1}, cycle number of current job J_i and the scheduled execution state. For a v/f schedule $\mathscr{A}$, the probability of y_i only depends on the first two factors. The final temperature after the execution of J_{i-1} is determined by the random cycle number of previous job J_{i-1}. Note that $T_f(i)$ represents the final temperature after completing the first i jobs. Let the peak temperature for schedule $\mathscr{A}$ be denoted by $T_p(\mathscr{A})$. Then, the survival probability can be formulated as:

$$\begin{aligned} Pr[T_p(\mathscr{A}) \leq T_m] &= Pr[y_1, y_2, \cdots, y_n] \\ &= Pr[y_1|T_o] \times Pr[y_2, \cdots, y_n|T_f(1)] \\ &= \prod_{i=1}^{n} Pr[y_i|T_f(i-1)] \end{aligned} \tag{6.3a}$$

The first equation follows from the thermal model and the selection of a single active v/f state for each job. Second equation is derived on the basis of conditional probability property. The last equation represents the survival probability of the $\mathscr{A}$ as the product of the conditional survival probability of every job.

The expected latency for each J_i executed at s_j, denoted as L_{ij}, is given by

$$L_{ij} = E[w_i]/f_j \tag{6.4}$$

$E[w_i]$ is the expected cycle number for J_i. It is the sum (for discrete probability distribution) or integration (for continuous probability distribution) of the probability of each possible w_i value multiplied by w_i. Thus, the expected latency L of the schedule $\mathscr{A}$ is the summation of expected latency of each job. Thus, STA_{min} can be formulated as follows.

$$\min L = \sum_{i=1}^{n}\sum_{j=1}^{m} L_{ij}x_{ij}$$

$$\prod_{i=1}^{n} Pr[y_i|T_f(i-1)] \geq \beta$$

$$\sum_{j=1}^{m} x_{ij} = 1, \forall J_i \in \mathscr{J}$$

$$T(0) = T_o, x_{ij} = \{0,1\}$$

$x_{ij} = 1$ denotes J_i is executed at s_j, otherwise 0. The objective is to find an optimal schedule $\mathscr{A}^*$ such that L is minimized and $Pr[T_p(\mathscr{A}) \leq T_m] \geq \beta$ based on $\mathscr{A}^*$. This problem involves nonlinear equations and stochastic random variables. Because the deterministic version of this problem is NP-hard [112], we have the following theorem.

Theorem 6.2.1. *The STA_{min} problem is at least NP-hard.*

In the following sections we present optimal and approximation algorithms as solutions for the problem.

6.3 Related work

There exists a considerable amount of work for micro-architecture level dynamic thermal management [31] [12] [92] [92] [58] [94] [50] for general purpose microprocessors. In the case of embedded systems researchers have proposed off-line techniques that exploit DVFS mechanisms to guarantee that peak temperature constraint is not violated [65] [55] [109] [79]. One only can achieve the optimal/approximated results with these techniques if the CPU clock cycle demands of applications do not vary. However, the application CPU demand in many embedded system does vary [56]. We demonstrate (in Section 6.5) that design with fixed clock cycle time (best case, average or even worst case) for a task as assumed by optimal existing technique [112] can cause thermal constraint violations in realistic scenarios. To the best of our knowledge, system-level stochastic thermal aware design problem has not yet been addressed.

In the past, researchers have addresses stochastic problems in the context of energy aware design. Stochastic energy aware design problem attempts to minimize expected energy for tasks with statistical CPU demands to meet the deadline constraint [56] [102] [29] [73]. However, these techniques cannot be utilized to solve our problem due to the non-linear behavior of the processor thermal model. Further, researchers have also addressed stochastic versions of classical theoretical problems such as knapsack [45] [24]. Although, existing approaches provide insight into problem formulation they cannot be utilized to solve the stochastic version of the thermal aware design problem.

To the best of our knowledge, this work is the first one that defines the stochastic version of the thermal-aware design problem and presents optimal and FPTAS algorithms as solutions.

6.4 Algorithms

At first, we present an optimal algorithm SO' for the STA_{min} problem with $\beta = 1$. Then an optimal algorithm SO for the problem with arbitrary β is proposed based on SO'. We further study the case that each job cycle demand is in normal distribution. We propose a FPTAS, named SA, for the STA_{min} problem instance where the cycle demand for each job follows a normal distribution.

Optimal algorithm

We first consider the STA_{min} problem with $\beta = 1$. Then we extend the solution for the extreme case to the general case with an arbitrary β. The main idea in our proposed optimal algorithms is: among the schedules for the first i jobs with the same expected latency and the same survival probability, lower the final temperature in the worst scenario, less time will be spent to complete the remaining jobs. We assume that all latency values are integral[2]. Let L_{ub} be the summation of execution time for each job with worst case cycle number at the lowest possible frequency. L_{ub} denotes the upper bound of the expected latency. We denote the optimal expected latency as L^*. Let L denote any possible value for L^* in $[1, L_{ub}]$.

Optimal schedule with $\beta = 1$

When $\beta = 1$, all $Pr[y_i|T_f(i-1)]$ values must be equal to 1. Thus, the stochastic thermal constraint becomes a hard constraint. This implies that a feasible v/f schedule should ensure that all the jobs survive in all scenarios. Given a schedule $\mathscr{A}$, the worst case scenario for peak temperature can be described as follows:

- if the execution of job J_i increases (or does not decrease) the processor temperature, J_i is executed with WCC_i cycles.
- if the execution of job J_i decreases the processor temperature, J_i is executed with BCC_i cycles.

Essentially, if the execution of job increases the peak temperature it is executed for the longest possible time (largest workload), and if the execution of a job reduces the peak temperature it is executed for the shortest possible time (smallest workload). Given an initial temperature T_o and $\mathscr{A}$, we can generate the thermal curve and determine the cycle number of each job based on the worst case scenarios described

[2] We can adjust the units of latency such that all the possible values are integral.

above. The jobs with non-decreasing (or increasing) thermal curve are denoted as *hot* jobs and the jobs with decreasing thermal curve are described as *cool* jobs. Note that the hot (or cool) job is defined by the worst case scenario for a particular schedule. It implies that the job is supposed to heat up (or cool down) temperature. In actual scenario, the job can heat up, cool down or maintain temperature.

The main observation is that the final temperature after the execution of each job in the worst case scenario is always the highest among all the other cases with any possible CPU cycle demands. Thus, the design based on the worst case scenario ensures that every job survives all scenarios if there exists a solution for the worst case scenario.

We present an optimal algorithm based on dynamic programming, named SO' algorithm, to solve this problem by minimizing the final temperature after finishing first i jobs in the worst case scenario. Our algorithm is differentiated from the optimal algorithm for the deterministic thermal aware design problem [112] by two critical steps – the definition of the worst case scenario, and the calculation of the final temperatures. In Section 6.5 we demonstrate that the solution techniques for the deterministic thermal aware design problem [112] cause temperature violations as they do not account for variable task execution times.

Let $\mathscr{A}_{i,L}$ denote a v/f schedule for the first i jobs whose expected latency is exactly equal to L. For each $J_i \in \mathscr{J}$ and each $L \in [1, L_{ub}]$, we define $T_f(i,L)$ to be the minimum final temperature in the worst scenario among all the $\mathscr{A}_{i,L}$ schedules subject to T_m. The final temperature after the execution of J_{i-1} is the initial temperature of J_i. $T_f(i,L)$ is initialized to T_o for $i = 0$ and $\forall L \in [1, L_{ub}]$. Thus, we have the following recursive relation in the dynamic program for determination of $T_f(i,L)$.

$$T_f(i,L) = \min_{\forall s_j \in \mathscr{M}} \{T_f(i-1, L-L_{ij}) + \Delta T(s_j, w_{il}) | T(i,L) \leq T_m\} \tag{6.6a}$$

Let $T_o^j(i) = T_f(i-1, L-L_{ij})$ be the initial temperature for J_i executing at s_j. Thus, if the steady state temperature for J_i in state s_j denoted by T_{ij}^s is no less than $T_o^j(i)$, J_i is classified as a hot job, alternatively it is a cool job. For hot jobs $w_{il} = WCC_i$ and for cool jobs $w_{il} = BCC_i$. Note that the notation of hot/cool jobs are all based on a v/f schedule, and initial temperature T_o for the first job. L_{ij} is calculated by Equation 6.4. $\Delta T(s_j, w_{ij})$ is the temperature change when J_i with w_{il} cycles starts at $T_o^j(i)$ and executes at s_j. It is calculated based on Equation 6.2.

The dynamic program calculates $T_f(i,L)$ for each i and L, and constructs a two dimensional state table shown in Table 6.1. The rows represent the jobs from J_1 to J_n and the columns represent possible L values in increasing order. The first row $i = 0$ all fills in T_o. In each cell (i,L), $T_f(i,L)$ is filled in and given by Equation 6.6. The SO' algorithm fills in cells column by column.

0	1	2	...	L	...	L_{ub}
0	T_o	T_o	...	T_o		T_o
J_1	$T_f(1,1)$	$T_f(1,2)$	...	$T_f(1,L)$		$T_f(1,L_{ub})$
...	...	...	...	...	...	...
J_i	$T_f(i,1)$	$T_f(i,2)$	...	$T_f(i,L)$		$T_f(i,L_{ub})$
...	...		...	...		...
J_n	$T_f(n,1)$	$T_f(n,2)$	...	$T_f(n,L)$		$T_f(n,L_{ub})$

Table 6.1: State table for SO' algorithm

Once the table is fully filled in, SO' algorithm can calculate the optimal L^* as.

$$L^* = \min_{\forall L}\{L|T(n,L) \leq T_m\} \tag{6.7}$$

Once L^* is found, the optimal schedule $\mathscr{A}^*$ can be obtained by a general backtracking step of dynamic programming. The overall computation complexity is equal to the complexity to construct Table 6.1, which is $O(nmL_{ub})$ and is pseudo-polynomial.

Optimal schedule with arbitrary β

We first describe a property for a modified version of the SO' algorithm, and then propose an optimal algorithm SO for the STA_{min} problem with arbitrary β ($\frac{1}{2} < \beta \leq 1$).

We assume the probability distribution of each job $\mathscr{P}_i$ is already discretized to a finite set $\mathscr{Q}_i$ with q_i CPU cycle numbers $\mathscr{Q}_i = \{w_{i1}, w_{i2}, ...w_{iq_i}\}$ in increasing order. w_{i1} is BCC_i and w_{iq_i} is WCC_i. For each w_{il} in $\mathscr{Q}_i$, a value p_{il} represents the probability that w_i is equal to w_{il}. And $\sum_{l=1}^{q_i} p_{il} = 1$. Figure 6.1 plots the probability distribution for cycle number of J_k as discrete bars. There are five possible cycle numbers for w_k. The probability of each cycle number is denoted as $p_1, ..., p_5$ as the height of the bars. Here $p_1 + p_2 + ... + p_5 = 1$ and $Pr[w_k = w_{k3}] = p_3$.

In the SO' algorithm we arbitrarily select a job J_k and for that job we utilize an arbitrary w_{kl} ($BCC_k \leq w_{kl} \leq WCC_k$) to calculate the final temperatures for row k. Notice that we do the modification for one and only one job. For all other jobs J_i we utilize BCC_i or WCC_i values as before. We get a new v/f schedule $\mathscr{A}_m$.

We define a variable α_k as follows (see Figure 6.1): if J_k is visualized as a hot job based on the v/f schedule, $\alpha_k = Pr[w_k \leq w_{kl}]$; otherwise, $\alpha_k = Pr[w_k \geq w_{kl}]$. For hot jobs, α_k is equal to the summation of the probability that w_k is no more than w_{kl} (in the figure, $\alpha_k = p_1 + p_2 + p_3$, if w_{k3} is chosen as w_{kl}); for cool jobs, α_k is equal to the summation of the probability that w_k is no less than w_{kl} (in the figure, $\alpha_k = p_3 + p_4 + p_5$, if w_{k3} is chosen as w_{kl})[3]. Then $\mathscr{A}_m$ has the following property.

[3]We also define α_k here for continuous distribution to be referenced in later section. For continuous distribution as represented

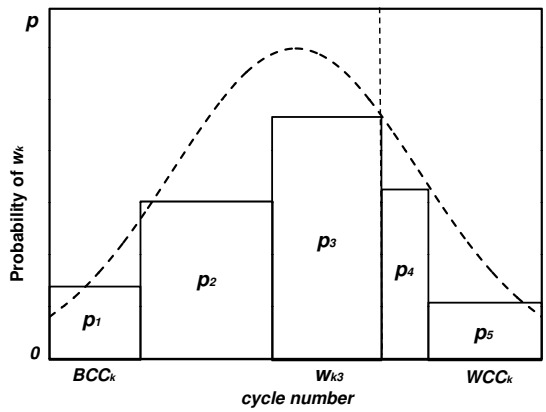


Figure 6.1: Definition of α_k

Lemma 6.4.1. *$\mathscr{A}_m$ might violate peak temperature T_m and α_k is a tight lower bound for survival probability based on $\mathscr{A}_m$.*

Proof. For the following two cases, we first show that $\mathscr{A}_m$ might violate T_m, then justify α_k is a tight lower bound.

- Case 1 – J_k is a cool job: J_k would satisfy T_m since the thermal curve is falling. In the generation of the schedule $\mathscr{A}_m$, $T_f(k)$ is calculated by w_{kl} ($\geq BCC_k$). Notice that for the worst case scenario we should have calculated $T_f(k)$ by considering its runtime on the basis of BCC_k. Now, when the job J_k is actually executed it may require BCC_k cycles. Thus, the temperature after execution of J_k would be greater than (or equal to) $T_f(k)$. Consequently, based on the schedule A_m (which was defined on the basis of $T_f(k)$), a hot job J_j ($j > k$) could violate T_m. However, if J_k executes with cycle number no less than w_{kl}, all the jobs are guaranteed not to violate T_m. This is because for all other jobs we consider the worst case scenario as described in algorithm SO'. Therefore, $Pr[T_p(\mathscr{A}) \leq T_m]$ is no less than the probability that the cycle demand of J_k is equal to or more than w_{kl}, which is $\alpha_k = Pr[w_k \geq w_{kl}]$.

- Case 2 – J_k is a hot job: As $T_f(k)$ is calculated by w_{kl}, the execution of job J_k may violate T_m if its actual cycle demand is greater than w_{kl}. Even if J_k does not violate T_m, there could exist a hot job J_j ($j > k$) that violates T_m because final temperature after execution of J_k might be greater than $T_f(k)$ (which was utilized to generate $\mathscr{A}_m$). However, if J_k executes with a cycle number no more than w_{ik}, all the jobs will satisfy T_m for sure. Thus, the survival probability $Pr[T_p(\mathscr{A}) \leq T_m]$ is no less than the probability that cycle demand of J_k is equal to or less than w_{ik}, which is $\alpha_k = Pr[w_k \leq w_{kl}]$.

by the dashed curve in Figure 6.1, if the cycle number on the dashed line is w_{k3} and w_{k3} is chosen as w_{kl}, α_k is the area (integration) under the curve to the left of the straight line for hot jobs, or to the right of w_{k3} for cool jobs.

□

From Lemma 6.4.1, we define the physical meaning of α_k as the survival probability due to J_k, because $T_f(k)$ is calculated by w_{kl}. Now, we modify the SO' algorithm further. We arbitrarily choose a w_{il} to calculate the final temperature of Row i for each (and every) J_i in Table 6.1. Thus, a schedule $\mathscr{A}'$ is generated. For each J_i, if J_i is a hot job based on $\mathscr{A}'$, $\alpha_i = Pr[w_i \leq w_{il}]$, else $\alpha_i = Pr[w_i \geq w_{il}]$. We define the following property for $\mathscr{A}'$.

Lemma 6.4.2. *The survival probability for $\mathscr{A}'$ is no less than $\prod_{i=1}^{n} \alpha_i$.*

Proof. Consider the situation that when the jobs are executing, each hot job J_i executes with cycle number w_i no more than w_{il} and for each cool job J_j executes with cycle number w_j no less than w_{jl} ($i \neq j \in \{1,2,...,n\}$). In this case, all the jobs based on $\mathscr{A}'$ do not violate T_m for sure. The probability of this situation occurring is $\prod_{i=1}^{n} \alpha_i$. Therefore, the lower bound of survival probability is $\prod_{i=1}^{n} \alpha_i$. By Lemma 6.4.1, it is a tight lower bound. □

The statistical thermal constraint can be formulated as follows because of Lemma 6.4.2.

$$\prod_{i=1}^{n} \alpha_i \geq \beta \tag{6.8}$$

The objective of the STA_{min} problem is then to achieve a schedule $\mathscr{S} =< \mathscr{A}, \mathscr{B} >$ such that the expected latency is minimized and Equation 6.8 is satisfied. $\mathscr{A}$ is a v/f schedule and $\mathscr{B} =< \alpha_1, \alpha_2, ..., \alpha_n >$ is the survival probability associated with each job J_i. Note that for each α_i there exists an associated w_{il} for J_i which is utilized to calculate $T_f(i)$.

We then present an optimal algorithm SO based on dynamic programming with three dimension state table – rows represent jobs from J_1 to J_n, columns represent the expected latency L from 1 to L_{ub}, and the third dimension represents the survival probability α due to the first i jobs. Each cell (i, L, α) in the state table contains $T_f(i, L, \alpha)$ which is the minimum final temperature when the first i jobs are finished with expected time equal to L and the survival probability due to the first i jobs is α.

Notice that in the dynamic programming table the L dimension represents all possible values from 1 to L_{ub}. Similarly, we require all possible values of α. We include all possible values of α in a set $\mathscr{D}$. $\mathscr{D}$ is constructed from the set of possible survival probability values associated with each job J_i denoted by $\mathscr{D}_i$. $\mathscr{D}_i$ can be obtained by calculating $Pr[w_i \leq w_{il}]$ and $Pr[w_i \geq w_{il}], \forall w_{il} \in \mathscr{Q}_i$. $|D_i| = 2 \times q_i, q_i = |\mathscr{Q}_i|$. We prune the D_i set to only include values in the range $[\beta, 1]$. Now, $\mathscr{D}$ can be constructed

by considering all possible product combinations of α_i, $\mathscr{D} = \bigcup \prod_{i \in = \{1,\ldots,n\}} \alpha_i, \forall \alpha_i \in \mathscr{D}_i$. $|\mathscr{D}| = q^n, q = \max(q_1, \ldots, q_n)$. We sort $\mathscr{D}$ in the ascending order to construct the third dimension of the dynamic programming table.

In the dynamic program, $T_f(i, L, \alpha)$ can be calculated by the following recursive relation:

$$T_f(i, L, \alpha) = \min_{\forall s_j \in \mathscr{M}; \forall \alpha_i \in \mathscr{D}_i} \{T_f(i-1) + \Delta T(s_j, w_{il}) | T_f(i, L, \alpha) \leq T_m\} \tag{6.9a}$$

$$T_f(i-1) = T_f(i-1, L - L_{ij}, \frac{\alpha}{\alpha_i}) \tag{6.9b}$$

When $T_{ij}^s \geq T_f(i-1)$, J_i is classified as a hot job and we can find a w_{il} for each α_i, where $\alpha_i = Pr[w \leq w_{il}]$; otherwise J_i is a cool job and $\alpha_i = Pr[w \geq w_{il}]$. Thus, the optimal L^* is achieved as the smallest L when a feasible $T_f(n, L, \beta')$ is obtained. Here β' is the smallest value in $\mathscr{D}$ and $\beta' \geq \beta$. Thus,

$$L^* = \min_{\forall L} \{L | T_f(n, L, \beta') \leq T_m\} \tag{6.10}$$

Each feasible schedule includes a v/f schedule $\mathscr{A}$ and a survival probability assignment $\mathscr{B}$ for all the jobs. With the L^*, we can back track and get the optimal schedule $\mathscr{S}^* =< \mathscr{A}^*, \mathscr{B}^* >$.

The computational complexity of SO is $O(nmqL_{ub}q^n)$ which is exponential in n. In the following section we present a $(1+\varepsilon)$ FPTAS algorithm for the STA_{min} problem when the cycle demand for each job follows a normal distribution. The discretization scheme for survival probability in the FPTAS algorithm can be utilized to reduce the complexity of SO to pseudo-polynomial.

Approximation algorithm

We present a $(1+\varepsilon)$ FPTAS, named SA, for the STA_{min} problem when the CPU demand w_i of each J_i follows a continuous normal distribution $\mathscr{P}_i$. Given a quality bound ε $(0 < \varepsilon \leq 1)$ and a peak temperature relaxation bound μ $(0 < \mu < 1)$, the FPTAS generates a schedule $\mathscr{S}^+ =< \mathscr{A}^+, \mathscr{B}^+ >$ which can achieve $(1+\varepsilon)$ times the optimal L^* under an $(1+\mu)$ relaxation of peak temperature limit T_m. The FPTAS discretizes the sets $[\beta, 1]$ and $[1, L_{ub}]$ in a manner that gives us a solution with expected latency no more than $(1+\varepsilon)L^*$ in polynomial time, at the expense of a slight loss in terms of feasibility - the solution schedule may overrun the peak temperature limit T_m up to $T_m + \mu(T_m - T_{amb})$.

Let $f_i(\cdot)$ $(g_i(\cdot))$ represent the survival probability distribution function for job J_i assuming that it is a hot (cold) job. We state:

Lemma 6.4.3. *$f_i(\cdot)$ and $g_i(\cdot)$ are continuous functions of w_i for hot and cold jobs, respectively. For hot jobs, $f_i(\cdot)$ is concave and monotonically increasing with w_i when $\beta > \frac{1}{2}$. For cool jobs, $g_i(\cdot)$ is concave and monotonically decreasing with w_i when $\beta > \frac{1}{2}$.*

$SA(\mu, \delta, T_m)$:

1 $l = 1$, $r = g = \lceil \lg L_{ub} \rceil$;
2 choose γ based on the definition;
3 $\mathscr{D}' = \{1, (1+\gamma)^{-1}, (1+\gamma)^{-2}, ..., (1+\gamma)^{-h}\}$;
4 binary search smallest 2^b s.t. $test(2^b, \mu, \delta, T_m)$ returns success;
5 $\mathscr{S}^+ = test(2^b, \mu, \delta, T_m)$;
6 return $\mathscr{S}^+$;

$test(L, \mu, \delta, T_m)$:

1 $K = \frac{\delta L}{n}$, $L' = \lceil \frac{L}{K} \rceil + n$, $L'_{ij} = \lceil \frac{L_{ij}}{K} \rceil$, $\forall \alpha, \alpha_i \in \mathscr{D}'$;
2 $\mathscr{S}^+ = SO_m(L', T_m + \mu(T_m - T_{amb}), \beta)$;
3 if $(\mathscr{S}^+ != null)$ return $\mathscr{S}^+$ and *success*;
4 else return $failure$; endif;

Figure 6.2: FPTAS for STA_{min} problem

The lemma follows trivially from normal distribution of w_i, and definitions of survival probability for hot and cold jobs.

The approximation scheme is described in Figure 6.2. It executes in polynomial time by scaling and reducing search spaces for possible L and α values. Given two variables μ, δ ($\delta = \varepsilon/2$) specified by the designer, the main algorithm is $SA(\mu, \delta, T_m)$. The algorithm conducts a binary search over the space $\mathscr{L} = \{1, 2, 2^2, ..., 2^g\}$ ($g = \lceil \lg L_{ub} \rceil$), until the smallest $L = 2^b$ is found such that a test procedure returns success. After 2^b is found, the solution $\mathscr{S}^+$ is generated by the test procedure on 2^b with $(1+\varepsilon)$ approximation.

The function $test(L, \mu, \delta, T_m)$ returns success if there exists a solution on the testing value L, otherwise returns failure. In $test()$ a modified SO algorithm, named SO_m, is invoked which is similar to SO algorithm. It uses dynamic programming on the scaled spaces for possible L and α values in Equation 6.9. L and L_{ij} are scaled to L' and L'_{ij} by a scaling factor $K = \frac{\delta L}{n}$ and rounded up. L' is the upper bound in state table as L_{ub} in SO algorithm.

The discretization of α values is more involved. Let $\mu\prime = \frac{\mu}{n}$. For each job J_i we define:

$$w_h(i) = WCC_i(1+\mu\prime)^{-1} \tag{6.11a}$$

$$w_c(i) = BCC_i(1+\mu\prime) \tag{6.11b}$$

$$\gamma_i = \min(\frac{1}{f_i(w_h(i))} - 1, \frac{1}{g_i(w_c(i))} - 1) \tag{6.11c}$$

We define $\gamma \leq \min_{\forall i}(\gamma_i)$. Possible α values lie in a finite set $\mathscr{D}'$ which is obtained by discretization of the range $[\beta, 1]$ as follows $\mathscr{D}' = \{1, (1+\gamma)^{-1}, (1+\gamma)^{-2}, ..., (1+\gamma)^{-h}\}$. Here $(1+\gamma)^{-h}$ is the smallest

value no less than β. Possible α_i values for each J_i are discretized similarly and belong to $\mathscr{D}'$.

