UNIVERSITY OF CALIFORNIA, SAN DIEGO

Accurate Temperature Sensing and Efficient Dynamic Thermal Management in MPSoCs

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science (Computer Engineering)

by

Shervin Sharifi

Committee in charge:

Professor Tajana Simunic Rosing, Chair
Professor Chung-Kuan Cheng
Professor Tara Javidi
Professor Ryan Kastner
Professor Joseph Pasquale

2011

The dissertation of Shervin Sharifi is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2011

DEDICATION

To those who have dedicated their lives to me, *Mom* and *Dad*.

And to my dearest *Avisha*.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The work presented in this thesis would not have been possible without the help and support of wonderful people that I have had the privilege of interacting with during my years at UC San Diego.

First and foremost, I would like to thank my advisor, Professor Tajana Simunic Rosing for her guidance, support and the many lessons I have learned from her. I especially thank her for her understanding, support and patience during the difficult times I had. Without her help, I would not have been able to overcome many technical and non-technical challenges. In addition, I really appreciate the supportive and friendly environment she has created in our research group.

I really appreciate the effort and the time my thesis committee members, Professor Chung-Kuan Cheng, Professor Tara Javidi, Professor Ryan Kastner and Professor Joseph Pasquale have taken to review my manuscript and conduct my defense.

I have also been very lucky to have wonderful colleagues and friends in our department and in our research group. I wish to thank them all for providing a friendly and joyful environment and for their valuable comments and discussions. Special thanks to Raid Ayoub, Yen-Kuan Wu, Ayse Coskun, Gaurav Dhiman, Edoardo Regini, Priti Aghera, Nima Nikzad, Bryan Kim, Yashar Asgarieh, Richard Strong, Aruna Ravinagarajan, Jamie Bradley Steck and Giacomo Marchetti. I had the privilege to collaborate with Raid Ayoub, Ayse Coskun, Dilip Krishnaswamy and Chun-Chen Liu.

I have been very fortunate to have a wonderful group of friends in San Diego. I sincerely thank them for their friendship, support and being my family when I was far from my own family. I would like to especially thank Kambiz Samadi, Amirali Shayan, Ahsan Samiee, Kiarash Kiantaj, Hamed Movahedpour, Ehsan Ardestanizadeh, Haleh Azartash, Behrokh Farzad and Setareh Setayesh among the others. We have shared many memorable moments which I will never forget.

My utmost and deepest gratitude, affection and love belong to my family, especially my parents for their unconditional and endless love and support. I am

greatly indebted to my older brother, Babak. He is the best brother one could ever ask for. I also truly appreciate the one-of-a-kind kindness and care of my sister, Negar, who has always been there for me. They have contributed in many ways to the person I am today. I would like to thank the other half of my family, especially my parents in law, for their support and giving me the most wonderful gift of my life, Avisha. I'm very lucky to have wonderful sisters and brothers-in-law, Ana, Newsha, Arshia and Mehdi whose sincere supports and prayers have encouraged me in this journey. I feel blessed to have such a family. Most of all, I wish to express my deepest and most sincere love and gratitude to my dearest Avisha, who has made my life sweeter than I could have ever imagined with her love, understanding and patience. As my other half, she has made me complete. This thesis is dedicated to her and my parents.

Chapter 2 in part, is a reprint of the material as it appears in Proceedings of the Great Lakes Symposium on VLSI, 2008. Sharifi, S. and Rosing, T. S. and IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems, 2010. Sharifi, S. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

Chapter 3 in part, is a reprint of the material as it appears in International Symposium on Quality Electronic Design, 2008. Sharifi, S. Liu, C. and Rosing, T. S. and IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems, 2010. Sharifi, S. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

Chapters 4 and 5 in part, are reprints of the material accepted for publication at Design, Automation and Test in Europe (*DATE*) 2012. Sharifi, S. Ayoub, R. and Rosing, T.S. and and the material under submission at IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems. Sharifi, S. Krishnaswamy, D. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

VITA

2000	B.S. in Computer Engineering, Sharif University of Technology, Tehran, Iran
2003	M.S. in Computer Engineering, University of Tehran, Tehran, Iran
2011	Ph.D. in Computer Science (Computer Engineering), University of California, San Diego

PUBLICATIONS

Shervin Sharifi, Raid Ayoub, Tajana Simunic Rosing, "TempoMP: Integrated Prediction and Management of Temperature in Heterogeneous MPSoCs", *To appear in the proceedings of Design, Automation and Test in Europe (DATE)*, 2012.