Lemma 6.4.4. *For each α_i in $[\beta, 1]$ with associated w_i of a hot or cool job J_i, $f_i^{-1}((1+\gamma)\alpha_i) \leq (1+\mu\prime)w_i$ if J_i is a hot job; $g_i^{-1}((1+\gamma)\alpha_i) \geq \frac{w_i}{1+\mu\prime}$ if J_i is a cool job.*

Proof. When $\alpha_i > f_i(w_h(i))$ for hot job (or $\alpha_i > g_i(w_c(i))$ for cool job), the lemma is clearly true because $w_i(1+\mu\prime) = WCC_i$ (or $w_i(1+\mu\prime)^{-1} = BCC_i$). For the case $\alpha_i \leq f_i(w_h(i))$ for hot job (or $\alpha_i \leq g_i(w_c(i))$ for cool job), we have

- Case 1 - hot job: Since $\gamma \leq \frac{1}{f_i(w_h(i))} - 1$, $\gamma \leq \frac{f_i((1+\mu\prime)w_i)}{\alpha_i} - 1$ as $\frac{1}{f_i(w_h(i))} \leq \frac{f_i((1+\mu\prime)w_i)}{\alpha_i}$ by Lemma 6.4.3. Thus, we have $(1+\gamma)\alpha_i \leq f_i((1+\mu\prime)w_i)$. Again by Lemma 6.4.3 we have $f_i^{-1}((1+\gamma)\alpha_i) \leq (1+\mu\prime)w_i$.
- Case 2 - cool job: Since $\gamma \leq \frac{1}{g_i(w_c(i))} - 1$, $\gamma \leq \frac{g_i(w_i(1+\mu\prime)^{-1})}{\alpha_i} - 1$ as $\frac{1}{g_i(w_c(i))} \leq \frac{g_i(w_i(1+\mu\prime)^{-1})}{\alpha_i}$ by Lemma 6.4.3. Thus, we have $(1+\gamma)\alpha_i \leq g_i(w_i(1+\mu\prime)^{-1})$. Again by Lemma 6.4.3 we have $g_i^{-1}((1+\gamma)\alpha_i) \geq \frac{w_i}{1+\mu\prime}$.

Thus, it is proved. □

The modified SO_m algorithm takes $T_m + \mu(T_m - T_{amb})$ and β as parameters. This specifies that the feasible solution for the scaled problem satisfies $Pr[T_p \leq T_m + \mu(T_m - T_{amb})] \geq \beta$. We prove that SA is an $(1+\varepsilon)$ FPTAS. Let the optimal schedule be $\mathscr{S}^* =< \mathscr{A}^*, \mathscr{B}^* >$ for the STA_{min} problem. Round each α_i^* in $\mathscr{B}^*$ up to the smallest value α_i^u in $\mathscr{D}'$ no less than α_i^*. Denote the new schedule as $\mathscr{S}^u =< \mathscr{A}^*, \mathscr{B}^u >$.

Lemma 6.4.5. *In the STA_{min} problem with non-scaled L and L_{ij} and $\forall \alpha, \alpha_i \in \mathscr{D}'$, S^u is a feasible solution.*

Proof. We first prove that in the worst case scenario with $\mathscr{B}^u$, the peak temperature with $\mathscr{A}^*$ is no more than $T_m + \mu(T_m - T_{amb})$. Then we show that the survival probability based on $\mathscr{B}^u$ is no less than β.

We consider two possible cases based on the thermal classification of the job J_i. We observe $\alpha_i^* \leq \alpha_i^u \leq \min((1+\gamma)\alpha_i^*, 1)$. Recall that $\mu\prime = \frac{\mu}{n}$. We then prove the statement $T_f^*(i) \leq T_f^u(i) \leq T_f^*(i) + i\mu\prime(T_m - T_{amb})$ by induction.

We first show the statement is true for the base case $i = 1$.

- Case 1 – hot job: From Lemmas 6.4.3 and 6.4.4 we have $f^{-1}(\alpha_i^*) \leq f^{-1}(\alpha_i^u) \leq f^{-1}(\min((1+\gamma)\alpha_i^*, 1)) \Rightarrow w_i^* \leq w_i^u \leq \min((1+\mu\prime)w_i^*, WCC_i)$. Thus, for a single job executed in state s_j starting from T_o, we have $\Delta T_{ij}^* \leq \Delta T_{ij}^u \leq (1+\mu\prime)\Delta T_{ij}^*$, where all the ΔT are non-negative. Therefore, $T_f^*(1) \leq T_f^u(1) \leq T_f^*(1) + \mu\prime(T_m - T_{amb})$ holds true.

- Case 2 – cool job: From Lemmas 6.4.3 and 6.4.4 we have $g^{-1}(\min((1+\gamma)\alpha_i^*, 1)) \leq g^{-1}(\alpha_i^u) \leq g^{-1}(\alpha_i^*) \Rightarrow \max(w_i^*(1+\mu\prime)^{-1}, BCC_i) \leq w_i^u \leq w_i^*$. Therefore, for a single job executed at s_j starting from T_o, we have $\Delta T_{ij}^* \leq \Delta T_{ij}^u \leq \frac{\Delta T_{ij}^*}{(1+\mu\prime)} \leq (1-\mu\prime)\Delta T_{ij}^*$, where all the ΔT are negative. Therefore, $T_f^*(1) \leq T_f^u(1) \leq T_f^*(1) + \mu\prime(T_m - T_{amb})$ holds true.

The last inequality in both cases is true because every $|\Delta T_{ij}^*|$ with S^* is no more than $(T_m - T_{amb})$.

Then we show that, if the statement is true when $i = k-1$, it is also true when $i = k$.

J_k is executed starting from $T_f^*(k-1)$ in S^* and from $T_f^u(k-1)$ in S^u. In S^* and S^u, J_k is executed in the same v/f state, say s_j.

- Case 1 – hot job: Similar to $i = 1$ case, we have $w_k^* \leq w_k^u \leq \min((1+\mu\prime)w_k^*, WCC_k)$. Because $w_k^* \leq w_k^u$ and $T_f^*(k-1) \leq T_f^u(k-1)$, we get $T_f^*(k) \leq T_f^u(k)$. On the other hand, because $w_k^u \leq (1+\mu\prime)w_k^*$ and $T_f^*(k-1) \leq T_f^u(k-1)$, we get $\Delta T_{kj}^u \leq (1+\mu\prime)\Delta T_{kj}^*$ by the definition of $\Delta T(s_j, w_i)$ and the base case $i = 1$. Note that ΔT_{kj}^* is non-negative. Therefore,

$$\begin{aligned} T_f^u(k) &= T_f^u(k-1) + \Delta T_{kj}^u \\ &\leq T_f^*(k-1) + (k-1)\mu\prime(T_m - T_{amb}) + (1+\mu\prime)\Delta T_{kj}^* \\ &\leq T_f^*(k) + k\mu\prime(T_m - T_{amb}) \end{aligned}$$

  The first equation follows from the thermal model. The second equation follows from the statement with $i = k-1$ case and the inequality $\Delta T_{kj}^u \leq (1+\mu\prime)\Delta T_{kj}^*$. The third equation follows from thermal model and $|\Delta T_{kj}^*| \leq T_m - T_{amb}$.

- Case 2 – cool job: Similar to $i = 1$ case, we have $\max(w_k^*(1+\mu\prime)^{-1}, BCC_k) \leq w_k^u \leq w_k^*$. Because $w_k^* \geq w_k^u$ and $T_f^*(k-1) \leq T_f^u(k-1)$, we get $T_f^*(k) \leq T_f^u(k)$. On the other hand, because $w_k^u \geq (1+\mu\prime)^{-1}w_k^*$ and $T_f^*(k-1) \leq T_f^u(k-1)$, we get $\Delta T_{kj}^u \leq (1-\mu\prime)\Delta T_{kj}^*$ by the definition of $\Delta T(s_j, w_i)$ and the base case $i = 1$. Note that ΔT_{kj}^* is negative. With similar proofs as case 1, we have

$$\begin{aligned} T_f^u(k) &= T_f^u(k-1) + \Delta T_{kj}^u \\ &\leq T_f^*(k-1) + (k-1)\mu\prime(T_m - T_{amb}) + (1-\mu\prime)\Delta T_{kj}^* \\ &\leq T_f^*(k) + k\mu\prime(T_m - T_{amb}) \end{aligned}$$

Thus, we have proved that, for each J_i in $\mathscr{J}$ with v/f schedule $\mathscr{A}^*$, we can always get the final temperature relationship: $T_f^*(i) \leq T_f^u(i) \leq T_f^*(i) + i\mu\prime(T_m - T_{amb})$. Therefore the peak temperature T_p relationship for n jobs is that $T_p^* \leq T_p^u \leq T_p^* + n\mu\prime(T_m - T_{amb})$. Since $T_p^* \leq T_m$ for $\mathscr{A}^*$ and $\mu\prime = \frac{\mu}{n}$, we can get $T_p^u \leq T_m + \mu(T_m - T_{amb})$. Thus, $\mathscr{B}^u$ is feasible for peak temperature limit $T_m + \mu(T_m - T_{amb})$.

Now we show that the survival probability is no less than β with $\mathscr{S}^u$. Since $\mathscr{S}^*$ is feasible with β, $\prod_{i=1}^n \alpha_i^* \geq \beta$. For $\mathscr{S}^u$, we have the following equation due to the relation between α_i^* and α_i^u

$$\prod_{i=1}^{n} \alpha_i^u \geq \prod_{i=1}^{n} \alpha_i^* \geq \beta$$

Thus, it is proved. □

Let $L^\#$ be the optimal for the STA_{min} problem with non-scaled L and L_{ij}. Note that in this case all the possible α and α_i lie in the discretized $\mathscr{D}'$. The peak temperature limit is set to be $T_m + \mu(T_m - T_{amb})$. The associated optimal schedule is denoted as $\mathscr{S}^\# =< \mathscr{A}^\#, \mathscr{B}^\# >$. We have:

Lemma 6.4.6. $L^\# \leq L^*$.

Proof. By lemma 6.4.5, we can achieve a feasible schedule $\mathscr{S}^u$ from $\mathscr{S}^*$ with peak temperature limit $T_m + \mu(T_m - T_{amb})$ and survival probability β. Then we have

$$L^\# \leq L^u = L^*$$

The first step is true because the $\mathscr{S}^\#$ is the optimal schedule for the problem with scaled α and non-scaled L. The second step follows that S^u has the same v/f schedule as S^*. □

Lemma 6.4.7. *L^+ achieved by $SA(\mu, \delta, T_m)$ is no more than $(1+2\delta)$ times L^*.*

Proof. Because of the scaling and rounding of L, the expected latency L^+ achieved by S^+ is no more than $(1+2\delta)$ times $L^\#$. The proofs parallel to those in Chapter 2 and are omitted here. Thus, we have $L^+ \leq L^\#(1+2\delta)$. By Lemma 6.4.6, $L^+ \leq L^*(1+2\delta)$. □

The computational complexity of the $test()$ function is $O(\frac{n^2mh}{\delta})$. By the definition of $\mathscr{D}'$, $h \leq \frac{-\lg\beta}{\lg(1+\gamma)}$. The binary search in SA invokes $test$ procedure $lglg(L_{ub})$ times. Thus, the computational complexity of SA is fully polynomial and equal to $O(\frac{n^2mh}{\delta}\lg\lg L_{ub})$. Note that the peak temperature limit with L^+ is no more than $T_m + \mu(T_m - T_{amb})$ because the input for SO_m is the relaxed value. We have the following theorem when $\varepsilon = 2\delta$.

Theorem 6.4.1. *$SA(\mu, \delta, T_m)$ is an $(1+\varepsilon)$ FPTAS when peak temperature T_m is relaxed to $T_m + \mu(T_m - T_{amb})$.*

6.5 Experimental results

Experiment Setup

Processor thermal model : We consider the 70nm CMOS processor model from [79] with 6 voltage/frequency levels from 0.6V/0.78GHz to 1.1V/3.8G Hz (0.1V per step). The thermal capacitance and resistance settings are chosen as those in [92]. We set $35°C$ as ambient temperature, $65°C$ as initial temperature and $100°C$ as peak temperature limit.

Applications : We evaluate the proposed techniques by experimentations with both multimedia applications and synthetic task sequences. The multimedia task sequence includes four kinds of multimedia encoder/decoders from MediaBench [59]: image compression (jpeg), speech compression (adpcm), encryption/ decryption (pegwit) and video compression (mpeg2). The synthetic task sequences include 10, 15, 20 nodes with the WCC in normal, piosson or equal distribution. Each task has different WCC.

Discrete distribution of cycle numbers : We assign each task three discrete CPU cycle numbers (0.01WCC, 0.3WCC, WCC) with probabilities (0.03, 0.85, 0.12), respectively. We utilize the discrete distributions to evaluate *SO* with respect to SO'.

Continuous distribution of cycle numbers : The cycle number for each task is generated by normal distribution in the range of [0.01·WCC, WCC] with mean as 0.505·WCC and deviation as 5000. We utilize the continuous distributions to evaluate *SA* with respect to SO'.

Evaluation of SO with respect to SO' : We set the β of *SO* at 0.8. SO' generates the solution with $\beta = 1$.

Evaluation of SA with respect to SO' : We set the survival probability of of *SA* at 0.8. To evaluate *SA* techniques, we still use $T_m = 100°C$ as the input of SO_m in Figure 6.2 to ensure the solution is feasible. μ is set as 0.02. Theoretically, the expected latency by solutions from *SA* is no more than $(1+\varepsilon)$ times the optimal expected latency with peak temperature limit no less than $98.7°C$. For each task set we generate solutions by varying the quality bound on *SA* as $\varepsilon = 0.05, 0.15, 0.25$.

Simulations : For each task set, we simulate the execution of applications with solutions generated by SO' for discrete distributions, SO' for continuous distributions, *SO* and *SA*. In order to evaluate average latency and actual survival probability, 10,000 iterations are tested with the schedules achieved by the techniques for each task sequence. We record the average latency over those iterations without thermal violation and obtain the survival probability over the 10,000 iterations.

Platform : The techniques were coded in C and experimentations were performed on a Pentium 4/2.4GHz/1GB Windows XP PC.

Limitation of existing deterministic technique

We generate a v/f schedule for the multimedia job sequence by utilizing the optimal technique in Chapter 5 for the deterministic version of the system-level thermal aware design problem. We assume that the designer only considers the worst case cycle number w_i for each task to generate schedule. The expected thermal curve with fixed cycle number for each job is shown in Figure 6.3. The expected thermal curve satisfies thermal constraint. We next consider that in actual case the cycle number of the jobs follows a discrete distribution in the range $[0.01 \cdot w_i, w_i]$ with $0.1 \cdot w_i$ as the average. Figure 6.4 plots the observed thermal curve when most (5 out of 8) of the jobs execute with their average cycle number $(0.1 \cdot w_i)$. As we can see, the peak temperature constraint is violated most of time in actual case when the optimal schedule by the deterministic algorithm is applied. Thus, we demonstrate that the optimal thermal aware design technique for the deterministic version of the problem [112] cannot address the stochastic version.

Performance improvement and survival probabilities

We compare the improvement in expected latency by comparing our *SO* and *SA* techniques with SO'. Performance improvement is defined as the ratio of the expected latency achieved by SO' divided by the expected latency achieved by *SO* or *SA*. Table 6.2 shows the performance improvement (*P* improve) with 10 task sets. The performance improvements were calculated by simulations over 10,000 iterations. The *SO* approach provides average performance improvement as 1.12 comparing to SO' when survival probability of *SO* is set as 80%. With poisson-20 job sequence, it can even improve to 1.35 with respect to SO' approach. The *SA* approach can achieve average performance improvement of 1.06 when the quality bound $\varepsilon = 0.05$. When $\varepsilon = 0.25$, the average improvement is still 1.04. For normal-10 job sequence, the performance improvement due to 1.25*SA* is over 1.11 in comparison to SO'.

The actual survival probability by SO' is verified to be always 1 on the task sequences when cycle number of each task follows either discrete or continuous distribution. The actual survival probability (actual β in the table) by *SO* and *SA*s for all the cases were also verified, and none of them is less than 80%. We notice that for some job sequences the observed survival probability is higher than $\beta = 80\%$ and sometimes even 100%. The reason is that the product $\prod_{i=1}^{n} \alpha_i$ might be larger than designer specified survival probability setting by Lemma 6.4.2. We conservatively calculate survival probability by taking cool jobs into account, and reduce potential thermal violation risks due to cool jobs in the worst-case scenario. Therefore, the observed survival probability becomes higher than the theoretical one. We also

Jobs	β	SO		1.05SA		1.15SA		1.25SA	
		P improve	actual β	P improve	actual β	P improve	actual β	P improve	actual β
multimedia-8	80%	1.18	88.2%	1.04	100%	1.02	100%	1.02	100%
normal-10	80%	1.10	84.0%	1.03	100%	1.03	100%	1.03	100%
poisson-10	80%	1.05	100%	1.04	94.9%	1.04	94.9%	1.04	100%
uniform-10	80%	1.24	88.2%	1.12	100%	1.11	100%	1.11	100%
normal-15	80%	1.04	83.1%	1.03	94.6%	1.03	94.6%	1.02	95.0%
poisson-15	80%	1.01	100%	1.10	100%	1.10	100%	1.07	100%
uniform-15	80%	1.14	89.0%	1.08	100%	1.08	100%	1.06	100%
normal-20	80%	1.03	83.0%	1.01	98.7%	1.01	99.9%	1.01	99.9%
poisson-20	80%	1.35	98.6%	1.02	99.7%	1.01	99.7%	1.01	100%
uniform-20	80%	1.01	89.0%	1.07	100%	1.07	100%	1.06	100%
average P improve		1.12	/	1.06	/	1.05	/	1.04	/

Table 6.2: Performance improvement and observed survival probabilities

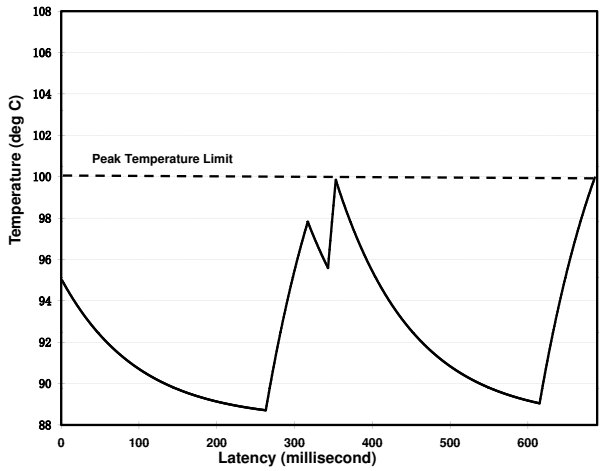


Figure 6.3: Expected thermal curve by deterministic thermal aware design in Chapter 5

notice that the observed survival probability by *SA*s are much higher than the setting and *SO*. This is primarily due to the fact that *SA* is an approximation algorithm and the ε and μ results in a conservative design.

Effect of survival probability setting

We consider the impact of survival probability setting on the performance improvement. Figure 6.5 depicts performance improvement with respect to SO' with different survival probability settings by the proposed techniques: *SO* and *SA* (ε = 0.05, 0.15, 0.25) on multimedia-8 benchmark. We observe that the performance improvement increases when the survival probability setting is lowered. The performance is improved to 1.17 by *SO* when survival probability setting is 85%. In other words, with 15% survival probability relaxation, we can achieve 1.17 performance improvement in comparison to the case with $\beta = 1$. The performance can be improved to 1.10 by 1.15*SA* when survival probability setting is 75%.

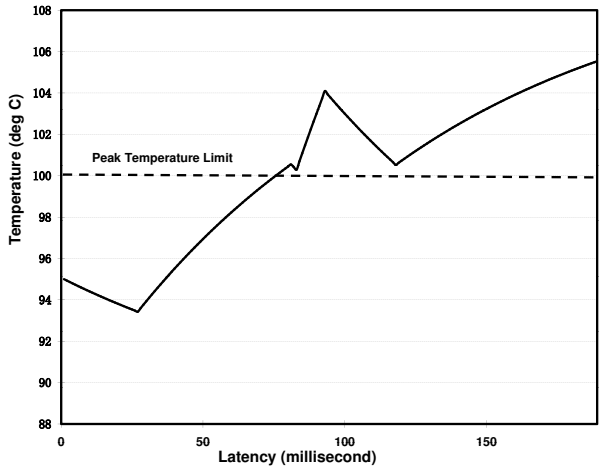


Figure 6.4: Actual thermal curve with average cycle number for the solution of deterministic thermal aware design in Chapter 5

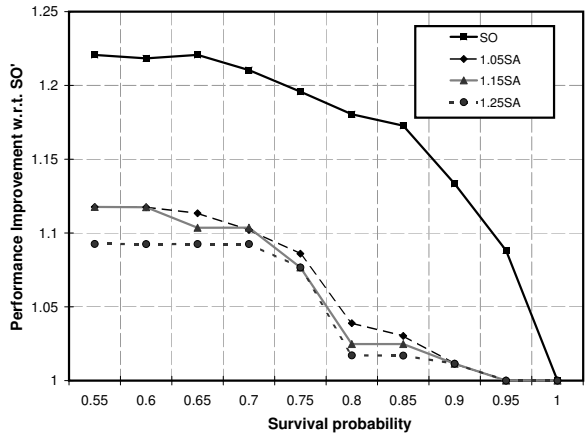


Figure 6.5: Performance improvement w.r.t. SO' with different survival probability β on multimedia-8 benchmark

6.6 Conclusion

We defined the stochastic version of the system level thermal aware design problem. The problem seeks an off-line v/f schedule such that the expected latency is minimized, and survival probability is no less than a specified value β. We proved that the problem is at least NP-hard. We presented an optimal algorithm SO' for the case when survival probability $\beta = 1$. We next presented an optimal algorithm SO that can solve problem instances with arbitrary $\frac{1}{2} \leq \beta < 1$ and discrete distribution of clock cycle demands for each job. Finally, we studied the problem instance when the CPU cycle demand

for each job follows a normal distribution. We proposed a $(1+\varepsilon)$ FPTAS algorithm that can generate solutions in polynomial time when the peak temperature limit T_m is relaxed to $T_m + \mu(T_m - T_{amb})$. We demonstrated that the techniques for deterministic thermal aware design cannot solve the stochastic version of the problem. We presented experimental results that evaluated the performance improvements due to relaxation of survival probability and comparisons of observed survival probabilities with designer specified values.

Chapter 7

Thermal aware task sequencing on embedded processors

The chapter presents the study on the thermal aware task sequencing or ordering problem on embedded processors with or without DVFS capabilities. The objective of the thermal aware design problem is to maximize the throughput for a periodic task set subject to a peak temperature T_m constraint. The problem (denoted as $\mathscr{T}_{\mathscr{S}}$) is motivated by two primary observations (i) task execution order or sequence has a significant impact on thermal profile and consequently the performance of an application, and (ii) arbitrarily long periodic execution of the task set requires the determination of an initial temperature setting T_o^* that enables feasible (T_m is not violated) schedules in all iterations. T_o^* which needs to be determined as part of the problem solution is the optimal initial temperature (at the start of each iteration) of the sequence in steady state that results in highest throughput.

The work is organized as follows: Section 7.1 addresses the motivation and related work, Section 8.3 describes the problem, Section 7.3 finds an optimal initial temperature setting, Sections 7.4 and 7.5 propose algorithms for the problem on processors without and with DVFS capabilities, Section 8.6 presents experimental results and Section 9.1 draws conclusions.

7.1 Motivation and related work

We are interested in the thermal aware sequencing problem for a periodic task set on an embedded processor. Recent work observes that task execution order or sequencing has a profound impact on temperature profile of an application. Jayaseelan et al. [40] enumerate task sequences for a task set with 8 tasks and observe $9.02°C$ difference in the peak temperatures between the worst task sequence (highest peak temperature) and the best task sequence (lowest peak temperature) for identical performance. As we are interested in throughput maximization for a task set subject to peak temperature constraint (T_m), the $9.02°C$ difference could be traded-off for improved throughput for the same T_m constraint in our problem. As the task set is executing on a DVFS and DPM equipped embedded processor, we also need to determine the v/f states of the tasks and sleep times of the processor in the selected sequence.

In addition to determining the task sequence, v/f states and sleep times, the solution for $\mathscr{T}_{\mathscr{S}}$ must also specify the initial temperature for each iteration in the steady state. Zhang et al. [111] observe that for a given task sequence the v/f and sleep time schedules generated by their technique have higher throughput with some initial temperature settings T_o. They observe that the throughput of the solution generated with $T_o = 100°C$ is 1.23 times higher than that with $T_o = 50°C$. Thus, we are interested in finding an optimal initial temperature setting T_o^* in the steady state that can lead to optimum schedules

for $\mathscr{T}_{\mathscr{S}}$, which have the highest throughput. The processors should be able to start with any initial temperature which is no more than the optimal initial temperature under a feasible schedule.

As we are interested in a periodic task set, we must also ensure the feasibility (wrt to peak temperature constraint T_m) of the schedule over multiple iterations. Existing techniques [40, 105] do not consider the periodic execution of their task set. Hence, their schedules for the task set may not be feasible in successive iterations. Existing work [111] has also ensure feasibility by imposing a constraint on the final temperature (T_f) at the end of each iteration $T_f \leq T_o$ (T_o is specified as part of their problem). The constraint is conservative because some schedules with $T_f > T_o$ are feasible under T_m [74]. Quan et al. [74] present necessary and sufficient conditions for the feasibility of schedules. However, their conditions require pseudo-polynomial time feasibility checks at all time points in a period if $T_f > T_o$. Therefore, we need a simpler strategy that ensures the feasibility of schedules over multiple iterations when T_o is not specified as part of the problem.

7.2 Thermal aware sequencing

System model

We consider a processor equipped with a finite number of active discrete v/f states and a sleep state s_s. Each active state s_j is associated with a voltage v_j and a frequency f_j. We assume that the application is specified as a task graph consisting of tasks communicating through finite sized FIFOs. The FIFOs contain data that is transmitted from the producer task to the consumer task. We assume that the various FIFOs in the graph are pre-loaded with sufficient preliminary data to permit the execution of every task. Thus, the tasks can be visualized as independent periodic tasks. Periodicity implies that the task set is executed in a repetitive manner. Once one iteration of the task set is finished, the processor continues to execute the next iteration. We assume each task (say τ_i) is executed at a unique v/f state (say s_j).

For task τ_i at s_j, the processor consumes power $\rho_{ij}(t)$ at time t, which includes dynamic power ρ^d_{ij} and leakage power $\rho^s(t)$ of the processor. The dynamic power ρ^d_{ij} is modeled as the average dynamic power which is a constant for τ_i at s_j. The leakage power $\rho^s(t)$ is temperature-dependent and is approximated by a piece-wise linear equation as $\rho^s(t) = \alpha T(t) + \beta$ [54]. Here α and β are leakage coefficients of the processor and $T(t)$ is the temperature at time t. Therefore, $\rho_{ij}(t) = \rho^d_{ij} + \alpha T(t) + \beta$. The execution time of τ_i at s_j is t_{ij}. *Latency* is defined as the completion time of the task set. To maximize throughput of a periodic task set, we aim at minimizing the latency of the task set per iteration.

We adopt a thermal model widely used by recent work [8, 40, 74, 111] in order to predict future die temperatures. The die temperature of the processor is modeled by a lumped RC circuit with specified

thermal parameters: resistance r and capacitance c. For a task τ_i at state s_j, the die temperature at the time t ($T(t)$) can be derived from the model

$$rc\frac{dT(t)}{dt} + T(t) - r\rho_{ij}(t) = T_{amb} \tag{7.1}$$

T_{amb} is the die's ambient temperature. With the dynamic and leakage power model, we decouple the temperature-leakage dependency to the following equation

$$\frac{rc}{1-\alpha r}\frac{dT(t)}{dt} + T(t) = \frac{(\rho_{ij}^d + \beta)r + T_{amb}}{1-\alpha r} = T_s(\tau_i, s_j) \tag{7.2}$$

Here $T_s(\tau_i, s_j)$ defines the steady state temperature of τ_i at s_j, which is a constant dependent on dynamic power ρ_{ij}^d. Clearly, higher dynamic power of a task results in higher steady state temperature for the task. Let T_o be the initial temperature when $t = 0$. For the task τ_i at s_j starting at time zero, the die temperature at the completion t_{ij} is

$$T(t_{ij}) = T_o e^{-Kt_{ij}} + T_s(\tau_i, s_j)(1 - e^{-Kt_{ij}}) \tag{7.3}$$

Here K is the chip-dependent time constant $K = \frac{rc}{1-\alpha r}$. We consider the processor has a peak temperature limit T_m, beyond which it is unsafe to execute tasks.

Problem definition

The thermal aware task sequencing problem $\mathscr{T}_{\mathscr{S}}$ is defined as follows. Given

- an embedded processor equipped with a set of active v/f states $\mathscr{M} = \{s_1, s_2, ...s_q\}$ with $s_j = \langle v_j, f_j \rangle$ and a sleep state s_s;
- a set of periodic independent tasks $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, where every task τ_i at s_j requires dynamic power ρ_{ij}^d and execution time t_{ij};
- a leakage power model with parameters α and β;
- a thermal model with thermal parameters r and c and a peak temperature limit T_m.

The objective is to seek a schedule $S = \{\Pi, A, T_o\}$ such that (i) the latency of the task set per iteration is minimized and (ii) the temperature of the periodic task set is no more than T_m over multiple iterations. We introduce a sleep task τ_s when processor could sleep in between the normal or active tasks. Π is the task execution sequence including sleep tasks. $\Pi = \tau_{\pi_1} ... \tau_{\pi_i} ... \tau_{\pi_l}$ ($|\Pi| = l \leq 2n+1$), where π_i is the task index at the i_{th} position. A is the corresponding execution time schedule for each task in Π, which specifies v/f state for each active task and sleep time for each sleep task. T_o is the initial temperature

setting of Π with the schedule A. Note that in implementation it is not necessary to start the executions of the task set with T_o. The solution schedules starting with any temperature below T_o are still feasible according to the thermal model. We assume that each task is only executed at one v/f state and processor consumes zero power with negligible switching overheads at sleep state.

There are several special cases of $\mathscr{T}_{\mathscr{S}}$. Based on the task execution order, the special case of the problem with fixed execution order and a given initial temperature T_o has been shown to be NP-hard by a reduction from the multiple choice knapsack problem in [111]. Based on the processor model, the subproblem of $\mathscr{T}_{\mathscr{S}}$ with a given T_o to minimize peak temperature without sleep task has been shown to be NP-hard by a reduction from the bottleneck traveling salesman problem in [40]. Thus, it is clear that the problem is NP-hard. In the next section, we show that, even though the problem is NP-hard, we can find an optimal initial temperature setting T_o^*.

7.3 Optimal setting of T_o

We seek an initial temperature setting T_o^* that can lead to optimum solutions for $\mathscr{T}_{\mathscr{S}}$. At first, we state the following lemma based on Equation 7.3 and the convexity of the function e^{-Kx}.

Lemma 7.3.1. *Consider schedules $S = \{\Pi, A, T_1\}$ and $S' = \{\Pi, A, T_2\}$ with $T_1 \geq T_2$. Let $T(S,t_i)$ and $T(S',t_i)$ be the temperatures at time t_i by S and S'. $0 \leq T(S,t_i) - T(S',t_i) \leq T_1 - T_2$ always holds true and $T(S,t_i) - T(S',t_i)$ is monotonically decreasing over time t_i.*

Proof. By thermal model in Equations 7.2 and 7.3, for a sequence of schedules, we can derive the temperature for any schedule S_a starting with T_o at time t_i as follows.

$$T(S_a,t_i) = T_o e^{-Kt_i} + f \tag{7.4}$$

Here f is a function that is independent to T_o but depends on the execution sequence and time schedules in S_a during $[0,t_i]$. Since the Π, A in S and S' are the same, the f functions in equation 7.4 by S and S' are equal. Thus, we get

$$T(S,t_i) - T(S',t_i) = (T_1 - T_2)e^{-Kt_i} \leq T_1 - T_2 \tag{7.5}$$

The second inequality follows from $0 < e^{-Kt_i} \leq 1$. Since $T_1 \geq T_2$, $T(S,t_i) - T(S',t_i) \geq 0$. Because e^{-Kt_i} is monotonically decreasing over t_i, $T(S,t_i) - T(S',t_i)$ is monotonically decreasing over t_i. □

Then, we find the optimal initial temperature setting as follows.

Theorem 7.3.1. *For $\mathscr{T}_{\mathscr{S}}$, T_m is an optimal initial temperature setting with the property: if there exists a feasible schedule S subject to a peak temperature limit T_m, there always exists one feasible schedule S' that starts with the initial temperature T_m and achieves the same latency as that by S.*

Proof. Assume a feasible schedule for $\mathscr{T}_{\mathscr{S}}$ is $S = \{\Pi, A, T_i\}$ ($\Pi = \tau_{\pi_1} ... \tau_{\pi_l}, T_i \neq T_m$) and the peak temperature by S in multiple iterations is T_p during the execution of task τ_{π_k}. We show that we can always find a feasible schedule starting with T_m for $\mathscr{T}_{\mathscr{S}}$ with the same latency as that by S.

We construct a new schedule $S' = \{\Pi', A', T_m\}$ with $\Pi' = \tau_{\pi_{k+1}} ... \tau_{\pi_l} \tau_{\pi_1} ... \tau_{\pi_k}$ and A' is the same execution time schedule as A but in the order of Π'. For example, suppose that $\Pi = \tau_1 \tau_2 \tau_4 \tau_3 \tau_5$ and the peak temperature T_p occurs during execution of τ_4. Based on thermal model, T_p must be the final temperature of τ_4. Thus, $\Pi' = \tau_3 \tau_5 \tau_1 \tau_2 \tau_4$. We show that S' is feasible for $\mathscr{T}_{\mathscr{S}}$ subject to thermal constraint in the following possible cases.

- Case I ($T_p = T_m$): Π' and A' starting with T_m is obviously feasible under thermal constraint. Thus, S' is feasible for $\mathscr{T}_{\mathscr{S}}$.
- Case II ($T_p < T_m$): Compare the temperatures by S and S' when both schedules start from the execution of $\tau_{\pi_{k+1}}$. The initial temperatures of S and S' are differently T_p and T_m. Because of the periodic nature of the task set, S' and S have the same execution sequence and time schedules after $\tau_{\pi_{k+1}}$. By Lemma 7.3.1, the temperature difference between S' and S is no more than $T_m - T_p$. Because T_p is the peak temperature by S, the peak temperature by S' is no more than T_m.

Finally, S' achieves the same latency as S because it schedules the same task set with the same execution time schedule as S. □

The schedules with initial temperature T_m are guaranteed to be feasible over multiple iterations based on Lemma 7.3.1, because the final temperature of one iteration is always no more than T_m. Therefore, we are able to transform $\mathscr{T}_{\mathscr{S}}$ to a problem with T_m (denoted as $\mathscr{T}_{\mathscr{S}}(T_m)$). $\mathscr{T}_{\mathscr{S}}(T_m)$ only considers $\mathscr{T}_{\mathscr{S}}$ for one iteration of the task set starting with T_m. Based on Theorem 7.3.1, we have

Theorem 7.3.2. *Solving $\mathscr{T}_{\mathscr{S}}(T_m)$ is equivalent to solving $\mathscr{T}_{\mathscr{S}}$.*

Proof. We show that a feasible schedule for $\mathscr{T}_{\mathscr{S}}$ is feasible for $\mathscr{T}_{\mathscr{S}}(T_m)$ with the same latency and vice versa.

Proofs for $\mathscr{T}_{\mathscr{S}} \Rightarrow \mathscr{T}_{\mathscr{S}}(T_m)$: Suppose that a feasible schedule for $\mathscr{T}_{\mathscr{S}}$ is $S = \{\Pi, A, T_i\}$ and the maximum temperature by S is T_p during the execution of task τ_{π_k}. Similar to the proofs for Theorem 7.3.1, we can always construct a feasible schedule starting with T_m for $\mathscr{T}_{\mathscr{S}}$. This schedule is clearly feasible for $\mathscr{T}_{\mathscr{S}}(T_m)$ with the same latency.

Proofs for $\mathscr{T}_{\mathscr{S}}(T_m) \Rightarrow \mathscr{T}_{\mathscr{S}}$: By the definition of $\mathscr{T}_{\mathscr{S}}(T_m)$ and Lemma 7.3.1, a feasible schedule for $\mathscr{T}_{\mathscr{S}}(T_m)$ is obviously feasible for $\mathscr{T}_{\mathscr{S}}$ with the same latency. □

$\mathscr{T}_{\mathscr{S}}(T_m)$ is NP-hard because the equivalent problem $\mathscr{T}_{\mathscr{S}}$ is NP-hard. In later sections, we propose optimal algorithm for several subproblems of $\mathscr{T}_{\mathscr{S}}$ and provide heuristic algorithms for more general instances of $\mathscr{T}_{\mathscr{S}}$ as solutions.

7.4 Sequencing without DVFS

In this section, we solve $\mathscr{T}_{\mathscr{S}}$ for processors without DVFS capability. We assume the v/f state of the processor is fixed. We first present an optimal algorithm for several special cases. Finally, we give an algorithm for the more general instance of the problem.

Task sets with homogeneous power

For a task set with homogeneous power, we assume that the dynamic power consumption of each task τ_i is identical to ρ^d and the associated execution time of τ_i is t_i. Thus, by Equation 7.2, the steady state temperature of each task at the fixed v/f state (denoted as T_s) is equal.

Suppose that $T_s \leq T_m$. This implies that T_m won't be violated due to the executions of these tasks. The optimal schedule is to execute all the tasks in an arbitrary order without sleep tasks. The optimal latency is the summation of execution times of all the tasks.

Now we consider that $T_s > T_m$. This implies that processor may violate T_m due to the executions of these tasks. We provide an optimal algorithm SEQ_f. The main idea of SEQ_f is to minimize sleep time since sleep task is the only candidate for cooling the processor. We set the initial temperature T_o to T_m according to Theorem 7.3.1. Then we arbitrarily pick a task τ_h such that it is executed in the following manner. Processor sleeps the minimum time t_s such that the task τ_h can be finished under T_m and the final temperature of τ_h (denoted as T_f) is T_m. The minimum sleep time t_s is derived from the following equation based on the thermal model

$$T_f = T_o e^{-K(t_s+t_h)} + T_s(\tau_h)(1 - e^{-Kt_h}) \tag{7.6}$$

Here t_h is the execution time of τ_h. $T_s(\tau_h)$ is the steady state temperature of τ_h, which is T_s in this special

case. Repeat this pattern for all the remaining tasks. Then output the schedule S_f. The computational complexity of SEQ_f is $O(n)$.

We can show SEQ_f is optimal for a task set with two tasks based on the thermal model due to the fact that we can only cool the processor with sleep tasks. Then we have Theorem 7.4.1 by visualizing the optimal schedule for two tasks as one unit task and constructing optimal schedule S_f from bottom-up.

We first show that the algorithm generate optimal solutions for task sets with two tasks. Then we show the optimality of the algorithm for a task set with arbitrary number of tasks.

Lemma 7.4.1. *For a task set with two tasks, schedule S_f has the smallest latency under T_m by starting T_m.*

Proof. We consider S_f and S' for two tasks τ_1, τ_2. $S_f = \{\Pi = \tau_s\tau_1\tau_s\tau_2, A = t_{s1}t_1t_{s2}t_2, T_m\}$ and $S' = \{\Pi' = \tau_s\tau_1\tau_2, A' = t'_s t_1 t_2, T_m\}$ with the same latency $D = \sum_{\forall t_i \in A} t_i$. S_f schedules processor sleeps the minimum time t_{s1} derived from Equation 7.6 with $T_o = T_f = T_m$. Repeat this pattern for τ_2. S' schedules processor sleeps $t'_s = t_{s1} + t_{s2}$ and then executes τ_1, τ_2. Let $T(S_f, D)$ and $T(S', D)$ be the temperatures by S_f and S' at time D. By the thermal model, $T(S_f, D) - T(S', D)$ is equal to

$$T_s((e^{-K(t_2+t_{s2})} - e^{-K(t_1+t_2+t_{s2})}) - (e^{-Kt_2} - e^{-K(t_1+t_2)}))$$

Because e^{-Kt} is monotonically decreasing over t and convex, we have $T(S_f, D) \leq T(S', D)$.

Since $T(S_f, D) = T_m$ and $T(S, D) \leq T(S', D)$, S' requires sleep time no less than t'_s such that $T(S', D) \leq T_m$. Therefore, the overall latency of S' is no less than D if S' becomes feasible under T_m. Further, S_f schedules processor to sleep the minimum time such that it is feasible under T_m. Thus, S_f causes the smallest latency. □

Thus, we have the theorem 7.4.1.

Theorem 7.4.1. *S_f is optimal for task sets with homogeneous power consumption on processors without DVFS capability.*

Proof. By Lemma 7.4.1, S_f has the smallest latency for two tasks. For task set with more than two tasks, we visualize the optimal schedule for two tasks as one task. Then we construct optimal schedule bottom-up and finally get an optimal schedule which is S_f. Because the final temperature of each task in S_f for two tasks is T_m, an arbitrary execution order is optimal for the independent periodic task set. Therefore, S_f is optimal for $\mathscr{T}_{\mathscr{S}}$. □

Task sets that all tasks raise temperatures

We consider a task set with heterogeneous power consumption on processors without DVFS capability. We assume the dynamic power of each task τ_i is ρ_i^d and its execution time is t_i. For the special case of $\mathscr{T}_{\mathscr{S}}$ where all the tasks raise temperatures, sleep task is the only candidate for cooling the processor. Thus, we have the following theorem proved similar to Theorem 7.4.1.

Theorem 7.4.2. *The special case of $\mathscr{T}_{\mathscr{S}}$ for task sets where all tasks raise temperatures on processors without DVFS capability is solved optimally by SEQ_f.*

General instance of the problem

We classify tasks into cool and hot tasks for a given T_m. The cool (or hot) tasks are the ones whose steady state temperatures calculated by Equation 7.2 are no more (or more) than T_m. Cool tasks imply that processor is safe to execute these tasks starting with any temperature below T_m, while hot tasks imply that processor is not safe to execute these tasks starting with some temperatures below T_m. The general instance of the problem involves task sets including both cool and hot tasks with heterogeneous power consumption. We have the same assumptions for the dynamic power and execution times of tasks as those in Section 7.4.

For a given T_m, both sleep and cool tasks are candidates for cooling the processor. Clearly, sleep tasks cool the processor faster than cool tasks and can cool the processor to temperatures lower than those by cool tasks. On the other hand, the advantage of cool tasks over sleep tasks is that cool tasks do not introduce extra time to finish the task set.

Thus, to minimize the latency of a task set, we provide a heuristic algorithm SEQ_s in Figure 7.1 based on a property of the schedules. The main idea of SEQ_s is as follows. Starting with T_m, we seek an optimized cool task sequence consisted of all cool tasks (without sleep and hot tasks) to lower the temperature as much as possible. Then we insert hot tasks into cool task sequence such that we can finish all hot tasks without sleep tasks under T_m. If we fail to do so, we introduce sleep tasks.

Optimizing the cool task sequence

Algorithm SEQ_s initially classifies tasks into cool and hot tasks based on their steady state temperatures calculated by Equation 7.2. Then it starts with a task sequence L_c consisted of all cool tasks. L_c arranges the tasks in the decreasing order of their power consumption based on the following lemma. This lemma

$SEQ_s(\Gamma, T_m)$:

```
classify tasks into cool and hot tasks;
L_c = cool tasks in decreasing order of power;
calculate the maximum {T_I} for hot tasks under T_m;
L_h = hot tasks in increasing order of T_I;
T_o := T_m;
while (L_h is not empty){
  τ_h = the first task in L_h;
  if (L_c is not empty) {
      get {T_F} by cooling thermal curve of L_c starting with T_o;
      if (T_F[π_|L_c|] ≤ T_I[h]){
        find T_F[π_{i-1}] > T_I[h] ≥ T_F[π_i];
        put τ_π1, ..., τ_πi, τ_h to tail of Π;
        put t_π1, ..., t_πi, t_h to tail of A;}
      else {
        put τ_π1, ..., τ_πk, τ_s, τ_h, to tail of Π;
        t_s = minimum sleep time starting with T_F[|L_c|] for τ_h;
        put t_π1, ..., t_πk, t_s, t_h, to tail of A; }}
  else {
    t_s = minimum sleep time starting with T_o for τ_h;
    put τ_s, τ_h to tail of Π;
    put t_s, t_h to tail of A; }
  T_o = final temperature by {Π, A, T_m};
  update L_c, L_h;}
return S_n = {Π, A, T_m}.
```

Figure 7.1: Algorithm for $\mathscr{T}_{\mathscr{S}}$ on processors without DVFS

can be proved by a task set with two tasks based on the thermal model. It also holds for an arbitrary task set by swapping neighboring tasks into decreasing order of power.

Lemma 7.4.2. *For a task set with no sleep tasks and thermal constraint, given an initial temperature T_o, the temperature at the completion of the task set is minimized if tasks are executed in the decreasing order of their power consumption.*

Proof. We first show that the lemma holds for two tasks $\{\tau_1, \tau_2\}$ with dynamic power ρ_1^d, ρ_2^d ($\rho_1^d \geq \rho_2^d$). Suppose that $S = \{\tau_1\tau_2, t_1t_2, T_o\}$ and $S' = \{\tau_2\tau_1, t_2t_1, T_o\}$.

Let T_{s1} and T_{s2} be the steady state temperatures for τ_1, τ_2. Clearly, $T_{s1} \geq T_{s2}$ because $\rho_1^d \geq \rho_2^d$. Let $T(S,D)$ and $T(S',D)$ be the final temperatures at $D = t_1 + t_2$ by S and S'. Based on the thermal model, we have

$$T(S,D) - T(S',D) = (T_{s2} - T_{s1})(1 + e^{-KD} - e^{-Kt_1} - e^{-Kt_2}).$$

Since $T_{s2} \leq T_{s1}$ and $1 + e^{-KD} \geq e^{-Kt_1} + e^{-Kt_2}$, we have $T(S,D) \leq T(S',D)$.

Then, we show the lemma holds for an arbitrary task set. Suppose that a task sequence $\Pi = \ldots\tau_i\tau_j\ldots\tau_k$ has the lowest final temperature at τ_k and $\rho_i^d < \rho_j^d$. We get a new task sequence Π' by swapping $\tau_i\tau_j$ to $\tau_j\tau_i$. Therefore, the final temperature at τ_i by Π' is no more than that at τ_j by Π. Because the final temperatures of the two tasks are the initial temperatures for the identical remaining task sequence in Π and Π', the final temperatures by Π' is no more than that by Π by Lemma 7.4.2. Because Π has the lowest final temperature, we can get a sequence in decreasing order of power consumption by swapping neighbor tasks with increasing order of power in Π, and achieve the lowest final temperature. □

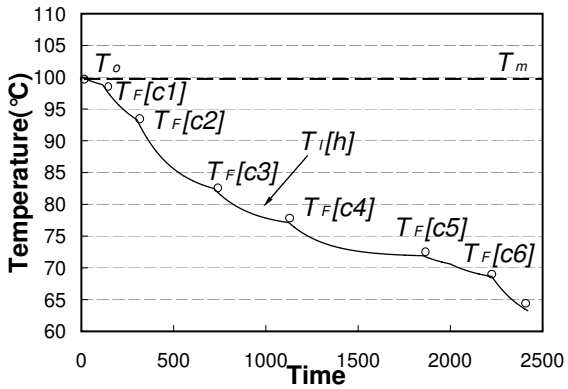
Lowering temperatures with cool tasks

The hot tasks are sorted in the increasing order of their respective maximum initial temperatures (T_I) that enables feasible execution of each task. T_I for each hot task can be derived from the thermal model by setting the final temperature T_f of this task as T_m ($T_f = T_m$). The hottest task is the one with the lowest T_I. SEQ_s in Figure 7.1 calculates T_I for each hot task and let L_h be the unscheduled hot task sequence in increasing order of T_I. The hottest task is the first one in L_h.

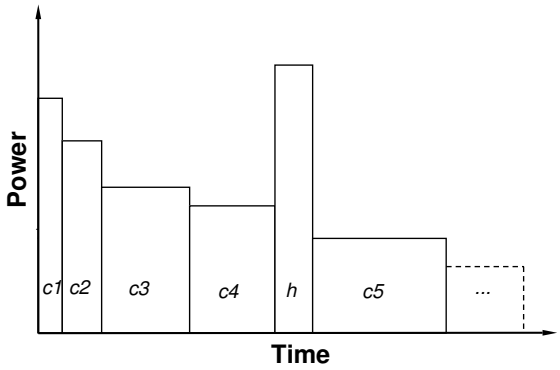
Initially, SEQ_s lowers the temperatures only with cool tasks. It then inserts hot tasks into the cool task sequence L_c. Suppose $L_c = \pi_1\ldots\pi_k\ldots\pi_{|L_c|}$, where π_k is the task index of the kth position in L_c. Set $T_o = T_m$. We generate a cooling thermal curve by executing L_c starting with T_o. The final temperature at each cool task by L_c is recorded in a sequence $\{T_F\} = T_F[\pi_1], \ldots, T_F[\pi_{|L_c|}]$. For instance, suppose that $L_c = c1, c2, \ldots, c7$ with all cool tasks in decreasing order of power. As showed in Figure 7.2(a), we generate a cooling thermal curve starting with T_m based on the thermal model. Each T_F point is the final temperature of a cool task in L_c.

Then, we pick the hottest task τ_h from L_h. Let the maximum initial temperature of τ_h is $T_I[h]$. If $T_F[\pi_{|L_c|}] > T_I[h]$, all the tasks in L_c are not enough to lower the temperature for executing τ_h under T_m. Thus, we add a sleep task with minimum sleep time right after L_c and before τ_h such that τ_h is executable under T_m and the final temperature of τ_h is T_m. $T_F[\pi_{|L_c|}] \leq T_I[h]$ implies that there are enough cool tasks for executing τ_h under T_m. We find the i_{th} position in $\{T_F\}$ such that $T_F[\pi_{i-1}] > T_I[h] \geq T_F[\pi_i]$. We insert τ_h between τ_{π_i} and $\tau_{\pi_{i+1}}$ in L_c. For instance, suppose that current hottest task is τ_h and we find $T_F[c3] > T_I[h] \geq T_F[c4]$ in Figure 7.2(a). We generate a sequence as in Figure 7.2(b).

Next, we update Π and A. We delete the first i tasks from L_c and τ_h from L_h, and update T_o by the final temperature of current schedule $\{\Pi, A, T_m\}$. We repeat the process until all the cool tasks are utilized to lower the temperatures for hot tasks (L_c is empty).



(a) Cooling thermal curve



(b) Generated task sequence

Figure 7.2: Example of SEQ_s

Lowering temperatures with sleep tasks

When L_c is empty, we can only lower temperatures with sleep tasks for the remaining hot tasks. Therefore, similar to algorithm SEQ_f, we add a sleep task with minimum sleep time t_s right before each remaining hot task. t_s is derived from Equation 7.6 with $T_f = T_m$. Thus, we get a solution $S_n = \{\Pi, A, T_m\}$. The computation complexity of SEQ_s is $O(n^2)$.

7.5 Sequencing with DVFS

We consider the general instance of $\mathscr{T}_{\mathscr{S}}$ for processors that have DVFS capabilities. We present a novel algorithm SEQ_d (shown in Figure 7.3). SEQ_d initially triggers SEQ_s with all the tasks at the highest v/f state and achieves a schedule S_V. It then eliminates some sleep times in S_V to reduce latency by scaling down some tasks. Because the scaling-down lowers the temperatures, there is a chance to speed up some tasks under T_m. Thus, SEQ_d incrementally speeds up some tasks to further reduce latency.

$SEQ_d(\Gamma, T_m)$:

```
S_V = SEQ_s(Γ, T_m); /* Γ at highest v/f state */
if (there exists a sleep task in S_V) {
  S_V = ScaleDown(S_V, T_m)
  sign_sp = 1;
  while (sign_sp){
    S_V = SEQ_s(Γ, T_m); /* Γ at v/f states by S_V */
    (S_V, sign_sp) = SpeedUp(S_V, T_m)}};
return S_V;
```

$ScaleDown(S_d, T_m)$:

```
for i = 2: |Π_d|{
  if τ_{π_i} is not τ_s and τ_{π_{i-1}} = τ_s in Π_d{
    t_s = execution time of τ_{π_{i-1}} in A_d
    T_f(i-2) = final temperature of the first i-2 tasks in S_d
    find j in [1 : q] with smallest t_g
    t'_s = preceding sleep time s.t. τ_{π_i j} is executed under T_m;
    if t'_s is not zero {
      replace t_s, t_{π_i} by t'_s, t_{π_i j} in A_d;}
    else{
      replace τ_s, τ_{π_i} by τ_{π_i} in Π_d;
      replace t_s, t_{π_i} by t_{π_i j} in A_d ;}}}
return S_d
```

$SpeedUp(S_u, T_m)$:

```
max G = 0; sign_sp = 0;
for i = 1 : |Γ|{
  t_i = execution time τ_i of in A_u;
  for k = 1 : q{
    if (t_ik < t_i and S_u is feasible when τ_i at s_k) calculate G_ik;
    if G_ik > max G {
     record τ_i as τ_g and v/f state s_k as s_g; update max G;
      sign_sp = 1;}}}
if (sign_sp) modify the v/f state of τ_g in S_u to s_g;
return S_u and sign_sp
```

Figure 7.3: Algorithm for $\mathscr{T}_{\mathscr{S}}$

Scaling down v/f states

For a schedule with sleep tasks, it is beneficial to scale down v/f states of some tasks with preceding sleep tasks. This can be proved by the thermal model and the convexity of e^{-Kx}.

Thus, we initially invoke SEQ_s with tasks at the highest v/f state and get S_V which is then set as the input schedule $S_d = \{\Pi_d, A_d, T_m\}$ of $ScaleDown()$. Note that in this step, if a task at the highest

v/f state is too hot and cannot be scheduled under T_m even starting with T_{amb}, we lower the v/f state of this task to a state such that the task is schedulable starting with T_{amb}. After we get S_d, we try to scale down the v/f state of all the tasks that are preceded by sleep tasks. Suppose $\tau_{\pi_i} \neq \tau_s$ and $\tau_{\pi_{i-1}} = \tau_s$ in Π_d. Here π_i is the task index at the i_{th} position of Π_d. Let t_s be the sleep time of $\tau_{\pi_{i-1}}$ in A_d and $T_f(i-2)$ be the final temperature of the first $i-2$ tasks in S_d. We scale down τ_i to the j_{th} v/f state and scale down the preceding sleep time to t'_s such that their sum $t_g = t'_s + t_{\pi_i j}$ is minimized subject to T_m. Here t'_s is the minimum sleep time preceding $\tau_{\pi_i j}$ starting with $T_f(i-2)$ and can be derived from Equation 7.6. We repeat it for all the tasks in the order of Π_d. Because every task preceded by a sleep task in S_d has final temperature T_m, the solution schedule after scaling down still satisfies thermal constraint based on Lemma 7.3.1. The scaled tasks do not necessarily have the final temperature as T_m because the scaling might cause the final temperatures of some tasks to fall below T_m. This implies there is a chance to speed up some tasks.

Speeding up some tasks

Let the schedule after scaling be S_u. We try to speed up tasks in S_u iteratively in order to further reduce latency. We pick the task τ_i at s_k with the largest gain G_{ik}. G_{ik} denotes the execution time difference (positive) of τ_i before and after the speedup to s_k. τ_i at s_k should maintain the feasibility of S_u. After τ_i is speed to s_k, we trigger SEQ_s with the task set associated with v/f schedule in S_u. Then, SEQ_s outputs a new schedule. We repeat the process until we cannot speed up any task and the final solution is generated. The computational complexity of SEQ_d is $O(n^2q)$ or $O(n^3)$.

7.6 Results

Experimental setup

We considered a PICA processor [89] with 4 DVFS scaling factors $\{1, 0.8, 0.6, 0.4\}$ and the frequencies were in the range of 1.12 GHz to 2.8GHz. The thermal capacitance and resistance settings were derived similar to those in [40, 111] (from HotSpot [93]). We set $50°C$ as die ambient temperature and $100°C$ as peak temperature limit. We set $100°C$ as initial temperature of the processor for the existing techniques.

We evaluated the proposed techniques by experimentations with 18 benchmarks from Mediabench [59], SPEC CPU95 and CPU2000 [2] on the processor model. The Mediabench benchmarks included 10 tasks with five kinds of multimedia applications: jpeg, mpeg2, adpcm, pegwit, epic. The SPEC benchmarks included 8 tasks: sha, compress, gcc, go, applu, mgrid, perl, anagram. We created 8 representative task sets each with 8 tasks chosen from the 18 benchmarks. We obtained cycle number

and dynamic power of each task at the highest v/f state from Wattch [13]. The cycle numbers of tasks were in the range of 10^6 to 10^9 and the dynamic power of tasks were in the range of 25W to 50W. We considered the leakage power with parameters derived from [54]. The steady state temperatures of tasks at the highest v/f state were calculated by our thermal model (Equation 7.2) and were in the range of $90°C$ to $140.3°C$. For each task τ_i at a v/f state with scaling factor s, the dynamic power of τ_i at that state was obtained by scaling the dynamic power of τ_i at the highest v/f state by s^3. The execution time of τ_i with cycle number c_i at the v/f state with scaling factor s was proportional to $\frac{c_i}{sf_b}$, where f_b was the frequency at the highest v/f state.

We obtained a task sequence $MinTP$ to evaluate our techniques. $MinTP$ was the task execution order with the lowest peak temperature (not necessarily lower than T_m) for a task set starting with $100°C$ and executed for 1 iteration. $MinTP$ did not include sleep tasks and it was generated by exhaustive enumeration in exponential time.

We evaluated our technique SEQ_s by assuming that the processor only executed at the highest v/f state. We compared SEQ_s against the following techniques:

- $MinTP+SP$: Because simply executing $MinTP$ starting with T_m violates T_m constraint, $MinTP+SP$ executes $MinTP$ with a sleeping policy SP. SP is: if a task in $MinTP$ violates T_m, we introduce a preceding sleep task with minimum sleep time that is given by Equation 7.6.

- JM_s: JM_s [40] heuristically minimizes the peak temperature of a task set including sleep tasks on processors without DVFS capability subject to a latency constraint. We do a binary search for the smallest latency such that JM_s outputs a feasible schedule under T_m. We recorded the smallest latency as the latency by JM_s.

We next considered the more general instance of the problem for a processor that has DVFS capabilities. We evaluated our technique SEQ_d against the following approaches:

- $MinTP+OptVS$: $OptVS$ [111] optimally solves the latency minimization (or throughput maximization) problem for a given task sequence subject to T_m constraint in pseudo-polynomial time. $MinTP+OptVS$ achieves solutions by invoking $OptVS$ on the $MinTP$ sequence of the task set.

- JM_d: JM_d in [40] heuristically minimizes the peak temperature of a task set under a deadline constraint. Similar to JM_s, we recorded the smallest deadline by a binary search with JM_d as the latency by JM_d.

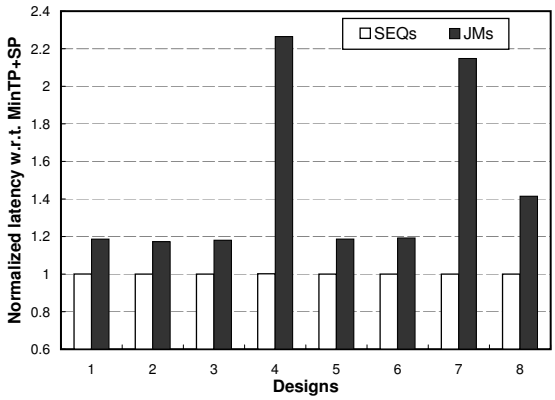


Figure 7.4: Normalized latency on processors without DVFS

The techniques and experimentations were coded in C and performed on a Pentium 4/2.4GHz/1GB Windows XP PC.

Evaluation for processors without DVFS

We compare the latencies for solutions by SEQ_s against $MinTP+SP$ and JM_s. Figure 7.4 plots the normalized latencies with respect to those by $MinTP+SP$. The latencies by SEQ_s are very close to those by $MinTP+SP$ in all tested cases and for task sets 7 and 8 they are exactly equal. Task sets 7 and 8 have tasks that all raise temperatures. As proved in Theorem 7.4.1, the special case of the problem instance (with tasks that all raise the temperature) is solved optimally by SEQ_f (and consequently by SEQ_s). Thus, the experimental results validate our theoretical proofs. Further, SEQ_s outperforms JM_s for all the 8 task sets. For task set 4, JM_s generates schedule with latency 2.26 times that by SEQ_s. On average, SEQ_s generates schedules with latencies that are 73% (27% improvement) of that by JM_s.

Evaluation for processors with DVFS

Figure 7.5 depicts normalized latencies for solutions by SEQ_d and JM_d with respect to those by $MinTP+OptVS$. Interestingly, SEQ_d outperforms or matches $MinTP+OptVS$ in all the 8 cases. This is because $MinTP+OptVS$ does not consider task sequencing and v/f scheduling simultaneously in comparison to SEQ_d. Thus, SEQ_d outperforms $MinTP+OptVS$ in many cases. Further, SEQ_d outperforms JM_d in all the tested cases. On average, SEQ_d generates sequences and schedules with latency 90.5% (min: 84.3% for Task set 5, 9.5% average performance improvement) of that by JM_d.

In terms of average runtime, SEQ_s and JM_s execute in 0.000015s and 9.358s, respectively. SEQ_d takes 0.00063 s, while JM_d and $MinTP+OptVS$ execute in 1.558 s and 16.066 s, respectively.

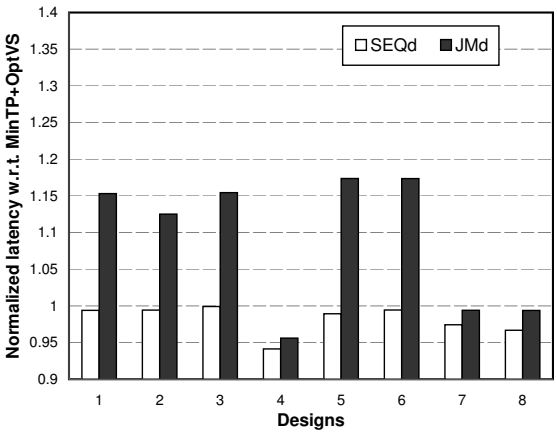


Figure 7.5: Normalized latency on processors with DVFS

7.7 Conclusions

We proposed thermal aware sequencing techniques to maximize the throughput of a periodic task set subject to a peak temperature limit. As part of the solution, we found an optimal initial temperature setting T_o^* that can lead to optimum solutions and guarantees the solution feasibility over multiple iterations. Next, we proposed an optimal algorithm for special cases on processors without DVFS capability for (i) task sets with homogeneous power consumption and (ii) task sets having heterogeneous power consumption where all tasks raise the temperature. We then developed several sequencing properties and proposed a novel algorithm SEQ_s for the problem on processors without DVFS capability. Finally, we proposed a novel algorithm SEQ_d for the general instance of the problem. Experimental results showed that our algorithms outperform existing techniques JM_s and JM_d [40].

Chapter 8

Thermal aware scheduling by considering the impact of package temperature

The chapter addresses a thermal aware design problem for a periodic task sequence on an embedded processor under a peak temperature constraint. We consider a temperature-dependent leakage power model with discrete voltage/frequency settings and a sophisticated thermal model derived from HotSpot for an embedded processor with die and package. We prove that the problem is NP-hard. We provide a pseudo-polynomial time optimal algorithm and a fully polynomial time approximation scheme (FPTAS) based technique as solutions to the problem. The solution techniques to the thermal aware design problem are constructed on the top of solutions to a subproblem with package temperature and power budget constraints. We show the NP-hardness of the subproblem. We provide a pseudo-polynomial time optimal algorithm and a bi-criteria FPTAS as solutions for the subproblem. The bi-criteria FPTAS generates solutions within guaranteed quality bound when the power budget constraint is relaxed to a certain amount. We evaluate our techniques by simulations with realistic and synthetic benchmarks mapped to an embedded CMOS processor. The simulation results demonstrate our FPTAS based technique for the addressed thermal aware design problem is able to match optimal solutions when a designer specified quality bound (QB) is set at 10%, can generate solutions that are quite close to optimal ($< 3\%$) even when QB is set at a higher value (50%), and executes within 20 seconds (with QB $\geq 50\%$) for large task sets with 50 nodes (while the optimal technique takes several hundreds of seconds).

The work is organized as follows: Section 8.1 discusses previous work and delineates the contributions of the work, Section 8.2 introduces the power, thermal and task models for the thermal aware scheduling problem, Section 8.3 formally defines the thermal aware scheduling problem and proves that it is NP-hard, Section 8.5 presents the optimal algorithm for the problem, Section 8.5 proposes the FPTAS for the problem, Section 8.6 presents the experimental results, and finally Section 9.1 concludes the work.

8.1 Previous Work

The work addresses thermal aware scheduling problem on a single processor based on a CTM derived from the HotSpot simulator. The related research can be classified into five categories as shown in Table 8.1 on the basis of the problem formulation, application domain and solution strategies. The first three classification schemes in the table are based upon the problem formulation, while the fourth and fifth schemes are based on application domain and solution strategy, respectively. The scope of our work can be classified as *optimal and approximation algorithms for off-line, inter task DVFS technique with*

Table 8.1: Classification based on problem formulation, application domain and solution strategy

Classification scheme		Existing work
1	Off-line or design time algorithms	[17, 19, 20, 22, 30, 55, 64, 74, 77, 79, 98, 99]
	On-line or real-time algorithms	[7–9, 12, 26, 61, 92–94]
2	Inter-task DVFS (task executes at one DVFS state)	[7–9, 17, 20]
	Intra-task DVFS (task runs at many DVFS states)	[10, 12, 19, 22, 26, 30, 55, 61, 64, 74, 76, 79, 92–94, 98, 99]
3	Discrete DVFS states	[12, 17, 19, 20, 26, 92–94]
	Continuous DVFS range	[7–9, 22, 55, 64, 77, 79, 98, 99]
4	General purpose computing	[17, 18, 23, 64, 79, 85]
	Embedded computing	[7–10, 19, 20, 22, 74]
5	Heuristic approaches	[12, 17, 26, 55, 61, 68, 92–94]
	Optimal or approximation algorithms	[7–9, 19, 20, 22, 64, 77, 79]

discrete DVFS and DPM states aimed at embedded computing systems.

Our problem instances are characterized by the consideration of a realistic processor model that supports only discrete v/f states (as opposed to the idealistic scenario with continuous speed settings). Note that our discrete v/f state consideration differs from the existing approaches [79, 99] in that the available v/f states are the inputs to our problem. Further, similar to [7–9, 107], we consider that each task operates at a single v/f state. This is due to the consideration that inter-task v/f scheduling has less overhead than intra-task techniques, and is easier (more practical) to implement [97]. General purpose thermal aware design approaches address the thermal aware design problem under a continuous workload model, and do not incorporate the notion of discrete tasks. Most of the approaches aim at workload maximization (clock cycles executed) under thermal constraints on a processor that supports

continuous DVFS settings. In contrast we are focused upon embedded computing systems with well defined task sets executing on a processor with discrete DVFS states. Finally, our research focuses on off-line optimal/approximation approaches for the addressed thermal management problems. The proposed approaches are able to generate DVFS and DPM schemes with provably solution quality bounds in polynomial time.

The thermal model considered by our approaches accounts for the impact of leakage power consumption and package temperature on the die temperature of the processor. In comparison to the existing techniques that fall within the scope of our work (as described above) there are two fundamental problems of system-level thermal management on processors with discrete v/f states that have not been addressed by previous work. *What is the tight upper bound of the performance under a thermal constraint with discrete DVFS for a periodic job sequence executing on an embedded processor ? How can we efficiently achieve a good schedule within a quality bound of the optimal ?* The answers to these problems would not only enable thermal aware (static) design of embedded systems, but also provide a good basis to evaluate some on-line techniques. Moreover, an efficient approximation algorithm with guaranteed quality bound would be applicable in run time. This work provides answers to these questions.

8.2 Preliminaries

Processor power consumption model

We consider an embedded processor with discrete active v/f states and a sleep state [41, 81]. In a particular active state, the dynamic power of the processor can be estimated off-line by static analysis of applications. We denote the total dynamic power of the processor during executing job J_i in state s_j at time t is $\rho_{ij}(t)$. If the job and execution state are not specified, we denote the dynamic power of the processor at time t as $\rho(t)$. The total leakage power dissipation of the processor at time t is temperature dependent, which can be approximated by a piece-wise linear form as $\alpha T(t)+\beta$ [55]. Here α and β are constant leakage coefficients of the processor and $T(t)$ is the current temperature of the die at time t. The total power dissipation of the processor at time t is the summation of dynamic power $\rho(t)$ and the leakage power at time t, which is given by

$$p(T,t) = \rho(t) + \alpha T(t) + \beta \tag{8.1}$$

In the equation the term $p(T,t)$ consists of two factors; a die temperature dependent component that is denoted as $p_T(t) = \alpha T(t)$, and a die temperature independent component which is given by $p_s(t) = \rho(t) + \beta$. The switching overheads between various active states and from active to sleep state are

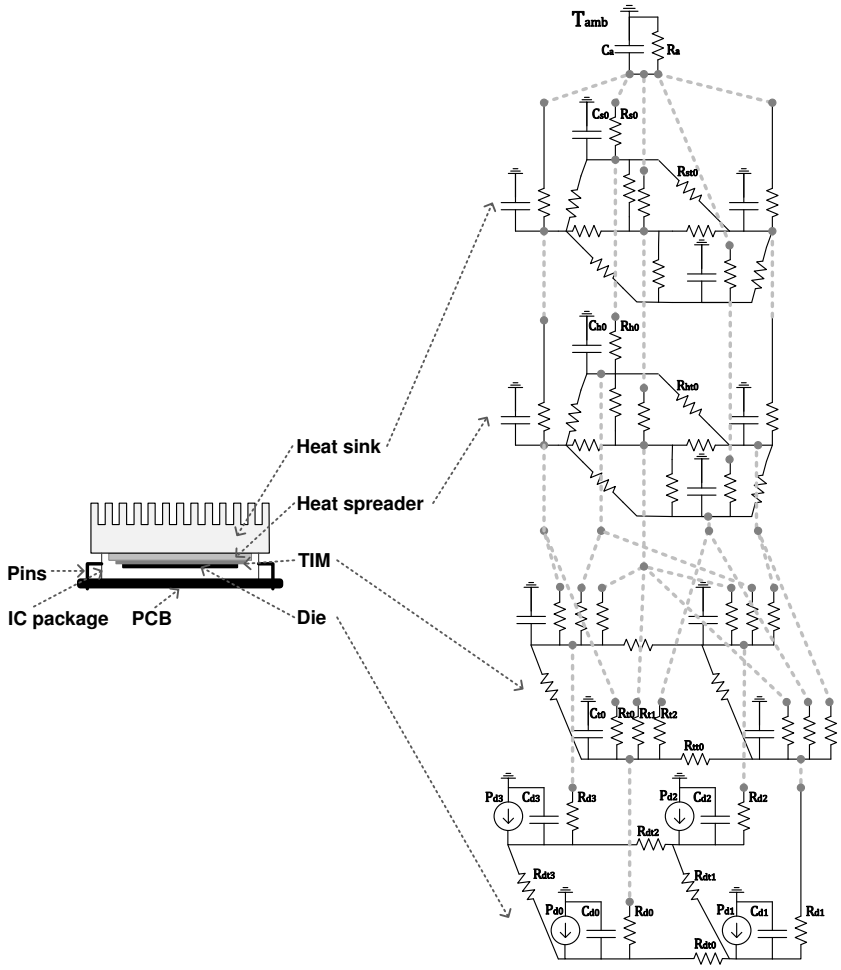


Figure 8.1: Hotspot compact thermal model for quad-core processor

considered to be negligible in comparison to task run times [68]. The wake-up overhead from the sleep state is assumed to be a processor dependent constant.

Processor thermal model

The processor temperature profile is generated from a first-order Resistor-Capacitor (RC) compact thermal model (CTM) derived from the Hotspot simulator [32]. Figure 8.1 depicts the Hotspot quad-core RC CTM. Hotspot models multiple layers of the processor including the die, thermal interface material (TIM), heat spreader, heat sink, and finally the ambient environment. In the model the intra-layer thermal conductance is denoted by R_{xti} (unit $W/°C$) where $x = d,t,h$ or s for die, TIM, heat spreader, or heat sink, respectively. The inter layer thermal conductance is denoted by R_{xi} (unit $W/°C$), and the thermal capacitance of each layer is given by C_{xi} (unit $J/°C$). The heat generation on the die is a function of the processor power consumption and is represented by perfect current sources in the RC CTM as P_{di}.

Finally, R_a and C_a denote the ambient thermal conductance and capacitance, respectively, and T_{amb} represents the ambient temperature. As such the Hotspot model is primarily meant for simulation, and is not conducive for design time thermal optimization techniques.

We derive a *RC* thermal model which is conducive for design time optimization from the Hotspot CTM by making several key observations. The intra-layer thermal conductances are much larger (at least 4 times) than the corresponding inter-layer thermal conductances, that is $R_{xti} >> R_{xi}$. This is due to the fact that the lateral heat-transfer cross sectional areas are much less than vertical ones, and consequently their contribution to the thermal RC time constants is negligible in comparison with the vertical thermal conductances. Further, the $R_{xi}C_{xi}$ time constants of the TIM, heat spreader and heat sink are much larger than the $R_{di}C_{di}$ time constant of the die. As the thermal aware design problem primarily focuses on satisfying the temperature constraints on the die, the *RC* model of the TIM, heat spreader and heat sink can be collapsed into a single *RC* model for the chip package. Further, as we are interested in a single embedded processor core, and we ignore the impact of intra-layer thermal conductances, the die temperature is approximated as uniform across the core. This assumption is supported by the observation that the hotspot of a processor is well-defined and alleviates the need for a finer granularity CTM [98]. The current ACPI specification [3] includes an example for thermal management with the processor specified as a single thermal zone. We derive the CTM (left hand side of Figure 8.2). In the following paragraphs we discuss the calculation of the die and package temperatures based on this CTM.

Die temperature calculation

In the derived CTM (see Figure 8.2) R_d is the thermal conductance from the die to the package, and C_d is the die capacitance. Similarly, R_p is the thermal conductance from the package to the ambient environment, and C_p is the package capacitance. The die temperature has a much smaller time constant ($R_dC_d \approx$ in the order of ten milliseconds) than the package temperature time constant ($R_pC_p \approx$ in the order of one minute). For time duration of the order of $0.1R_pC_p$, the package temperature can be considered to be constant. Consequently, the die temperature can be modeled by the bottom right hand side of Figure 8.2, and calculated by the following equation:

$$C_d \frac{dT(t)}{dt} = -\frac{T(t) - T_p(t)}{R_d} + p(T(t), t) \tag{8.2}$$

$T(t)$ and $T_p(t)$ are the respective die and package temperature at time t. We replace $p(T(t), t)$ by the power model in Equation 8.1, and combine the parameters of $T(t)$. Let $R'_d = \frac{R_d}{1-\alpha R_d}$. We have the following equation:

$$C_d \frac{dT(t)}{dt} = -\frac{T(t)}{R'_d} + (\frac{T_p(t)}{R_d} + \beta + \rho(t)) \tag{8.3}$$

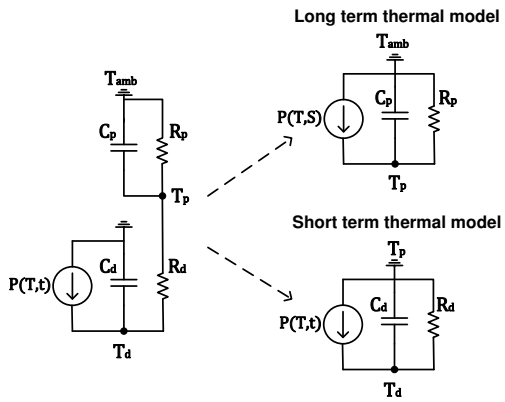


Figure 8.2: Compact thermal model for single core derived from Hotspot

Note that the $T_p(t)$ is considered as a constant during $0.1R_pC_p$. Therefore, the equation is an ordinary differential equation (ODE). By solving Equation 8.3, the die temperature is calculated by the following equation:

$$T(t) = T(0)\exp^{-\frac{1}{R'_dC_d}t} + T^s_d(t)(1-\exp^{-\frac{1}{R'_dC_d}t}) \tag{8.4}$$

Here $T^s_d(t)$ is considered as the steady state die temperature at the end of the time period $t = 0.1R_pC_p$. Because $0.1R_pC_p >> R_dC_d$, $T^s_d(t)$ is calculated by considering $\frac{dT(t)}{dt} \approx 0$ at the end of time period $0.1R_pC_p$.

$$T^s_d(t) \approx R'_d(\frac{T_p(t)}{R_d} + \beta + \rho(t)) \tag{8.5}$$

From the equation, $T^s_d(t)$ is a function of package temperature $T_p(t)$ and the dynamic power $\rho(t)$.

Calculation of the package temperature

The package temperature is approximated in the top right hand side of Figure 8.2 and is updated every $0.1R_pC_p$. We have the following equation.

$$C_p\frac{dT_p(t)}{dt} = -\frac{T_p(t)-T_{amb}}{R_p} + p(T(t),t) \tag{8.6}$$

We substitute $p(T(t),t)$ by Equation 8.1, and replace the $T(t)$ in Equation 8.1 by Equation 8.5 since the package temperature is updated every $0.1R_pC_p$. Then we combine the parameters of $T_p(t)$. Let $R'_p = \frac{R_pR_d}{R_d-\alpha R'_dR_p}$. Thus, we have

$$C_p\frac{dT_p(t)}{dt} = -\frac{T_p(t)}{R'_p} + \frac{\alpha R'_d}{R_d}T_{amb} + (1+\alpha R'_d)(\beta+\rho(t)) \tag{8.7}$$

Again, this is the form of an ODE. We solve it to update $T_p(t)$ every $0.1R'_pC_p$. When the processor dissipates a fixed amount of dynamic power $\rho(t)$ for a long time $t >> R'_pC_p$, the package temperature

reaches the steady state temperature, which is given by

$$T_p^s(t) \approx R'_p(\frac{\alpha R'_d}{R_d} T_{amb} + (1+\alpha R'_d)(\beta + \rho(t))) \tag{8.8}$$

Calculation of the die temperature at the completion of each job

When the processor executes a job J_i in state s_j, we can achieve dynamic power consumption traces by offline analysis. Assume that we know the die temperature $T_d(0)$ and package temperature $T_p(0)$ at the start of the first job. Once the job schedule (execution state of each job) is known, we feed the corresponding dynamic power into the Equation 8.4, and calculate the final die and package temperatures at job completion by Equation 8.4. The package temperature is updated by solving Equation 8.7 every $0.1R_pC_p \approx 10s$ or at the completion of each job (depending upon the earliest time instance). The die and package temperatures at the completion of a job are the starting die and package temperatures of the following job. Then we calculate the die and package temperatures at the completion of the following job. Once a job is finished (starting from time 0 and finishing at time t), the die and package temperature changes during the time t are respectively denoted by $\Delta T_d = T_d(t) - T_d(0)$ and $\Delta T_p = T_p(t) - T_p(0)$.

When the processor is in sleep state, the dynamic power is very small and negligible. The die temperature gradually approaches the package temperature. If the time is long enough, the package temperature gradually approaches to the ambient temperature T_{amb}. Typical transition time of the die temperature to package temperature is of the order of tens of milliseconds [93].

Task model

We consider a periodic task set described as a sequence of n jobs $J = \{J_1, J_2, ..., J_n\}$. The jobs are independent and the order of execution is specified by the sequence. Periodic task set denotes that the sequence of n jobs are executed in an iterative manner[1]. Once one run of the task set is finished, the processor continues to execute the task set for the next run. We are interested in the design time or static version of the thermal aware design problem. Thus, we assume that the tasks have been characterized for their run times. The worst case execution time of each job J_i in v/f state s_j is known and is denoted by t_{ij}. The duration of each job at the lowest v/f state (slowest frequency) is in the range of ten to hundreds of milliseconds which is comparable to the die temperature time constant (R_dC_d). The number of iterations of the entire application is many or infinite.

The task model as described is encountered in communication and multimedia sub-systems of many embedded computing systems. These sub-systems display dataflow behavior, and they can

[1]Note that the concept of periodicity is different from the traditional concept in real-time schedules [19, 52, 53, 98, 99].

be most naturally specified as a set of jobs iteratively executing over a stream of data. For example H.264 decoding can be expressed by our task model. As such applications are often encountered in mobile embedded devices which only include the basic convection cooling mechanism without a fan, the thermal aware design problem as addressed in the work is of particular significance.

8.3 Problem Description

The thermal-aware performance optimization problem TA_{min} can be described as follows. Given:

- a processor with one sleep state s_{sleep} with power consumption ρ_{sleep} and a set of active v/f states $M(|M| = m)$ with technology dependent parameters α and β;
- a processor thermal model with die thermal resistance as R_d and thermal capacitance as C_d, package thermal resistance as R_p and thermal capacitance as C_p;
- a periodic sequence of n independent jobs $J = \{J_1, J_2, ..., J_n\}$ with t_{ij} denoting the run time of job J_i at v/f state s_j and ρ_{ij} denoting the dynamic power consumption of job J_i at v/f state s_j ($s_j \in M$);
- a peak temperature limit T_{max}.

The objective is to obtain an assignment of one active v/f state for each job, and select the processor sleep times such that the latency of the n jobs is minimized subject to the peak temperature constraint. Our problem definition considers that the entire task set executes in a periodic manner for a long time (infinite for the purposes of thermal modeling). As we are interested in generating schedules that are valid under all temperature conditions, we consider the initial die temperature at the start of each iteration to be T_{max}. The associated initial package temperature is a function of the schedule that is generated as part of the solution. The problem as described is a discrete optimization problem with nonlinear continuous feedback constraint. In the remainder of the work we use jobs and tasks to refer to the same entity.

We incorporate the sleep modes in the problem formulation by considering a sequence of $N = 2n+1$ jobs $J' = \{J'_1, J'_2, ..., J'_{2n+1}\}$. Each J'_i when i is an even number refers to the job $J_{i/2}$ from the original set (named active jobs), and when i is odd refers to a job J_s that denotes that the processor is in sleep state (named sleep jobs). The only difference between J and J' is that J' includes all the jobs in J and sleep jobs before and after each job in J. For example, given a task sequence J with $n = 3$ active jobs, $J = \{J_1, J_2, J_3\}$ we can construct a new sequence J' with $2n+1 = 7$ jobs as follows $J' = \{J_s, J_1, J_s, J_2, J_s, J_3, J_s\} = \{J'_1, J'_2, J'_3, J'_4, J'_5, J'_6, J'_7\}$. Note that the order of active jobs in J remains the same in set J'.

Assume the maximum die cooling transient time at a steady state package temperature is t_{ms}, estimated by cooling the processor die from T_{max} to T_{amb} in sleep mode (package temperature is assumed to be at T_{amb}). The execution time of J_s is in the range of $[0, t_{ms}]$. A sleep time of more than t_{ms} lowers the performance in terms of more execution time with no reduction in temperature. We consider the range $[0, t_{ms}]$ as q distinct values $\{t_1, t_2, \ldots t_q\}$ in increments of $t_{ms}/(q-1)$. Thus, if $t_{ms} = 100$ and $q = 11$ we consider the following values $\{0, 10, 20 \ldots 100\}$ [2]. We assume that the length of sleep interval is selected from one of the distinct values in the range. Note that 0 belongs to the distinct set of values and it implies that the processor does not go into the sleep mode. We can integrate the decision problem associated with sleep and active jobs by considering that each job J_i' has $r = (m \text{ or } q)$ different choices ($r = m$ if i is even, else $r = q$), and each choice has an associated execution time given by t_{ij} ($1 \leq i \leq 2n+1, 1 \leq j \leq r$). Thus, TA_{min} can be formulated as follows:

$$TA_{min} : \min Z = \sum_{i=1}^{2n+1} \sum_{j=1}^{r} t_{ij} x_{ij}$$

$$subject\ to\ C_d \frac{dT(t)}{dt} = -\frac{T(t)}{R_d'} + (\frac{T_p(t)}{R_d} + \beta + \rho(t)) \tag{8.9a}$$

$$C_p \frac{dT_p(t)}{dt} = -\frac{T_p(t) - T_{amb}}{R_p} + p(T(t), t) \tag{8.9b}$$

$$\sum_{j=1}^{r} x_{ij} = 1, \forall i \in [1, 2n+1]; \tag{8.9c}$$

$$T_d(t=0) = T_{max}; T_d(t) \leq T_{max}; \tag{8.9d}$$

$$x_{ij} = \{0, 1\}; \tag{8.9e}$$

The objective is to minimize the execution time per iteration of the job sequence. Constraints 8.9a and 8.9b specify the thermal model. Constraint 8.9c demonstrates if i is even and $x_{ij} = 1$ the solution to the above formulation denotes that job $J_{i/2}$ executes in active state s_j for time t_{ij}. Similarly, when i is odd and $x_{ij} = 1$ the processor enters the sleep state for time t_{ij}. We assume the various time values in the problem formulation are integral. Constraint 8.9d specifies the initial die temperature setting. It also specifies that the peak die temperature during multiple iterations of job sequence execution should be no more than T_{max}. The above formulation includes non-linear die and package thermal models, where temperatures are determined by equivalent first order RC circuits. However, even if the thermal models were linear and the package temperature is stable, the problem can be shown to be NP-hard.

Theorem 8.3.1. *TA_{min} is NP-hard.*

Proof. Consider a special case of TA_{min}. We assume the package temperature is stable at T_{amb}. Further, we assume that processor sleeps only at the beginning for t_{ms} time such that the task sequence can be

[2] The distinct values in the range do not necessarily have an equal interval between any two neighboring values. For example, we can make the first value as 0, which represents that the processor does not go into sleep mode. We can make the second value as the wakeup overhead plus the sleep time that represents the wakeup overhead is considered.

executed under T_{max} for repetitive execution of the schedule. After the sleep job, the die temperature becomes T_{amb}. We assume the maximum run time of each actual job is small enough such that the thermal curve is linear. The special case implies that the thermal curve of a feasible schedule would be monotonically increasing after the sleep job. The final die temperature is achieved on the completion of all actual jobs, which should be no more than T_{max}. Thus, the objective function can be specified in terms of the execution time of actual jobs (without the sleep jobs) $\min Z = \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} x_{ij}$. As we consider that the die thermal curve is linear, die thermal constraints (8.9a) and (8.9d) can be replaced by $T_{amb} + \sum_{i=1}^{n} \sum_{j=1}^{m} \Delta T_{ij} x_{ij} \leq T_{max}$ where ΔT_{ij} denotes the die temperature increase due to the execution of job J_i in active state s_j.

The special case of TA_{min} can be shown to be NP-hard by a polynomial reduction from the well known multiple-choice knapsack problem (MCKP), which is NP-hard. Let t_{max} be the upper bound on the execution time of any job, that is $t_{max} = \max\{t_{ij}\}, \forall J_i \in J, j \in m$. The saving in execution time due to a job J_i operating in active state s_j is given by $t_{max} - t_{ij}$. Finding an optimal solution to the problem with an objective of maximizing the execution savings is equivalent to solving the MCKP. Thus, the TA_{min} is NP-hard. □

In the following sections we provide optimal and FPTAS based algorithms as solutions to the thermal aware scheduling problem.

8.4 TA_{min} for periodic job sequences

In this section, we address the TA_{min} problem for periodic job sequences as specified in the task model. We consider the die and package temperature vary during the short and long term of the job sequence execution. We also consider the impact of the package temperature on the die temperature. When the job sequence is executed with many (or even infinite) iterations by a schedule, the package temperature is heated up to a steady state temperature by the average power dissipated by the processor. We consider the average power dissipated by the processor as the average power of the schedule per iteration when package temperature reaches steady state. We seek the optimal schedule with minimal latency such that the die and package temperatures remain under T_{max} all the time.

Optimal solution

Main idea

The main idea for the optimal schedule is based on the following lemma.

Lemma 8.4.1. *Given a schedule S consuming the average power ρ_{av}, if S is feasible under T_{max} constraint when package temperature is in the steady state, S is always feasible under T_{max} constraint.*

Proof. Based on the package thermal model in Equation 8.8, a fixed amount of average power of the job sequence by schedule S causes package temperature rise to a steady state (say T_{sp}). According to the package temperature model in Equation 8.7, T_{sp} is the highest that the package temperature could rise during multiple iterations of S. Based on the die temperature model in Equation 8.4, the higher package temperature causes the higher die temperature profile for a schedule. Since T_{sp} is the highest package temperature during multiple iterations of S, S executed at T_{sp} has the highest die temperature profile comparing to S executed at all the package temperatures lower than T_{sp}. Because S executed at T_{sp} is feasible under T_{max} constraint, the lemma is proved. □

From the lemma, the optimal schedule (denoted by S^*) has the following properties.

i. When the package temperature is in steady state (denoted by T^*_{sp}), the die temperature per iteration is no more than T_{max}.

ii. When the package temperature is in steady state, the average power consumed by S^* keeps the package temperature below T^*_{sp}.

iii. The latency of the job sequence per iteration is minimized.

The properties (1) and (2) specify S^* is feasible under T_{max} constraint all the time based on the lemma. The property (3) specifies the latency of S^* is the smallest. To achieve the optimal schedule S^*, we utilize the following steps.

i. For a given steady state package temperature T_{sp}, we calculate the average power ρ_{av}. ρ_{av} is the maximum average power of feasible schedules that ensures the package temperature is no more than T_{sp}. We define ρ_{av} as the associated power budget to T_{sp}.

ii. For a given T_{sp}, we have a test procedure that answers the following question. *Suppose that the package temperature is in a steady state T_{sp} associated with a power budget ρ_{av}. Assume the processor starts from the die temperature T_{max} and the package temperature T_{sp}. Does there exist a schedule S such that during one iteration of the schedule the peak temperature limit T_{max} is not exceeded, the latency of one iteration is minimized and the average power of S is no more than ρ_{av}?* A solution schedule for the question is the feasible schedule with minimal latency when the package temperature is in the steady state T_{sp}.

iii. We utilize a search algorithm for the optimal solution based on the test procedure with many T_{sp} values. The test procedure returns solution schedules for these T_{sp} values. The solution schedule with the smallest latency is the final solution.

Find the associated power budget to a given T_{sp}

For a given steady state package temperature T_{sp}, the associated power budget is the average power consumption that leads to the T_{sp}. According to the thermal model in Equation 8.8, we replace $T_p^s(t)$ by T_{sp} and replace $\rho(t)$ by ρ_{av} for the steady state package temperature. Thus, ρ_{av} is given by

$$\rho_{av} = \frac{1}{1+\alpha R'_d}(\frac{1}{R'_p}T_{sp} - \frac{\alpha R'_d}{R_d}T_{amb}) - \beta \tag{8.10}$$

Here ρ_{av} is the average power that continuously heats the chip package when the job sequence is executed in many iterations. Since ρ_{av} is calculated from the steady state package temperature model, ρ_{av} is the maximum average power of schedules that guarantees the package temperature is below T_{sp}.

Test procedure for TA_{min}

We address the question to be answered by the test procedure as a subproblem of the TA_{min} problem (denoted by TAP_{min}). The TAP_{min} problem is the TA_{min} problem for the job sequence with power budget constraint when the package temperature is in steady state. The TAP_{min} problem described in Section 8.5 has two more constraints.

i. The package temperature remains in a steady state T_{sp};

ii. The average power of a solution schedule should not exceed the associated ρ_{av} to T_{sp}.

The first constraint specifies the steady state package temperature setting. The second constraint ensures during multiple iterations of a solution schedule the package temperature does not exceed the steady state package temperature setting for the TAP_{min} problem.

In Section 8.5, we provide an optimal algorithm $TAP-OPT$ and a polynomial-time approximation algorithm $TAP-FPTAS$ as solutions to the TAP_{min} problem. The solution techniques in Section 8.5 perform as test procedures for the TA_{min} problem. If a solution exists for the subproblem, our test procedure $TAP-OPT/TAP-FPTAS$ produces an optimal/approximated schedule at the package temperature setting T_{sp} such that the peak temperature constraint is satisfied, the actual average power of the schedule is no more than power budget associated with T_{sp}, and the latency per iteration is minimized.

Search for the optimal solution

We utilize a search algorithm based on the test procedure to find the optimal steady state package temperature setting that produces the optimal solution for the TA_{min} problem. In the search algorithm, the test procedure is the $TAP-OPT$ algorithm described in Section 8.5. One straightforward method for the optimal solution is to utilize the test procedure to test each possible package temperature setting in the range of lower and upper bound. The upper bound of the steady state package temperature setting is T_{max} since T_{max} is the peak temperature limit. The lower bound of the steady state package temperature setting is T_{amb}. We discreterize the range of the steady state package temperature settings at the granularity of $1°C$. In our technique we further reduce the search space based on the following property.

We observe that the package temperature and power budget constraints in the subproblem TAP_{min} are correlated. When the steady state package temperature setting T_{sp} is set to a higher value (the package temperature setting constraint is tighter), the associated power budget to T_{sp} is bigger (the power budget constraint is looser). The correlated constraints cause the existence of a knee with an optimal steady state package temperature setting T^*_{sp} specified in the following property.

Lemma 8.4.2. *If $T_{sp} > T^*_{sp}$, the optimal latency to the TAP_{min} problem is monotonically increasing as T_{sp} increases. If $T_{sp} < T^*_{sp}$, the optimal latency to the TAP_{min} problem is monotonically decreasing as T_{sp} increases.*

We prove the property with the illustration of the figure 8.3. Figure 8.3 shows two scenarios of the TAP_{min} problems with relaxed constraints. The gray full line plots the optimal latency Z values to the TAP_{min} problems when the power budget constraint is relaxed to infinity. x axis represents the package temperature setting T_{sp} for the TAP_{min} problems. y axis represents the optimal solution Z for the TAP_{min} with various T_{sp} settings. As T_{sp} increases, the processor sleeps more or executes at a lower v/f state in order to maintain the die temperature under T_{max}. Hereby, the optimal latency to the TAP_{min} problem monotonically increases as T_{sp} increases. In the other scenario, the black dotted line plots the optimal returns Z to the TAP_{min} problems when the package temperature setting T_{sp} is relaxed to T_{amb}. x axis represents the power budget. Note that the power budget is a function of the package temperature setting. We map the power budget axis to the associated package temperature setting axis in the plot. y axis represents the optimal solution Z for the TAP_{min} with various power budget values. As the power budget increases, the processor reduces sleep time or executes at a higher v/f state because the average power of solution schedules can be bigger. Hereby, the optimal latency to the TAP_{min} problem monotonically

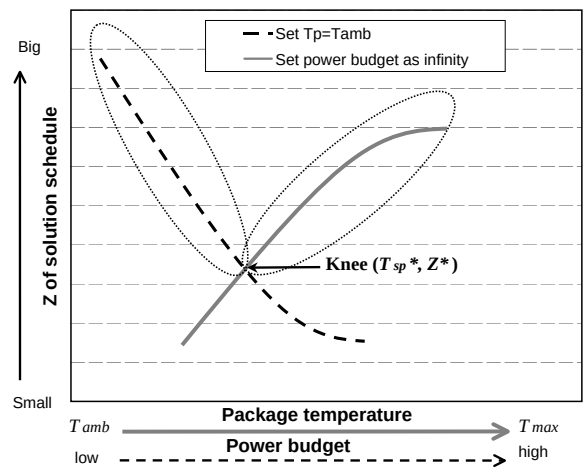


Figure 8.3: Illustration of the knee with the optimal Z^* and optimal T^*_{sp}

$TA-OPT(|J'|)/*TA-FPTAS(\varepsilon,|J'|)*/$:

```
0     set T_LB = T_amb and T_UB = T_max;
1     set Z* = ∞, S* = NULL;
2     set Z = 0, T_sp = T_UB;
3     do {
4       if (TAP − OPT(T_sp, |J'|)
          returns success with Z and S){
/* 4    if (TAP − FPTAS(ε, T_sp, |J'|)
          returns success with Z and S){ */
5         if (Z* ≥ Z) {record Z* = Z, S* = S };
6         else { break; }}
7       T_sp = T_sp − 1; }
8     while (T_sp > T_LB);
9     return Z*,S*;
```

Figure 8.4: Optimal algorithm for TA_{min} (Specifications in /**/ are the modifications for the approximation algorithm $TA-FPTAS$ procedure

decreases as the power budget increases. The two scenarios cross at a knee with package temperature setting T^*_{sp} and solution Z^*. The solutions to the original TAP_{min} problem exist in the upper half of the black dotted line ($T_{sp} \leq T^*_{sp}$) and the upper half of the gray full line ($T_{sp} \geq T^*_{sp}$) both to the knee. At the knee with T^*_{sp}, the optimal latency to the TAP_{min} problem is the smallest among all the solutions to the TAP_{min} problems with possible package temperature settings. Thus, the optimal latency Z^* to the TAP_{min} problem at the knee is the optimal to the TA_{min} problem. We further verify the property with experiments in later section 8.6. Next we provide the search algorithm.

Figure 8.4 depicts the search algorithm for the optimal package temperature setting T^*_{sp} and the optimal solution Z^*. The search range is between T_{amb} and T_{max}. Line 0 sets the upper bound and

lower bound of the package temperature setting. Lines 3-8 executes the search based on the $TAP-OPT$ procedure with input T_{sp}. Line 2 sets the package temperature setting T_{sp} as the upper bound of package temperature T_{max}. Line 4 triggers $TAP-OPT$ procedure with T_{sp} as input. If $TAP-OPT$ returns success in Line 4, Lines 5-6 record the current solution and determines whether the optimal solution is found. Based on Lemma 8.4.2 we seek the knee in the range of T_{sp} from the upper bound. According to the lemma, the solution latency Z should first decrease then increase as the test T_{sp} value decreases. Line 5 compares the current latency Z with the previous recorded latency Z^*. Until the previous recorded latency Z^* is less than current latency Z, we stop the search because the previous recorded solution is the knee. Z^* is the final solution for the TA_{min} problem when we stop the search. If search continues, Line 8 increments the next package temperature setting. Lines 9 returns the solution according to the search result.

We denote the range of package temperature settings at the granularity of $1C^{\circ}$ as a constant L. The computational complexity of the optimal algorithm $TA-OPT$ is L times the computational complexity of the $TAP-OPT$ algorithm. In the later section we show that the computation complexity of the $TAP-OPT$ algorithm is pseudo-polynomial time. To reduce the runtime of the solution technique, we further provide a polynomial time algorithm based on the FPTAS in Section 8.5 for the TAP_{min} problem. Next we present the FPTAS based algorithm.

FPTAS based algorithm

We modify the optimal algorithm to an FPTAS based algorithm by replacing the test procedure $TAP-OPT$ with the FPTAS $TAP-FPTAS$ presented in Section 8.5 for the subproblem TAP_{min}. The segmentation /**/ in Figure 8.4 describes the algorithm $TA-FPTAS$. $TA-FPTAS$ requires a designer-specified quality bound ε as input. The ε is an input to the test procedure $TAP-FPTAS$ that can produce quality-guaranteed solutions for the subproblem TAP_{min} at each package temperature setting T_{sp}.

In the $TA-FPTAS$ described in Figure 8.4 similar to $TA-OPT$ algorithm, we utilize the search algorithm to search for the knee with the steady state package temperature setting that leads to the smallest latency. The search algorithm triggers the $TAP-FPTAS$ with ε instead of $TAP-OPT$ procedure to test each package temperature setting. Finally it finds the knee that leads to the smallest latency. The smallest latency is the final solution.

The test procedure $TAP-FPTAS$ in Line 4 of Figure 8.4 is able to achieve an approximated solution for a given steady state package temperature T_{sp} and a given quality bound ε. As proved in Section 8.5, $TAP-FPTAS$ with ε is a bi-criteria approximation algorithm that generates solutions within

proved bounds. Because $TAP-FPTAS$ is fully polynomial time technique, the run time of $TA-FPTAS$ is polynomial.

In the next section, we describe the TAP_{min} problem and provide the $TAP-OPT$ and $TAP-FPTAS$ techniques as solutions. The techniques $TAP-OPT$ and $TAP-FPTAS$ are utilized as the test procedures as part of the techniques $TA-OPT$ and $TA-FPTAS$ for the TA_{min} problem.

8.5 TA_{min} for job sequence with power budget constraint

In this section, we consider the subproblem TAP_{min} as the TA_{min} problem with a power budget constraint when package temperature is in steady state. The problem description is similar to the TA_{min} problem except that the solution schedule is constrained by a power budget and a steady state package temperature setting. During the job execution, the package temperature remains the same as the steady state package temperature setting (say T_{sp}) because the job execution time is much shorter comparing to the package temperature time constant. The average power of the solution schedule is constrained by the associated power budget ρ_{av} to T_{sp} calculated from Equation 8.10.

Similar to the formulation of TA_{min}, we formulate the subproblem for a job sequence J' integrating sleep jobs and active jobs as follows.

$$TAP_{min} : \min Z = \sum_{i=1}^{2n+1} \sum_{j=1}^{r} t_{ij} x_{ij}$$

$$subject\ to\ C_d \frac{dT(t)}{dt} = -\frac{T(t)}{R'_d} + (\frac{T_p(t)}{R_d} + \beta + \rho(t)); \tag{8.11a}$$

$$E = \sum_{i=1}^{2n+1} \sum_{j=1}^{r} e_{ij} x_{ij} \leq \rho_{av} \cdot Z; \tag{8.11b}$$

$$\sum_{j=1}^{r} x_{ij} = 1, \forall i \in [1, 2n+1]; \tag{8.11c}$$

$$T_d(t=0) = T_{max}; T_p(t) = T_{sp}; T_d(t) \leq T_{max}; \tag{8.11d}$$

$$x_{ij} = \{0, 1\}; \tag{8.11e}$$

The objective of the TAP_{min} problem is to minimize the execution time of one run of the job sequence including total $N = 2n+1$ sleep and active jobs. x_{ij} represents that job J'_i is executed at the j_{th} power level or option. t_{ij} is provided as the execution time of J'_i at the j_{th} option. Constraint 8.11a specifies that the die temperature in the TAP_{min} problem follows Equation 8.3, which is a decoupled die temperature thermal model. We assume the package temperature remains the same as the initial package temperature T_{sp} specified in Constraint 8.11d. The starting die temperature is initialized as the peak temperature limit T_{max}. Constraint 8.11b specifies the average power of the solution schedule is no more than the power budget ρ_{av} calculated from T_{sp}. We specify the power budget constraint in the form of energy constraint. The energy consumption E of the solution schedule is no more than the power budget ρ_{av} times the

execution time Z. e_{ij} is provided as the energy consumption of J'_i at the j_{th} option. Similarly constraint 8.11c specifies that only one v/f level is selected for each job.

The TAP_{min} problem is NP-hard because a special case of TAP_{min} has been proved to be NP-hard in the proof of Theorem 8.3.1. The NP-hard problem addressed in the proof of Theorem 8.3.1 is a special case of the TAP_{min} problem without considering power budget constraint and assuming that the package temperature is the ambient temperature. Since the special case of TAP_{min} is NP-hard, TAP_{min} is NP-hard. Next, we provide the optimal algorithm and the approximation algorithm as solutions to the TAP_{min} problem.

Optimal algorithm for TAP_{min}

Overview

The optimal algorithm is based on a dynamic programming (DP) approach that runs in pseudo-polynomial time similar to the knapsack problem [96] . However, TAP_{min} is differentiated from the knapsack problem because the problem includes multiple constraints, especially the non-linear thermal constraint. The central idea of the DP originates from the following property of the problem.

Lemma 8.5.1. *Consider an optimal solution S^* for the problem that executes the job sequence in Z^* time. Let the optimal solution finish execution for the first i jobs in J' with execution time Z and energy consumption E. Then a partial solution $S'_{1:i}$ that minimizes the final die temperature after executing the first i jobs of J' in exactly Z time and exactly E energy can be utilized to generate a complete solution that executes in Z^* time (same as the optimal solution).*

Proof. The final die temperature after the execution of the first i jobs with exactly Z time and E energy by S^* and $S'_{1:i}$ are differently denoted as $T_d^{S^*}$ and $T_d^{S'_{1:i}}$ with $T_d^{S'_{1:i}} \leq T_d^{S^*}$. We construct a full schedule S' with the partial schedule $S'_{1:i}$ and the remainder of the optimal schedule S^* for jobs $(J'_{i+1},...,J'_N)$. According to the die temperature thermal model for executing one iteration of the job sequence, the remainder of the optimal schedule S^* for jobs $(J'_{i+1},...,J'_N)$ starting from $T_d^{S^*}$ is feasible under thermal constraint when starting from $T_d^{S'_{1:i}}$. Thus, S' is feasible under thermal constraint. Furthermore, S' consumes identical execution time Z^* and identical energy consumption as those of S^*. Consequently, $S'_{1:i}$ leads to a solution that executes by Z^*. □

Our DP incrementally generates a schedule that minimizes the final die temperature for all possible jobs i with total execution time Z and total energy consumption E. Then, the smallest Z value

with $i = N$ under energy constraint and the associated traceback schedule are obtained as the solution for the overall problem. Let $T(i,Z,E)$ be the minimum final die temperature, when the first i jobs are executed in exactly Z time and exactly E energy. In the DP algorithm, $T(i,Z,E)$ is minimized subject to T_{max} for $i \in \{1,2,3,\ldots,2n+1\}$, $Z \in [1,Z_{UB}]$ and $E \in [1,E_{UB}]$ where Z_{UB} is an upper bound on the optimal value of Z and E_{UB} is an upper bound on the energy of a feasible schedule. Let Z^* denote the optimal value. Z^* is determined by the smallest value of Z such that $T(2n+1,Z,E) \leq T_{max}$ and $E \leq \rho_{av}Z$.

Calculation of Z_{UB} and E_{UB}

Z_{UB} can be calculated by considering a schedule S_{init} as follows. Given an initial die temperature T_{max} and an initial package temperature T_{sp}, the processor first sleeps such that $T_d \approx T_{sp}$. Then the processor executes the first active job at the highest voltage (fastest frequency) that does not violate the temperature constraint T_{max}. Let T_1 denote the die temperature at the end of execution of job J_1. Next the processor again sleeps for some time such that the die temperature reduces to T_{sp} from T_1. Then it executes the second active job at the highest voltage that does not violate the temperature constraint and again sleeps till the die temperature is equal to T_{sp}. The processor repeats the execution pattern for all jobs.

For example, suppose there is an active job sequence $J = \{J_1,J_2,J_3\}$ with S_{init}. The package temperature is $55°C$ and the peak temperature is $100°C$. The corresponding task sequence J' becomes $\{J_s,J_1,J_s,J_2,J_s,J_3,J_s\ \}$. In S_{init}, the first sleep job J_s is executed such that temperature reaches $55°C$[3]. Then, J_1 is executed at an available v/f state as fast as possible such that peak temperature remains below T_{max}. S_{init} repeats the execution pattern for J_2 and J_3 until the last J_s is executed to reach $55°C$. Clearly, such a schedule is feasible under thermal constraint. Therefore the latency by the schedule is a valid upper bound Z_{UB} on Z^*.

E_{UB} can be calculated from Z_{UB} and the average power budget ρ_{av} for the package temperature setting T_{sp} achieved from Equation 8.8. E_{UB} is given by $Z_{UB} * \rho_{av}$.

Dynamic programming algorithm

Let $S_{i,Z,E}$ be the schedule with $T(i,Z,E)$. If $S_{i,Z,E}$ does not exist, we define $T(i,Z,E) = \infty$. Set $T(0,Z,E) = T_{max}$ for $Z \in [1,...,Z_{UB}]$ and $E \in [1,...,E_{UB}]$. We set $T(1,0,0) = T_{max}$, because the first job is a sleep job and it can have zero sleep time and zero energy. The recurrence relation for the DP

[3] According to the thermal model in a sleep mode the temperature will approach T_{sp} asymptotically. Therefore, in practice for a sleep mode we consider the time required for the temperature to fall reasonably close (say within 1%) of T_{sp}.

$TAP-OPT(T_{sp},N)$/*$TAP-OPT_m(N,Z_{UB},t'_{ij},E_{UB},e'_{ij})$*/

```
0   set T(i,Z,E) = ∞(∀i = 1 : N, ∀Z = 1 : Z_UB, ∀E = 1 : E_UB);
1   set T(0,Z,E) = T_max(∀Z = 1 : Z_UB, ∀E = 1 : E_UB);
2   for i = 1 : N {
3     for Z = 1 : Z_UB {
4     for E = 1 : E_UB {
5         T_min = ∞;
6         for j = 1 : r {
7           calculate T_h = T(i−1, Z−t_ij, E−e_ij) + ΔT(s_j);
/*7        calculate T_h = T(i−1, Z−t'_ij, E−e'_ij) + ΔT(s_j);*/
8           if (T_h ≤ T_max) and (T_h < T_min), set T_min = T_h and j_h = j; }
9         fill T_min in cell (i,Z,E) as T(i,Z,E) and record j_h; } } }
10  find the smallest Z* with T(N,Z,E) ≤ T_max and E ≤ Z * ρ_av;
/*10  find the smallest Z* with T(N,Z,E) ≤ T_max and E ≤ Z+N;*/
11  if found, trace back and return success, Z* and S*;
12  else return failure;
```

Figure 8.5: Optimal algorithm for the TAP_{min} (Specifications in /* */ are the modifications for the $TAP-OPT_m$ procedure invoked by approximation algorithm in Section 8.5)

algorithm is given by:

$$T(i,Z,E) = \min_{j\in[1,r]} \{T(i-1,Z-t_{ij},E-e_{ij}) + \Delta T(s_j) | T \leq T_{max}\} \tag{8.12}$$

The non-linear decoupled die temperature equation (Equation 8.3) is utilized to achieve the die temperature change (denoted by $\Delta T(s_j)$) due to a particular sleep option or an active job execution. From the recurrence, we can find $T(N,Z,E)$, for all $Z \in [1,Z_{UB}]$ and $E \in [1,E_{UB}]$. The optimal solution is then $S_{N,Z^*,E}$ (denoted by S^* in the remainder of the work), where

$$Z^* = \min\{Z | T(N,Z,E) \leq T_{max}, E \leq \rho_{av} * Z\} \tag{8.13}$$

The recurrence relation leads to an algorithm $TAP-OPT(T_{sp},N)$ in Figure 8.5 that constructs a 3-dimension DP table (refer to Lines 2-9 in Figure 8.5). The x index of the table represents the sequence of $N = 2n+1$ jobs (including both active jobs and sleep jobs). The y index represents all possible objective values $Z \in [1,Z_{UB}]$. And the z index represents all possible energy consumption values $E \in [1,E_{UB}]$. Each cell has an entry of the minimum final die temperature when Z time and E energy are spent, and the first i jobs are finished. Further, each cell (i,Z,E) also has an entry for the time t_{ij} and the energy e_{ij} associated with sleep or active state s_j that generates the minimum final die temperature value. The t_{ij} and e_{ij} values will be essential for tracing back the final solution. The table is constructed in the order of x index increasing. Thus, after the algorithm enters the cell (i,Z,E), the cells for all the x index smaller than i are filled in. And, the previous cells with $1:Z-1$, $1:E-1$ and the ith

index are also filled in. The algorithm need not re-calculate the optimal solution for a given subproblem $T(i-1, Z-t_{ij}, E-e_{ij})$. For each cell, r calculations are needed to find the minimum final temperature. Once the algorithm finds Z^*, the associated schedule, denoted by S^*, is achieved by tracing back in the solution table from $(2n+1)$ to 1. This can be easily implemented by $2n+1$ table lookups.

Figure 8.5 describes the pseudo-codes of dynamic programming algorithm $TAP-OPT$ (Specifications in /* */ are the modifications for the $TAP-OPT_m$ procedure invoked by approximation algorithm in Section 8.5). N and T_{sp} are the inputs representing the total number of the new job sequence and the steady state package temperature setting. From the jobs and T_{sp} we derive the upper bound of the optimal Z^* and the upper bound of energy consumption. Lines 0-1 initialize the DP table. Lines 2-9 construct the DP table by Equation 8.12. Lines 10-12 find the optimal Z^* by Equation 8.13 and return the optimal schedule, or return failure if no feasible solution exists.

Computational complexity analysis

The computational complexity of the DP algorithm is pseudo-polynomial. For each cell, it needs $O(r)$ computations (Lines 5-9 in Figure 8.5). The algorithm has $O((2n+1) \cdot Z_{UB} \cdot E_{UB})$ iterations to fill in the cells (Lines 2-3 in Figure 8.5). Thus, the computation complexity is $O(rn \cdot Z_{UB} \cdot E_{UB})$.

Proofs of optimality

Next we prove the optimality of the proposed algorithm.

Lemma 8.5.2. *Given N, Z and E the recurrence relation $T(N,Z,E)$ (defined by Equation 8.12) gives the lowest die temperature when N jobs are executed in exactly Z time and exactly E energy if a feasible solution exists.*

Proof. We prove by induction.

- For $N=1$, J' only has one job (sleep job) and thus there always exists a feasible solution. For any given Z, E and by Equation 8.12, $T(1,Z,E)$ is equal to $min_{s_j \in [1,r]}\{T_{max} + \Delta T(s_j)\}$. s_j is chosen from q choices in sleep state. It is clear that Equation 8.12 gives the lowest final die temperature for $N=1$ job executed in exactly Z time and exactly E energy.
- Suppose that, for $N=i$ and any given Z, E, $T(i,Z,E)$ is either the lowest final die temperature for the first $N=i$ jobs with exactly Z execution time and exactly E energy subject to thermal constraint, which is given by Equation 8.12. Or, $T(i,Z,E)$ remains the initialized value as infinity if no feasible solution exists.

- Let us consider $N = i+1$. For any given Z and E and by Equation 8.12, when processor chooses the j_{th} state to execute J'_{i+1}, $T(i+1,Z,E)$ is given by

ch

$$T_j(i+1,Z,E) = \{T(i,Z-t_{(i+1)j},E-e_{(i+1)j}) + \Delta T(s_j) | T \leq T_{max}\}$$

Therefore, there are two cases for $T_j(i+1,Z,E)$:

  - Case I ($T(i,Z-t_{(i+1)j},E-e_{(i+1)j})$ is infinity): $T_j(i+1,Z,E)$ is also infinity, because there is no feasible solution for the first i jobs executed in exactly $Z-t_{(i+1)j}$ time and exactly $E-e_{(i+1)j}$ energy subject to thermal constraint.
  - Case II ($T(i,Z-t_{(i+1)j},E-e_{(i+1)j})$ is not infinity): By thermal model, the lower starting die temperature leads to the lower final die temperature for the $i+1_{th}$ job. Thus, by induction hypothesis, $T_j(i+1,Z,E)$ achieves the lowest final die temperature for the first $(i+1)$ jobs subject to thermal constraint.

Thus, $T_j(i+1,Z,E)$ minimizes the final die temperature for the first $(i+1)$ jobs in exactly Z execution time and exactly E energy. Then, Equation 8.12 enumerates all possible j states for J'_{i+1}. Therefore, for any given Z, E, Equation 8.12 gives the lowest final die temperature such that the first $N = i+1$ jobs are executed in exactly Z time and exactly E energy.

□

Theorem 8.5.1. *Our dynamic programming algorithm generates optimal solutions for TAP_{min}.*

Proof. By Lemma 8.5.2, for a given Z and a given E, Equation 8.12 calculates the lowest final die temperature for N jobs with exactly Z execution time and exactly E energy. And it finally generates a feasible solution under thermal constraint (if it exists). By Lemma 8.5.1, given the optimal value Z^* for N jobs, Equation 8.12 always finds a feasible solution, and the feasible solution does not preclude optimal solutions for TAP_{min}. Therefore, given Z^*, the algorithm is able to generate a feasible solution for N jobs, which is an optimal solution for the problem. Further, Equation 8.13 enumerates all possible Z values and E values for the N jobs and picks the smallest Z value with feasible solution as the result. Therefore, our algorithm generates optimal solutions for TAP_{min}. □

$(1+\varepsilon)$ *FPTAS for* TAP_{min}

Overview

The DP algorithm for the optimal solution is not polynomial due to the factor Z_{UB} and E_{UB} in the computational complexity which could be exponential in the input size of the problem. We now develop a bi-criteria fully polynomial time approximation scheme (FPTAS) for TA_{min}. A FPTAS is an approximation algorithm whose run time complexity is bounded by a polynomial in the input size of the problem and $(1/\varepsilon)$. A FPTAS is the best one can hope for a NP-hard optimization problem [96]. A bi-criteria approximation algorithm is an algorithm with quality bounds (η, ζ) for the problem, where η and ζ are constants. If there exists a feasible solution, our algorithm finds a feasible schedule such that the total execution time is no more than ηZ^* when the energy constraint is relaxed to $\zeta \cdot \rho_{av} Z^*$. The proposed bi-criteria FPTAS generates schedules whose execution time is guaranteed to be no more than $(1+\varepsilon)Z^*$ and whose energy consumption is guaranteed to be no more than $(1+2\varepsilon)\rho_{av}Z^*$, where ε (typically $0 < \varepsilon \leq 1$) is a designer specified quality bound.

The approximation algorithm works by scaling and reducing the search space[4] for the optimal Z^*. The algorithm utilizes a probe procedure to test a possible optimal value Z in a search space for the optimal. The probe procedure can fail or succeed on a testing value Z. The failure/success result can be used to adjust the upper or lower bound of the search space for the optimal. The search space is iteratively narrowed down by repetitive invocation of the probe procedure until the ratio between the upper and lower bounds of the optimal is a constant. Then, the algorithm invokes an approximation procedure to get the approximated result.

Approximation algorithm

The algorithm is described in Figure 8.6. The main algorithm is the $TAP_{min} - FPTAS(\varepsilon, T_{sp}, N)$. Initially, the algorithm finds the search space $[Z_{LB}, Z_{UB}]$ for Z^*. As described earlier Z_{UB} can be calculated from S_{init}. Z_{LB} can also be estimated from S_{init} by summation of the execution time of the jobs in the active state. Let $t_{i,init}$ denote the execution time for a job J_i' (i is even) in the active state for the schedule S_{init}. Thus, $Z_{LB} = \sum_{J_i' \in J} t_{i,init}$. The algorithm then narrows down the search space by probing the scaled problem in lines 2 to 5. Here, $probe(Z)$ acts as a test procedure that returns success if the scaled problem has a feasible schedule, otherwise returns failure. The search procedure continues until the solution

[4]Our approximation scheme parallels the FPTAS for the restricted shortest path problem [57]. However, TAP_{min} is distinctly different from the restricted shortest path problem due to the non-linear thermal constraints and the power budget constraint. Thus, although there are some similarities in the solution approaches, the problem formulation and proofs are different.

$TAP-FPTAS(\varepsilon, T_{sp}, N)$:

```
initially get Z_LB and Z_UB;
Z_UB = Z_UB/3;
while (Z_UB ≥ 2·Z_LB)
   { let Z = √(Z_LB · Z_UB);
   if probe(Z) = failure, Z_LB = Z;
   else Z_UB = Z; /* probe(Z) = success */}
Z_f = TAPapprox(3·Z_UB, Z_LB, ε);
return Z_f;
```

probe(Z):

```
   set K_Z = Z/N; t'_ij = ⌊t_ij/K_Z⌋; Z' = ⌊Z/K_Z⌋ + N;
   set K_E = Z*ρ_av/N; e'_ij = ⌊e_ij/K_E⌋; E' = N;
   return TAP-OPT_m(N, Z', t'_ij, E', e'_ij);
```

$TAPapprox(UB, LB, \varepsilon)$:

```
set K_Z = ε·LB/N; t'_ij = ⌈t_ij/K_Z⌉; Z' = ⌈UB/K_Z⌉ + N;
set K_E = ε·LB·ρ_av/N; e'_ij = ⌈e_ij/K_E⌉; E' = ⌈UB/K_Z⌉ + N;
return Z_f = TAP-OPT_m(N, Z', t'_ij, E', e'_ij);
```

Figure 8.6: A FPTAS for the TAP_{min}

space is narrowed down to $[Z_{LB}, 6Z_{LB}]$. Finally, $TAPapprox(UB, LB, \varepsilon)$ is invoked that returns an $(1+\varepsilon)$ approximated result. In both *probe* and $TAPapprox$, the $TAP-OPT_m$ procedure in Figure 8.5 is utilized, which is similar to our proposed $TAP-OPT$ procedure with the recurrence equation 8.12 and the optimal equation 8.13. The only difference to the $TAP-OPT$ procedure with equation 8.12 is that the scaled values (Z', t'_{ij} and e'_{ij}) are utilized when searching for $T(i-1, Z-t_{ij}, E-e_{ij})$ (Line 7 in Figure 8.5). However, non-scaled values of t_{ij} (that is, the original values of t_{ij}) are utilized for calculation of ΔT. Thus, the feasible solution for the scaled problem is feasible for the non-scaled problem with the thermal constraint and vice versa, because the temperature calculation is made with the non-scaled time values. The only difference to the $TAP-OPT$ procedure with equation 8.13 is that the scaled value of total energy consumption is constrained by the scaled Z value plus N (Line 10 in Figure 8.5). This is because the scaling factor K_E for energy is ρ_{av} times K_Z. Thus we utilize a scaled energy constraint to replace the non-scaled one. This leads to the energy constraint relaxation in the final solution by the approximation algorithm.

Proofs for FPTAS

Next, we prove $TAP-FPTAS$ is an $(1+\varepsilon)$ FPTAS.

Let the $t_{min} = \min_{\forall J_i \in J, s_j \in M}\{t_{ij}\}$ denote the minimum execution time of any job in an active

state. Let $\theta = t_{ms}/t_{min}$. Recall that t_{ms} is the maximum cooling transient time for the processor to cool from T_{max} to T_{sp} in sleep mode. In the schedule S_{init} let θ_i denote the ratio between the sleep time preceding the active job J_i' ($J_i' \in J$) and the execution time of the job $t_{i,init}$. It is clear that $\theta \geq \theta_i$. Thus, we have

$$Z_{UB} \leq t_{ms} + \sum_{J_i' \in J} (\theta_i + 1) t_{i,init} \leq \theta t_{min} + (\theta + 1) \sum_{J_i' \in J} t_{i,init} \leq (2\theta + 1) Z_{LB}$$

The first inequality follows from S_{init} that processor first sleeps for time t_{ms} before executing all the active jobs. The second inequality follows from $\theta \geq \theta_i$. The last inequality follows from $t_{min} \leq Z_{LB}$. Thus, initially $Z_{UB}/Z_{LB} \leq 2\theta + 1$ and $Z^* \in [Z_{LB}, Z_{UB}]$. Because Z_{UB} is initialized as $Z_{UB}/3$ before entering the while loop of the $TAP - FPTAS(\varepsilon)$ procedure, inside the while loop $Z_{UB}/Z_{LB} \leq \frac{2\theta+1}{3}$.

Lemma 8.5.3. *If $probe(Z)$ returns failure, $Z^* > Z$.*

Proof. We prove it by contradiction. Suppose that $Z^* \leq Z$ and $probe(Z)$ returns failure.

We first show that the latency of the S^* in the scaled problem is no more than the searching Z' upper bound. Note that Z^* is the optimal execution time for the original problem and S^* is the associated optimal schedule. Recall that a feasible schedule of the original problem is also a feasible schedule in the scaled problem under thermal constraint. Thus, S^* is still feasible for the scaled version of the problem. Denote $Z'(S^*)$ as the optimal for the scaled version of problem with S^*.

$$Z'(S^*) = \sum_{S^*} \lfloor \frac{t_{ij}}{K_Z} \rfloor \leq \sum_{S^*} \frac{t_{ij}}{K_Z} = \frac{Z^*}{K_Z} \leq \frac{Z}{K_Z} \leq \lfloor \frac{Z}{K_Z} \rfloor + N$$

In the first equation according to the objective function, $Z'(S^*)$ can be represented by the summation of scaled execution time for N jobs with S^*. The second inequality follows from the inequality property of floor operation. The third equality follows from $Z^* = \sum_{S^*} t_{ij}$. Because of $Z^* \leq Z$, the fourth inequality holds true. The last inequality follows from the property of floor operation. Then, $Z'(S^*)$ is no more than the upper bound of the search in $probe(Z)$.

Then we show that the energy consumption of the S^* in the scaled problem (denoted as $E'(S^*)$) is no more than the searching upper bound of energy E'.

$$E'(S^*) = \sum_{S^*} \lfloor \frac{e_{ij}}{K_E} \rfloor \leq \sum_{S^*} \frac{e_{ij}}{K_E} = \frac{\sum_{S^*} e_{ij}}{K_E} \leq \frac{\rho_{av} * Z^*}{K_E} = N \frac{Z^*}{Z} \leq N$$

In the first equation, $E'(S^*)$ is represented by the summation of scaled energy consumption time for N jobs with S^*. The first inequality follows from the inequality property of floor operation. The second inequality follows from the energy constraint for Z^*. The third equality follows from the definition of Z^*.

Since the upper bound of E in the scaled problem is more than E', S^* in the scaled problem should satisfy the energy constraint. Since the upper bound of Z in the scaled problem is more than Z', S^* in the scaled problem should be found. So, *probe* succeeds on Z with S^*. This is contradiction to the assumption that *probe* would fail on Z. □

Lemma 8.5.4. *If probe(Z) returns success, $Z^* \leq 3 \cdot Z$.*

Proof. Because the *probe* procedure succeeds, there is at least one feasible schedule S with the scaled problem such that

$$Z'(S) \leq \lfloor \frac{Z}{K_Z} \rfloor + N \leq \frac{Z}{K_Z} + N \tag{8.14}$$

Also,

$$Z'(S) = \sum_S \lfloor \frac{t_{ij}}{K_Z} \rfloor \geq \sum_S \frac{t_{ij}}{K_Z} - N \geq \frac{Z^*}{K} - N \tag{8.15}$$

The first inequality follows from $\lfloor \frac{t_{ij}}{K_Z} \rfloor \geq \frac{t_{ij}}{K_Z} - 1$. The second inequality follows from that Z^* is the optimal in the original problem. The following inequality follows from Equations 8.14 and 8.15, and the definition of K_Z in *probe*(Z):

$$Z^* - NK_Z \leq Z + NK_Z \Rightarrow Z^* \leq 3 \cdot Z \tag{8.16}$$

□

Lemma 8.5.5. *If $LB \leq Z^* \leq UB$, TAPapprox(UB, LB, ε) succeeds and returns $Z_f \leq (1+\varepsilon)Z^*$ subject to $(1+2\varepsilon)$ relaxation of energy constraint.*

Proof. We first show that *TAPapprox*(UB, LB, ε) succeeds if $LB \leq Z^* \leq UB$. We have two steps to justify it.

i. We show that the scaled Z value of S^* (denoted by $Z'(S^*)$) is smaller than the upper bound of search space.

$$Z'(S^*) = \sum_{S^*} \lceil \frac{t_{ij}}{K_Z} \rceil \leq \sum_{S^*} \frac{t_{ij}}{K_Z} + N \leq \frac{UB}{K_Z} + N \tag{8.17}$$

The first and second equations follow from the definition of time value scaling in the *TAPapprox*. The third equation follows from the assumption $Z^* \leq UB$. Therefore, $Z'(S^*$ is smaller than $\lceil \frac{UB}{K_Z} \rceil + N$, which is the upper bound of search space.

ii. We show that the scaled E value of S^* (denoted by $E'(S^*)$) is smaller than the upper bound of search space and satisfies the scaled energy constraint.

$$E'(S^*) = \sum_{S^*} \lceil \frac{e_{ij}}{K_E} \rceil \leq \sum_{S^*} \frac{e_{ij}}{K_E} + N \leq \frac{\rho_{av} Z^*}{K_E} + N = \frac{Z^*}{K_Z} + N \tag{8.18}$$

The first and second equations follow from the definition of energy value scaling in the *TAPapprox*. The third equation follows from the energy constraint of S^* in the original problem. The fourth equation follows from $K_E = K_Z * \rho_{av}$. Therefore, the scaled E value of S^* is smaller than $\lceil \frac{UB}{K_Z} \rceil + N$, which is the upper bound of search space. And, $E'(S^*)$ is smaller than $Z'(S^*) + N$ because of upper rounding of t_{ij} values. Thus, S^* in the scaled problem satisfies the scaled energy constraint.

Since S^* in the scaled problem are within the search space, it satisfies the scaled energy constraint and the non-scaled thermal constraint, *TAPapprox* succeeds.

Next, we show that *TAPapprox* returns a succeeded solution S^+. S^+ has execution time Z_f in the non-scaled problem with $Z_f \leq (1+\varepsilon)Z^*$ subject to $(1+2\varepsilon)$ relaxation of energy constraint. We also denote the total execution time of S^+ in the non-scaled problem by $E(S^+)$.

By the design of *TAPapprox*, S^+ is the optimal schedule in the scaled problem and S^* is a feasible schedule in the scaled problem. We have

$$Z_f = K_Z \sum_{S^+} t'_{ij} \leq K_Z \sum_{S^*} t'_{ij} \leq \sum_{S^*} t_{ij} + NK_Z = Z^* + \varepsilon LB \leq Z^*(1+\varepsilon) \tag{8.19}$$

The first inequality follows from the fact that optimal schedule S^* is a feasible solution for the scaled version of the problem, and the optimal schedule S^+ in the scaled problem would achieve execution time no more than that with S^*. The second inequality follows from $K_Z t'_{ij} \leq t_{ij} + K_Z$, when t_{ij} is rounded up. The third inequality follows from $LB \leq Z^*$. Therefore $Z_f \leq Z^*(1+\varepsilon)$.

We denote the the total execution time and the total energy consumption of S^+ in the scaled problem by $Z'(S^+)$ and $E'(S^+)$. We then show that S^+ generated by *TAPapprox* is feasible under the $1+2\varepsilon$ relaxation of energy consumption constraint. We have two steps to justify it.

i. We first seek the upper bound of $E'(S^+)$ in the scaled problem.

$$E'(S^+) \leq Z'(S^+) + N \leq \sum_{S^*} t'_{ij} + N \leq \frac{Z^*}{K_Z}(1+\varepsilon) + N \tag{8.20}$$

The first inequality follows from the scaled energy constraint in the scaled problem (Line 10 in Figure 8.5 for $TAP-OPT_m$ procedure). The second inequality follows from that S^+ is the optimal schedule in the scaled problem. The third inequality has been proved in the equation 8.19(see the third and sixth items).

ii. Then we seek the upper bound of $E(S^+)$ in the non-scaled problem.

$$E(S^+) = \sum_{S^+} e_{ij} \leq K_E E'(S^+) \leq K_E\left(\frac{Z^*(1+\varepsilon)}{K_Z} + N\right) \tag{8.21}$$

$$= (1+\varepsilon)\rho_{av}Z^* + \varepsilon LB\rho_{av} \leq (1+2\varepsilon)\rho_{av}Z^* \leq (1+2\varepsilon)\rho_{av}Z_f \tag{8.22}$$

The first equation follows from the ceil operation to scale e_{ij} in $TAPapprox$. The second inequality follows from Equation 8.20. The last equation is derived due to $\frac{K_E}{K_Z} = \rho_{av}$. The third inequality follows $LB \leq Z^*$. The last inequality follows that Z^* is the optimal in the non-scaled problem. Since the energy constraint of S^+ is $\rho_{av} Z_f$, S^+ is feasible under $1+2\varepsilon$ relaxation of the energy constraint.

□

Lemma 8.5.6. *$TAP-FPTAS$ generates an $(1+\varepsilon)$ approximation schedule subject to $(1+2\varepsilon)$ relaxation of energy constraint.*

Proof. By the Lemma 8.5.3, 8.5.4 and the algorithm, we have the following equation in the kth iteration of the while loop.

$$Z_{LB}^{[k]} \leq Z^* \leq 3 \cdot Z_{UB}^{[k]} \tag{8.23}$$

In the line 6 of $TAPapprox$, $Z_{UB} < 2 \cdot Z_{LB}$. In the input of TAapprox, $Z_{LB} \leq Z^* \leq 3 \cdot Z_{UB} < 6 \cdot Z_{LB}$. By the Lemma 8.5.5, $TAPapprox$ generates an $(1+\varepsilon)$ approximation schedule subject to $1+2\varepsilon$ relaxation of energy consumption. □

Lemma 8.5.7. *The complexity of $TAP-FPTAS(\varepsilon)$ is $O(\frac{n^3 r}{\varepsilon} + n^3 r \log\log\theta)$.*

Proof. In the line 0 of the $TAP-FPTAS$, the complexity is $O(nr)$. In the *probe*, the complexity is $O(n^3 r)$, because Z is scaled by $K_Z = \frac{Z}{N}$ and E is scaled by $K_E = \frac{\rho_{av} Z}{N}$. In the $TAPapprox$, the complexity is $O(\frac{n^3 r}{\varepsilon})$, because Z is scaled by $K_Z = \frac{\varepsilon Z_{LB}}{N}$, E is scaled by $K_E = \frac{\varepsilon \rho_{av} Z_{LB}}{N}$ and $Z_{LB} \leq 3 \cdot Z_{UB} < 6 \cdot Z_{LB}$ in Line 6. Now the complexity from line 2 to line 5 is critical for the whole complexity.

In the $(k+1)^{th}$ iteration of the while loop, we always have $\frac{Z_{UB}^{[k+1]}}{Z_{LB}^{[k+1]}} = \sqrt{(\frac{Z_{UB}^{[k]}}{Z_{LB}^{[k]}})} = (\frac{Z_{UB}^{[k]}}{Z_{LB}^{[k]}})^{\frac{1}{2}}$. Recall that the while loop works only when $Z_{UB} \geq 2 \cdot Z_{LB}$. Let the number of iterations be p. We obtain an upper bound on p with the following equation:

$$\frac{Z_{UB}^{[p]}}{Z_{LB}^{[p]}} = (\frac{Z_{UB}^{[0]}}{Z_{LB}^{[0]}})^{(\frac{1}{2})^p} \geq 2 \tag{8.24}$$

As due to line 1 in $TAP-FPTAS\varepsilon$) we initially have $\frac{Z_{UB}^{[0]}}{Z_{LB}^{[0]}} = \frac{2\theta+1}{3}$, p is no more than $O(\log\log\theta)$. So, the complexity of line 2 to 5 is $O(n^3 r \log\log\theta)$. Thus, the overall complexity is $O(\frac{n^3 r}{\varepsilon} + n^3 r \log\log\theta)$, which is polynomial to the problem size. □

Theorem 8.5.2. *$TAP-FPTAS$ is a $\{1+\varepsilon, 1+2\varepsilon\}$ bi-criteria FPTAS.*

Proof. The theorem directly follows from Lemmas 8.5.6 and 8.5.7. □

8.6 Results

Experiment Setup

We derive the thermal model with thermal capacitances and thermal resistances for the die and package of an embedded processor from HotSpot [93]. We set the maximum temperature constraint as $100°C$ corresponding to a typical thermal constraint on modern processors. The ambient temperature is set as $45°C$. The initial die temperature and the initial package temperature are set differently as $65°C$ and $55°C$. Since $0.1°C$ rise/fall may take 10^5 cycles with a 3GHz processor [93], the granularity of the time in the experiments is set as milliseconds. We obtained the power consumption model from [79] which is based on the data of an embedded CMOS processor from [41]. We choose 6 voltage levels ranging from 0.6V to 1.1V (0.1V per step). The associated frequencies were between 0.78GHz and 3.8GHz. We coded the proposed optimization techniques in C++ and the experimentations were performed on a core i5/ 2.4GHz/ 8GB Windows 7 PC.

We experimented with realistic benchmarks by combining two kinds of applications from Mediabench [59] and SPEC CPU benchmarks [2] to obtain a task set with 8 jobs. The Mediabench benchmarks include decryption (pegwit), speech compression (rawcaudio, rawdcaudio) and image compression (cjpeg). The SPEC benchmarks include sha, gcc, epic and compress95. We obtained the workload (worst case cycle numbers) of each job from SimpleScalar [89]. The workload of these jobs were in the range of $10^6 - 10^8$ cycles. We evaluated our techniques by experimenting with large synthetic task sets with up to 50 nodes. The number of jobs in each set was varied from 5 to 50 in steps of 5 or 10. At each task set number, we generated 10 sets of tasks. The workload of each job was uniform randomly generated, and varied in the range of $10^6 - 10^8$ cycles. Then, we calculated the execution time and the energy consumption of each job at each active state by the processor model from [79].

Comparisons with the thermal-aware OPT based on a thermal model only considering steady state package temperature

We implemented schedules with both our thermal-aware optimal algorithm for TA_{min} and thermal-aware optimal technique from [112] for the realistic applications with the same amount of iterations. The thermal aware problem addressed in [112] is similar to TA_{min} except that the thermal model in [112] considers only a steady state package temperature. The thermal-aware optimal technique in [112] (denoted as $TAF - OPT$) performs a dynamic programming technique for the optimal latency of periodic task sequence operating at discrete v/f levels under a peak temperature constraint. The technique in [112]

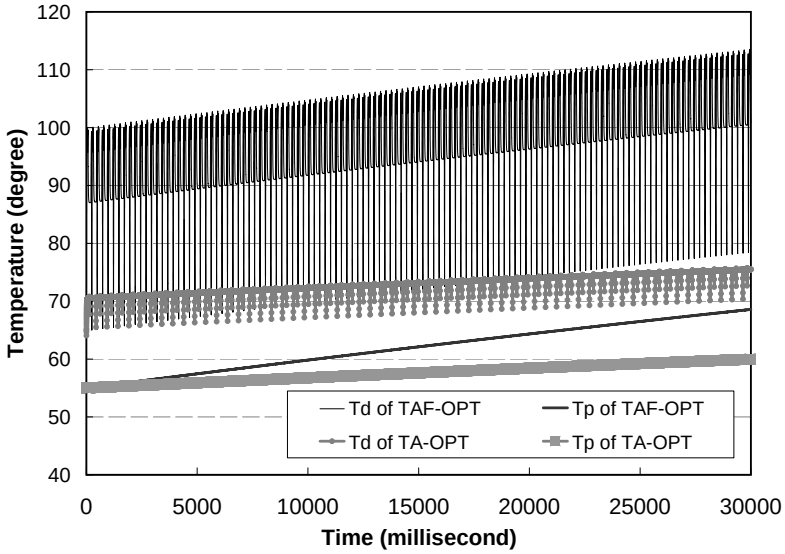


Figure 8.7: Temperature profile of optimal solutions generated by $TA-OPT$ and $TAP-OPT$ with multiple iterations

assumes the package temperature remains steady as the initial package temperature, which does not consider the impact of the package temperature change to the die temperature when the package temperature is not in steady state. We compared the $TA-OPT$ and $TAF-OPT$ schedules for 3000 iterations on a processor with an initial die temperature $65°C$ and an initial package temperature $55°C$. The two schedules execute identical job sequence.

We depict the thermal curves with both optimal schedules in Figure 8.7. Both the package temperatures of the two schedules are rising when the schedules are executed in multiple iterations. As we can see, the package temperature by $TAF-OPT$ schedule is rising faster than the one by the $TA-OPT$ schedule. The rising package temperatures cause die temperatures rising by the two schedules. The $TAF-OPT$ schedule generates thermal constraint violations (up to $180°C$). The $TA-OPT$ schedule keeps the die temperature under thermal constraint all the time. Figure 8.6 depicts the thermal profiles of both schedules when the package temperature reaches a steady state. The steady state package temperature of $TAF-OPT$ schedule is at $132.2°C$ (in practice such a high package temperature can even cause the temperature runaway), while the one of $TA-OPT$ schedule is at $83.6°C$. Compared to the $TAF-OPT$ schedule, the $TA-OPT$ schedule consumes less average power, which lowers the package temperature in steady state. These observations demonstrate that the thermal aware OPT technique based on a thermal model without considering the impact of package temperature is unable to satisfy the thermal constraints, which justifies the need for addressing the problem based on a sophisticated thermal

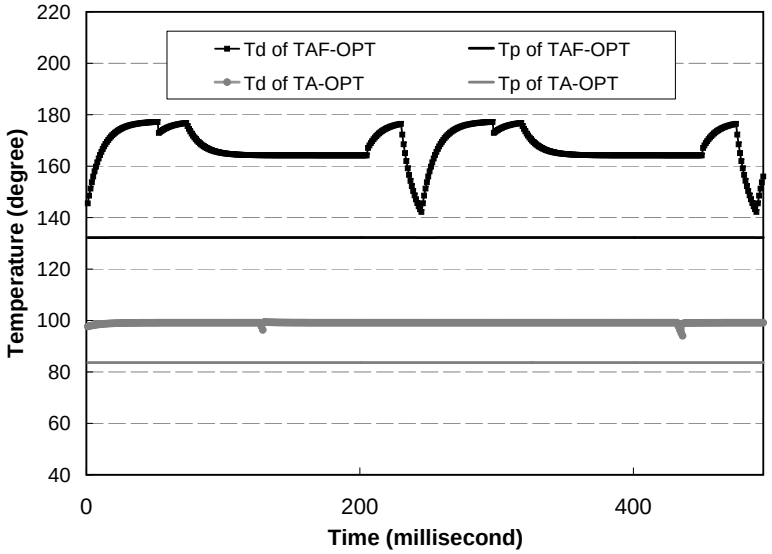


Figure 8.8: Temperature profile of optimal solutions in steady state

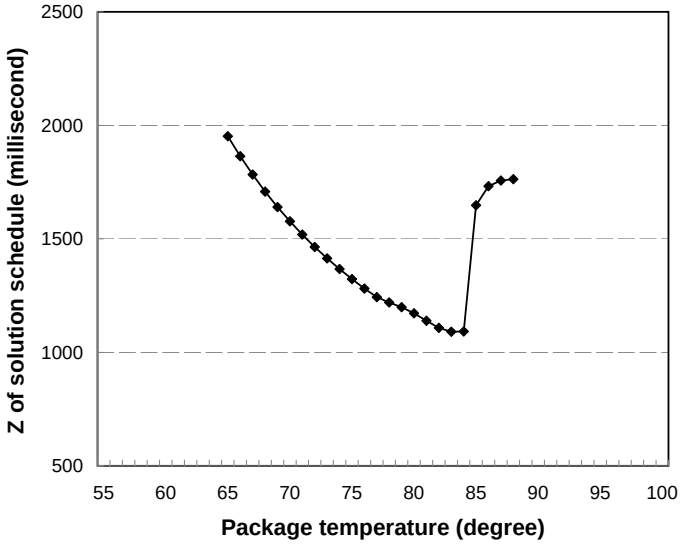


Figure 8.9: Subproblem optimal solution vs. steady state package temperature

model with considering the impact of variable package temperature to the die temperature.

Effect of steady state package temperature settings for TAP_{min}

As we describe in previous section, the TAP_{min} problem is a subproblem of TA_{min}, which considers the task sequence with power budget constraint when the package temperature stays in a particular steady state temperature. In our technique $TA-OPT$ for TA_{min}, we explore the solutions to the TAP_{min} with

various steady state package temperature settings in the range of $[T_{amb}, T_{max}]$. In the experiment, we varied the package temperature settings and generated optimal solutions by our optimal technique $TAP-OPT$ for TAP_{min} with a synthetic application including 30 tasks. We recorded the optimal solution Z value for each TAP_{min} with a package temperature setting and plotted them in the Figure 8.6.

Figure 8.6 demonstrates that solutions to the TAP_{min} problem exist in a certain window of package temperature settings between $65°C$ and $88°$. At the package temperature settings of $T_{sp} > 88°$, there is no feasible solution for the TAP_{min} problem due to too high package temperature setting. At the package temperature settings of $T_{sp} < 65°C$, because of tight power budget constraint, processor has to sleep much long even if the package temperature falls to the ambient temperature. Obviously this kind of schedules are not optimal solutions with minimal latency, we exclude them in the plot. Within the window of package temperature settings with solutions, the optimal solution Z values for the TAP_{min} problems are monotonically decreasing with the increase of T_{sp} in the range of $[65°C, 84°C]$. When T_{sp} is in the range of $[65°C, 84°C]$, the power budget constraint dominates the TAP_{min} problem. Due to the increase of T_{sp}, the associated power budget constraint is relaxing. Thus the solution schedule sleeps less or is able to execute the task at higher power level such that the execution time of solution schedule is reduced. We also found the optimal solution Z values for the TAP_{min} problems are monotonically increasing with the increase of T_{sp} settings in $[84°C, 88°C]$. This is because the peak temperature constraint dominates the TAP_{min} problem when T_{sp} is in $[84°C, 88°C]$. Due to the increase of T_{sp}, the peak temperature constraint is tightened. Thus the solution schedule sleeps more or need to execute at a lower power level to avoid violating the peak temperature constraint, which results in the increase of the latency of solution schedule. $84°C$ is the optimal T_{sp} that both power budget and peak temperature constraints are evenly balanced and thus the optimal schedule solution is the smallest. Since the package temperature settings have such a monotonically decreasing/increasing pattern to the TAP_{min} solutions, we utilize the property in our technique by finding the monotonically increasing/decreasing knee to reduce the search time for the optimal solution to TA_{min}.

Evaluation of the quality of the $TA-FPTAS$ techniques

We evaluated thermal aware schedules by our FPTAS based technique $TA-FPTAS$ with designer specified quality bounds of 5% ($\varepsilon = 0.05$), 10% ($\varepsilon = 0.10$), 15% ($\varepsilon = 0.15$), 25% ($\varepsilon = 0.25$) and 50% ($\varepsilon = 0.50$) with synthetic benchmarks.

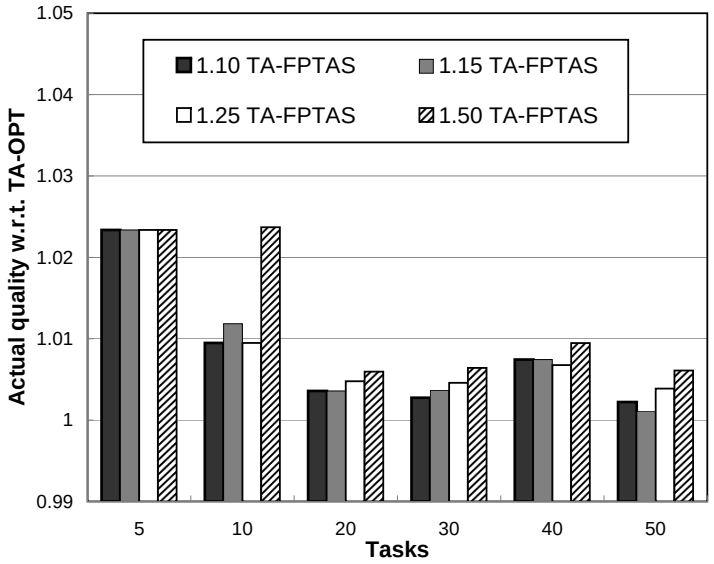


Figure 8.10: Real quality bound w.r.t. the optimal solution

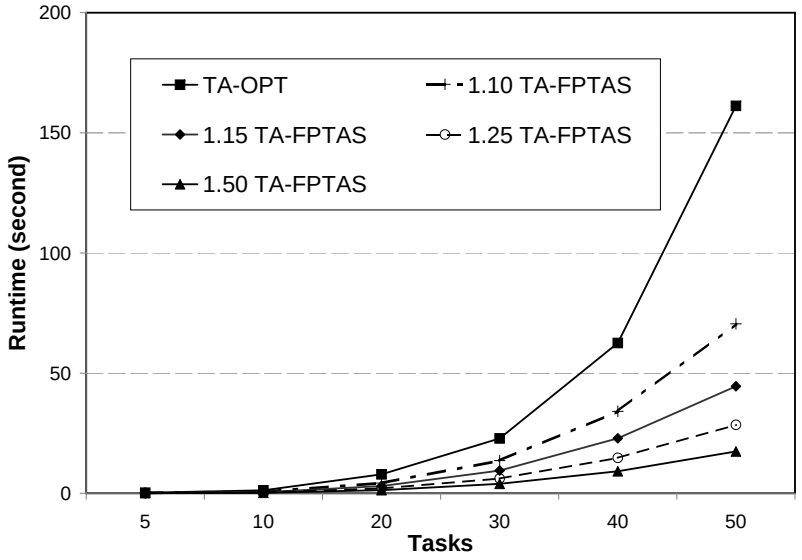


Figure 8.11: Runtime vs. N

Evaluation of the quality bound of $TA-FPTAS$

We evaluated the quality bounds of the $TA-FPTAS$ with the synthetic benchmarks for 5 to 50 tasks. Figure 8.6 illustrates the worst approximation ratio with respect to the solutions to $TA-OPT$ for each task number from 5 to 50. The $TA-FPTAS$ with approximation bound from 10% to 50% matches the $TA-OPT$, since the actual approximation ratios of those are no more than 1.025. Even with the 50% quality bound, the real approximation ratio is no more than 1.024 and the standard deviation of these ratios is no more than 0.01. We also executed the schedules with 1000 iterations and recorded the peak die temperatures during the execution. As expected, none of the schedule exceeds the peak temperature limit $100°C$.

In summary, for the synthetic benchmarks in the experiments, the actual approximation ratios of the schedules generated by the proposed FPTAS are much better than the theoretical bounds.

Evaluation of the run time of $TA-FPTAS$

Figure 8.6 depicts the average running times (in seconds) of the $TA-FPTAS$ with different values of ε for synthetic benchmarks. As expected, the run time of the $TA-OPT$ algorithm is the slowest, while the 1.50 $TA-FPTAS$ is the fastest. The runtime by the $TA-OPT$ algorithm is increasing much faster than the $TA-FPTAS$. As we increases the task number in the experiments which increases the total execution time and the total energy of the task sequence, the figure infers that the runtime by the $TA-OPT$ algorithm is exponential to the increase in execution time and energy consumption, while those of the FPTAS are near-linear to the ones. The run time of the $TA-FPTAS$ algorithm with 50 tasks for a 50% quality bound is under 20 seconds.

In summary, we can obtain a good trade-off between the design quality and solution time by varying ε.

8.7 Conclusions

We introduced a thermal aware performance maximization problem for short task sequence with multiple iterations. Distinct from our previous work in [112], we considered a temperature dependent leakage power model and a sophisticated thermal model derived from HotSpot [93] for a processor with die and package. We justified the problem by demonstrating the inability of the existing thermal aware technique to satisfy the thermal constraint without considering the impact of variable package temperature to the die temperature. We defined the thermal aware scheduling problem TA_{min} and proved that it is NP-hard.

We provided the optimal and FPTAS based techniques for the TA_{min} problem. The techniques are based on the solution techniques to a sub-problem of the TA_{min} problem. The subproblem consider the TA_{min} problem at a steady state package temperature for the short task sequence with power budget constraint. We presented the optimal algorithm and bi-criteria FPTAS algorithm for the subproblem. Experimental results demonstrated that the proposed FPTAS based algorithm for the TA_{min} problem can generate very high quality results even with a designer specified quality bound of 50%. Evaluations of the runtime of the approximation technique showed that our technique is efficient for large task sets with up to 50 nodes.

Chapter 9

Conclusions and future directions

In this chapter, we conclude the research work in the dissertation and present several future potential research directions inspired by this work.

9.1 Conclusions

Th work focuses on the system level power and thermal management for periodic applications on embedded processors with discrete DVFS and DPM capabilities.

System level power management

We address the following work in the context of system level power management.

- We considered the power minimization problem under real time schedules (EDF and RM) on an embedded processor. We formulated the problem as a discrete NP hard problem to minimize the energy consumption of a set of real time applications under utilization bound constraint for EDF/RM schedules. We presented a (1+ε) fully polynomial approximation scheme for the problem that generates a solution within (1+ε) times the optimal. The proposed algorithm offers the lowest computational complexity among existing techniques for the same problem.

- We addressed the energy efficient problem on homogeneous and heterogeneous CMP architectures. we formulated the problem as an integer linear programming problem in order to minimize the latency for a set of applications with an energy budget constraint. We first showed the strongly NP-hardness of the problem and presented 2-approximation algorithms for the problem on both homogeneous and heterogeneous CMP architectures. The proposed algorithms offer the tightest approximation bound among the existing approaches up to the date.

- We considered the battery widely utilized on mobile devices as a limited power source and addressed the battery-aware energy management problem based on a nonlinear battery discharging model. We formulated the problem as a bicriteria problem based on the nonlinear battery discharging model and a deadline constraint. We first showed the NP-hardness of the problem. Then we presented an optimal and a tri-criteria fully polynomial approximation algorithm for the same. The proposed approximation algorithm that generates a solution within (1+ε) times the optimal when the deadline and battery capacity constraints are relaxed to certain bounds. To the best of our

knowledge, this is the first known approximation algorithm the battery-aware energy management problem with nonlinear battery discharging model.

System level thermal management

We address the following work in the context of system level thermal management.

- We addressed a thermal aware scheduling problem that minimizes the latency for a sequence of periodic tasks on an embedded processor under a peak temperature constraint. We showed the problem is NP-hard. Then we proposed the optimal and $(1+\varepsilon)$ fully polynomial approximation scheme as solutions. To the best of our knowledge, this is the first work that propose both optimal and approximation algorithm for the thermal-aware scheduling problem.

- We defined the stochastic version of the thermal aware scheduling problem when the tasks have uncertain execution times. Then we presented an optimal algorithm when the execution cycles of tasks follow discrete distribution. For the tasks whose execution cycles follows normal distribution, we proposed an approximation algorithm as solution. This is also the first work that propose both optimal and approximation algorithm for the stochastic thermal-aware scheduling problem.

- We addressed the task sequencing and scheduling problem on an embedded processor under thermal constraints. The problem seeks to minimize the latency for a periodic application by obtaining an optimized task sequence and DVFS schedules subject to a peak temperature constraint. We first derived an optimal initial temperature that can generate optimum solutions. This is the first work that finds such an optimal initial temperature setting for the addressed problem till to date.We then presented optimal solutions for several sub-problems and a novel algorithm for the general instances of the problem.

- We addressed the thermal aware scheduling problem for periodic applications with many iterations by considering the effect of the package temperature to the die temperature. The problem is to maximize the throughput for a periodic task sequence executing on an embedded processor with multiple iterations. We considered a sophisticated thermal model including die and package and a temperature-dependent leakage power model. We first proved that the problem is NP-hard. We provided a pseudo-polynomial time optimal algorithm and a fully polynomial time approximation scheme (FPTAS) based technique as solutions to the problem. The solution techniques to the thermal aware design problem are constructed on the top of solutions to a subproblem with package temperature and power budget constraints. We showed the NP-hardness of the subproblem. Then

we provided a pseudo-polynomial time optimal algorithm and a bi-criteria FPTAS as solutions for the subproblem. The bi-criteria FPTAS generates solutions within guaranteed quality bound when the power budget constraint is relaxed to a certain amount.

Summary

In the dissertation, we addressed several key system level power and thermal management problems for periodic applications executing on the general embedded processors with discrete DVFS and DPM capabilities. We developed optimal algorithms and/or efficient approximation algorithms for solving the key problems. We also conducted theoretical analysis for certain problems such as deriving the optimal initial temperature setting for the thermal aware task sequencing problem. The proposed efficient algorithms can be utilized in practical embedded applications with fast run time. For all the proposed algorithms, we validated them with extensive experiments for the quality bounds and studied the effects of various parameters to the solutions generated by the proposed techniques.

9.2 Future directions

Thermal aware sequencing for applications with uncertain execution times In the dissertation, we consider the thermal aware scheduling for applications with uncertain execution times. The sequence of the schedule is given. For some applications that are fully pipelined, the task sequence could affect the feasibility of schedules under thermal constraints for tasks with uncertain execution times. One future direction is to study the thermal aware task sequencing for applications with uncertain execution times. The future direction can help the worst case analysis for fully pipelined applications.

Battery-aware power management on CMP We are entering a CMP architecture era. Many embedded systems are designed with CMP architectures in order to increase application throughput etc. The power source of these systems are mainly battery. Therefore, the study on battery-aware power management on CMP architectures is still very important for the design of embedded systems. There exists a considerable amount of work on battery-aware power management on CMP architectures. Chowdhury et al. [21] design static scheduling algorithms for periodic real-time applications on single and multiple processors. Yuan et al. [108] present online battery-aware scheduling algorithms on multiprocessors and extend their work in [15]. Although these techniques are efficient for battery-aware energy management, all of them are heuristic techniques and the quality of the solutions cannot be guaranteed.

In the dissertation, we addressed the battery-aware energy management problem on single processor and power management on CMPs. We provided optimal and fully polynomial approximation

algorithms for both problems as solutions. The provided solutions could potentially enable the study on guaranteed quality techniques for the battery-aware energy management on CMP architectures.

Thermal management for throughput maximization on CMPs The performance of many embedded applications are specified by latency. Thus, the objective to maximize performance becomes to minimize latency of the applications. The latency minimization problem with thermal management on CMPs considers an application is specified by a set of tasks to be executed on the CMP architecture with m identical cores. Latency is defined by the makespan of the set of tasks to be mapped on the CMP. The objective is to minimize makespan of the set of tasks. The outcome involves the mapping from tasks to cores, the execution order of tasks on each core, the v/f assignment for the execution of each task, and the sleep time selection on each core.

In recent past, researchers have begun to address the latency minimization problem with thermal management on CMP architectures [17,26,61,64,101]. [17,26,61,101] consider task specifications, and also consider task allocation as part of the problem. However, all of these techniques assume continuous v/f states for the processors. To the best of our knowledge, there is no existing solutions with quality bounds for latency minimization problem with thermal management on CMP architectures with discrete v/f states. Our work on thermal management for periodic applications for single embedded processors could potentially enable the work on the approximation algorithms for the thermal management problem on CMP architectures.

REFERENCES

[1] Lp/ilp solver. *http://lpsolve.sourceforge.net/5.5/*.

[2] Spec cpu95 and cpu2000 benchmarks. *http://www.spec.org/*.

[3] ACPI. Advanced configuration power interface specification 3.0b. *http://www.acpi.info/*, 2006.

[4] R. Ahuja, T. Magnanti, and J. Orlin. Network flows: Theory, algorithms and applications. *Prentice Hall*, 1993.

[5] A. Andrei, P. Eles, and Z. Peng. Energy optimization of multiprocessor systems on chip by voltage selection. *IEEE Transaction on VLSI Systems*, 15(3):262–275, 2007.

[6] H. Aydin, R. Melhem, D. Mosse, and P. M. Alvarez. Dynamic and aggressive scheduling techniques for power aware real-time systems. In *Proceedings of IEEE Real-Time Systems Symposium*, 2001.

[7] N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 520–529, 2004.

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

[9] N. Bansal and K. Pruhs. Speed scaling to manage temperature. *Proceedings of Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 460–471, 2005.

[10] M. Bao, A. Andrei, P. Eles, and Z. Peng. On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration. *Proceedings of Design Automation Conference (DAC)*, 2009.

[11] L. Benini and G. D. Micheli. A survey of design techniques for sytem-level dynamic power management. *Transactions on VLSI Systems*, 2000.

[12] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. *Proceedings of High Performance Computer Architecture (HPCA)*, pages 171–182, 2001.

[13] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of International Society for Computers and Their Applications (ISCA)*, 2000.

[14] D. Bunde. Power-aware scheduling for makespan and flow. In *Proceedings of ACM symposium on parallelism in algorithms and architectures*, 2006.

[15] C. Yuan and S.M. Reddy and I. Pomeranz and B.M. Al-Hashimi. Workload-ahead-driven online energy minimization techniques for battery-powered embedded systems with time-constraints. *Transaction on Design Automation of Electronic Systems*, 12(1), 2007.

[16] A. K. Chandra, D. S. Hirschberg, and C. K. Wong. Approximation algorithms for some generalized knapsack problems. *Theoretical Computer Science*, (3):293–304, 1976.

[17] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. *Proceedings of Design, Automation and Test in Europe (DATE)*, 2008.

[18] T. Chantem, X. S. Hu, and R. P. Dick. Online work maximization under a peak temperature constraint. *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*, 2009.

[19] J. Chen, C. Hung, and T. Kuo. On the minimization of the instantaneous temperature for periodic real-time tasks. In *Proceedings of RTAS*, 2007.

[20] J. Chen, T. Kuo, and C. Shih. $(1+\varepsilon)$ approximation clock rate assignment for periodic real-time tasks on a voltag-scaling processor. In *Proceedings of International Conference on Embedded Software (EMSOFT)*, 2005.

[21] P. Chowdhury and C. Chakrabarti. Static task-scheduling algorithms for battery-powered dvs systems. *Transactions on VLSI systems*, 13(2):226–237, February 2005.

[22] A. Cohen, L. Finkelstein, A. Mendelson, R. Ronen, and D. Rudoy. On estimating optimal performance of cpu dynamic thermal management. *IEEE Computer Architecture Letters*, 2(1), January 2003.

[23] A. K. Coskun, T. S. Rosing, K. Whisnant, and K. Gross. Static and dynamic temperature-aware scheduling for multiprocessor socs. *IEEE Transactions on VLSI*, 16(9):1127–1140, September 2008.

[24] B. Dean, M. Goemans, and J. Vondrak. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Proceedings of IEEE Symposium on Foundations of Computer Science*, 2004.

[25] S. Gochman, A. Mendelson, A. Naveh, and E. Rotem. Introduction to intel core duo processor architecture. *Intel Technology Journal*, 10(2):89–97, 2006.

[26] M. Gomaa, M. D. Powell, and T. N. Vijaykumar. Heat-and-run: leveraging smt and cmp to manage power density through the operating system. *ACM SIGOPS operating systems review*, 38(5):260–270, December 2004.

[27] D. Hochbaum. Approximation algorithms for np-hard problems. *PWS Publishing Company*, 1997.

[28] H. Hsu, J. Chen, and T. Kuo. Multiprocessor synthesis for periodic hard real-time tasks under a given energy constraint. In *Proceedings of Design, Automation and Test in Europe (DATE)*, 2006.

[29] S. Hua, G. Qu, and S. Bhattacharyya. Energy reduction techniques for multimedia applications with tolerance to deadline misses. *Proceedings of Design Automation Conference (DAC)*, 2003.

[30] H. Huang, G. Q. J. Fan, and M. Qiu. Throughput maximization for periodic real-time systems under the maximal temperature constraint. *Proceedings of Design Automation Conference (DAC)*, 2011.

[31] M. Huang, J. Renau, S. Yoo, and J. Torrellas. A framework for dynamic energy efficiency and temperature management. *Proceedings of International Symposium on Micro-architecture*, pages 202–213, 2000.

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

[33] IBM. Ibm powerpc 970fx risc microprocessor data sheet. 2007.

[34] Intel. Intel pentium 4 processor in the 478-pin package thermal design guidelines. *ftp://download.intel.com/design/pentium4/guides/24988903.pdf*, 2002.

[35] Intel. Intel pxa270 processor: electrical,mechanical, and thermal specification. 2005.

[36] Intel. Intel pentium d processor, intel pentium processor extreme edition, intel pentium 4 processor and intel core2^{TM} duo extreme processor x6800 - thermal and mechanical design guidelines. 2007.

[37] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the 14th Symposium on Discrete Algorithms*, 2003.

[38] C. Isci and A. B. et al. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. *IEEE/ACM International Symposium on Micro-architecture (MICRO)*, 2006.

[39] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamic variable voltage processors. In *Proceedings of International Symposium on Low Power Electronics (ISLPED)*, 1998.

[40] R. Jayaseelan and T. Mitra. Temperature aware task sequencing and voltage scaling. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2008.

[41] R. Jejurikar, C. Pereira, and R. Guptar. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of Design Automation Conference (DAC)*, 2004.

[42] N. Jha. Low power system scheduling and synthesis. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2001.

[43] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer-Verlag, 2004.

[44] W. Kim, J. Kim, and S. L. Min. A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis. In *Proceedings of Design, Automation and Test in Europe (DATE)*, 2002.

[45] J. Kleinberg, Y. Rabani, and E. Tardos. Allocating bandwidth for bursty connections. *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 664–673, 1997.

[46] C. M. Krishna and Y. H. Lee. Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems. *IEEE Transactions on Computers*, 52(12), December 2003.

[47] S. Krumke, M. Marathe, H. Noltemeier, R. Ravi, and S. Ravi. Approximation algorithms for certain network improvement problems. *Journal of Combinatorial Optimization*, 2(3):257–288, September 1998.

[48] C. Lasance. Recent progress in compact thermal models. *IEEE Semiconductor Thermal Measurement and Management Symposium*, pages 290–299, March 2003.

[49] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, (4):339–356, 1979.

[50] W. Lee, K. Patel, and M. Pedram. Dynamic thermal management for mpeg-2 decoding. *Proceedings of International Symposium on Low Power Electronics (ISLPED)*, pages 316–321, 2006.

[51] J. Li and J. Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *Proceedings of High-Performance Computer Architecture (HPCA)*, 2006.

[52] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in hard-real-time environment. *Journal of ACM*, 20(1), January 1973.

[53] J. Liu. Real-time systems. *Prentice-Hall*, 2000.

[54] Y. Liu, R. Dick, L. Shang, and H. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *Proceedings of Design, Automation and Test in Europe (DATE)*, 2007.

[55] Y. Liu, H. Yang, R. Dick, H. Wang, and L. Shang. Thermal vs energy optimization for dvfs-enabled processors in embedded systems. *Proceedings of International Symposium on Quality Electronic Design (ISQED)*, 2007.

[56] J. Lorch and A. J. Smith. Improving dynamic voltage scaling algorithms with pace. *Proceedings of ACM SIGMETRICS*, 2001.

[57] D. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28:213–219, 2001.

[58] R. Mcgowen. Adaptive designs for power and thermal optimization. *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, pages 118–121, 2005.

[59] MediaBench. Mediabench ii benchmark. *http://euler.slu.edu/ fritts/mediabench/*.

[60] P. Mejia-Alvarez, E. Levner, and D. Mosse. Adaptive scheduling server for power aware real-time tasks. *ACM Transactions in Embedded Computing Systems (TECS)*, 3(2):284–306, May 2004.

[61] A. Merkel, A. Weissel, and F. Bellosa. Event-driven thermal management in smp systems. *Proceedings of the Second Workshop on Temperature-aware Computer Systems*, June 2005.

[62] B. Mochocki, X. S. Hu, and G. Quan. A unified approach to variable scheduling for nonideal dvs processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(9), September 2004.

[63] B. Mochocki, X. S. Hu, and G. Quan. Practical on-line dvs scheduling for fixed-priority real-time systems. In *Proceedings of Real-time and Embedded Technology and Applications Symposium (RTAS)*, 2005.

[64] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. P. Boyd, and G. D. Micheli. Temperature-aware processor frequency assignment for mpsocs using convex optimization. *Proceedings of International Conference on Hardware/Software Co design and System Synthesis*, pages 111–116, 2007.

[65] N. Bansal and K. Pruhs. Speed scaling to manage temperature. *Proceedings of Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 460–471, 2005.

[66] D. S. Naidu. Optimal control systems. *CRC press*, 2003.

[67] K. Niyogi and D. Marculescu. Speed and voltage selection for gals systems based on voltage/frequency islands. In *Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC)*, 2005.

[68] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. *ACM Symposium on Operating Systems Principles*, 2001.

[69] C. Papadimitriou and K. Steiglitz. Combinatorial optimization: algorithms and complexity. *Dover Publications*, 1998.

[70] H. Pape and G. Noebauer. Generation and verification of boundary independent compact thermal models for active components according to the delphi/seed methods. *IEEE Semiconductor Thermal Measurement and Management Symposium*, pages 201–211, March 1999.

[71] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of ACM Symposium on Operating Systems Principles*, 2001.

[72] K. Pruhs, R. van Stee, and P. Uthaisombut. Speed scaling of tasks with precedence constraints. In *Proceedings of the 3rd workshop on approximation and on-line algorithms*, volume 3879 of LNCS, 2005.

[73] M. Qiu, C. Xue, and H.-M. Sha. Voltage assignment with guaranteed probability satisfying timing constraint for real-time multiprocessor dsp. *Journal of VLSI Signal Processing*, 46:55–73, 2007.

[74] G. Quan, Y. Zhang, W. Wiles, and P. Pei. Guaranteed scheduling for repetitive hard real-time tasks under the maximal temperature constraint. In *Proceedings of International Conference on Hardware/Software Co-design and System Synthesis (CODES+ISSS)*, 2008.

[75] R. Viswanath et al. Thermal performance challenges from silicon to systems. *Intel Corporation, Technical Report*, 2000.

[76] D. Rai, H. Yang, I. Bacivarov, J. Chen, and L. Thiele. Worst-case temperature analysis for real-time systems. *Proceedings of Design, Automation, and Test in Europe Conference (DATE)*, 2011.

[77] D. Rajan and P. S. Yu. Temperature-aware scheduling: when is system throttling good enough? *IBM Research Report*, 2007.

[78] D. Rakhmatov and S. Vrudhula. Energy management for battery-powered embedded systems. *Transactions on Embedded Computing System*, 2(3):277–324, 2003.

[79] R. Rao, S. Vrudhula, C. Chakrabarti, and N. Chang. An optimal analytical for processor speed control with thermal constraints. *Proceedings of International Symposium on Low Power Electronics (ISLPED)*, pages 292–297, 2006.

[80] R. Rao, S. Vrudhula, and N. Chang. Battery optimization vs energy optimization: which to choose and when? *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2005.

[81] S. Martin and K. Flautner and T. Mudge and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2002.

[82] M. Sabry. Compact thermal models for electronic systems. *IEEE Transactions on Components and Packaging Technologies*, 26(1):179–185, March 2003.

[83] M. Sabry and S. Hossam. Compact thermal models: a global approach. *Proceedings of Thermal Issues in Emerging Technologies: Theory and Application*, 26(1):33–39, January 2007.

[84] M. Sabry, M. Tawfik, H. Elahawy, S. Garcia-Sabiro, and J. Besnard. A novel and efficient technique for transient analysis of tightly coupled circuits: The integral equation method (iem). *Proceedings of European Design and Test Conference (EURO-DAC)*, pages 86–89, 1993.

[85] S. Sharifi and T. Rosing. Package-aware scheduling of embedded workloads for temperature and energy management on heterogeneous mpsocs. *Proceedings of International Conference on Computer Design (ICCD*, 2010.

[86] Y. Shin, K. Choi, and T. Sakura. Power optimization of real-time embedded systems on variable speed processors. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2000.

[87] Y. Shin, K. Choi, and T. Sakura. Dynamic voltage and frequency scaling under a precise energy model considering variable and fixed components of the system power dissipation. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2004.

[88] D. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–471, 1993.

[89] SimpleScalar. http://www.simplescalar.com/.

[90] A. Sinha and A. P. Chandrakasan. Jouletrack-a web based tool for software energy profiling. In *Proceedings of Design Automation Conference (DAC)*, 2001.

[91] K. Skadron. Hybrid architectural dynamic thermal management. *Proceedings of Design, Automation and Test in Europe (DATE)*, 1:10–15, 2004.

[92] K. Skadron and M. S. et al. Temperature-aware microarchitecture. *Proceedings of International Symposium on Computer Architecture*, 2003.

[93] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware micro-architecture. *Proceedings of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2003.

[94] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. *Proceedings of International Conference on Supercomputing (ICS)*, 2003.

[95] G. Varatkar and R. Marculescu. Communication-aware task scheduling and voltage selection for total systems energy minimization. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2003.

[96] V. Vazirani. Approximation algorithms. *Springer-Verlag*, 2001.

[97] V.Swaminathan and K.Chakrabarty. Network flow techniques for dynamic voltage scaling in hard real-time systems. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 2004.

[98] S. Wang and R. Bettati. Delay analysis in temperature-constrained hard real-time systems with general task arrivals. In *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2006.

[99] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. In *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2006.

[100] F. Xie, M. Martonosi, and S. Malik. Bounds on power savings using runtime dynamic voltage scaling: An exact algorithm and a linear-time heuristic approximation. In *Proceedings of International Symposium on Low Power Electronics (ISLPED)*, 2005.

[101] Y. Xie and W.-L. Hung. Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (mpsoc) design. *Journal of VLSI signal processing*, 45(3):177–189, December 2006.

[102] R. Xu, D. Mosse, and R. Melhem. Minimizing expected energy consumption in real-time systems through dynamic voltage scaling. *Proceedings of ACM SIGMETRICS*, 2001.

[103] G. Xue and A. S. et al. Finding a path subject to many additive qos constraints. *IEEE/ACM Transaction on Networking*, 15(1):201–211, 2007.

[104] L. Yan, J. Luo, and N. K. Jha. Combined dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2003.

[105] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin. Dynamic thermal management through task scheduling. In *Proceedings of International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2008.

[106] P. Yang and F. Catthoor. Pareto-optimization-based run-time task scheduling for embedded systems. In *Proceedings of International Conference on Hardware/Software Co-design and System Synthesis (CODES+ISSS)*, 2003.

[107] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *IEEE Annual Foundations of Computer Science*, 1995.

[108] C. Yuan, S. Reddy, I. Pomeranz, and B. Al-Hashimi. Battery-aware dynamic voltage scaling in multiprocessor embedded system. *Proceedings of International Symposium on Circuits and Systems (ISCAS)*, 2005.

[109] L. Yuan, S. Leventhal, and G. Qu. Temperature-aware leakage minimization technique for real-time systems. *UMIACS Technical Report, University of Maryland, UMIACS-TR-2006-02*, 2006.

[110] H. Yun and J. Kim. On energy-optimal voltage scheduling for fixed-priority hard real-time systems. In *Transactions on Embedded Computing Systems*, 2003.

[111] S. Zhang and K. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2007.

[112] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature aware scheduling problem. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2007.

[113] X. Zhong and C. Xu. System-wide energy minimization for real-time tasks: Lower bound and approximation. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2006.

[114] Y. Zhu and F. Mueller. Feedback edf scheduling exploiting dynamic voltage scaling. In *Proceedings of Real-time and Embedded Technology and Applications Symposium (RTAS)*, 2004.

[115] J. Zhuo and C. Chakrabarti. System-level energy-efficient dynamic task scheduling. In *Proceedings of Design Automation Conference (DAC)*, 2005.