Yen-Kuan Wu, Shervin Sharifi, Tajana Simunic Rosing, "Distributed Thermal Management for Embedded Heterogeneous MPSoCs with Dedicated Hardware Accelerators", *IEEE International Conference on Computer Design (ICCD)*, 2011.

Shervin Sharifi, Yen-Kuan Wu, Tajana Simunic Rosing, "Temperature-aware Scheduling for Embedded Heterogeneous MPSoCs with Special Purpose IP Cores", *IEEE International Workshop on Energy and Thermal Management of Embedded Computing (ETMEC)*, 2011.

Shervin Sharifi, Tajana Simunic Rosing, "Accurate Direct and Indirect On-Chip Temperature Sensing for Efficient Dynamic Thermal Management", *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems.* Vol. 29, No. 10, October 2010.

Shervin Sharifi, Tajana Simunic Rosing, "Package-Aware Scheduling of Embedded Workloads for Temperature and Energy Management on Heterogeneous MPSoCs", *IEEE International Conference on Computer Design (ICCD)*, 2010.

Raid Ayoub, Shervin Sharifi, Tajana Simunic Rosing, "Cooling Aware Proactive Workload Scheduling in Multi-Machine Systems", *Design, Automation and Test in Europe (DATE)*, 2010.

Shervin Sharifi, Ayse Kivilcim Coskun, Tajana Simunic Rosing, "Dynamic Energy and Thermal Management in Heterogeneous Embedded Multiprocessor SoCs", *Asia and South Pacific Design Automation Conference*, 2010.

Mohammad Hosseinabady, Shervin Sharifi, Fabrizio Lombardi, Zainalabedin Navabi, "A Selective Trigger Scan Architecture for VLSI Testing", *IEEE Transactions on Computers*, Vol. 57, No. 3, March 2008.

Shervin Sharifi, Tajana Simunic Rosing, "An Analytical Model for the Upper Bound on Temperature Differences on a Chip", *Great Lakes Symposium on VLSI*, 2008.

Shervin Sharifi, ChunChen Liu, Tajana Simunic Rosing, "Accurate Temperature Estimation for Efficient Thermal Management", *International Symposium on Quality Electronic Design*, 2008.

Reza Barzin, Satoru Fukushima, William Howden, Shervin Sharifi, "Superfit Combinational Elusive Bug Detection", *IEEE International Computer Software and Applications Conference*, 2008.

Shervin Sharifi, Javid Jaffari, Mohammad Hosseinabady, Ali Afzali-Kusha, Zainalabedin Navabi, "Scan-Based Structure with Reduced Static and Dynamic Power Consumption", *Journal of Low Power Electronics*, Vol. 2, No. 3, December 2006.

Shervin Sharifi, Javid Jaffari, Mohammad Hosseinabady, Ali Afzali-Kusha, Zainalabedin Navabi, "Simultaneous Reduction of Dynamic and Static Power in Scan Structures", *Design, Automation and Test in Europe (DATE)*, 2005.

Safar Hatami, Shervin Sharifi, Hossein Ahmadi, Mahmoud Kamarei, "Real-Time Image Compression based on Wavelet Vector Quantization, Algorithm and VLSI Architecture", *IEEE International Sympium on Circuits and Systems (ISCAS)*, 2005.

Safar Hatami, Shervin Sharifi, Mahmoud Kamarei, Hossein Ahmadi, "Hardware implementation of 2D Discrete Wavelet Transform by using Non-separable Lifting Scheme", *International Workshop on Systems, Signals and Image Processing*, 2005.

Safar Hatami, Shervin Sharifi, Mahmoud Kamarei, Hossein Ahmadi, Ahdiyeh Delfan Abazari, "Non-Separable 2-D and 3-D Discrete Wavelet Transform for Image and Video Processing Using Lifting Scheme", *International Workshop on Systems, Signals and Image Processing*, 2005.

Shervin Sharifi, Mohammad Hosseinabady, Pedram Riahi, Zainalabedin Navabi, "Reducing Test Power, Time and Data Volume in SoC Testing Using Selective Trigger Scan Architecture", *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, 2003.

Mohammad Hosseinabady, Shervin Sharifi, Zainalabedin Navabi, "A Novel Partition-based technique for Reducing Power, Time and Data Volume in SoC Testing", *IEEE Workshop on RTL and High Level Testing*, 2003.

Shervin Sharifi, Mohammad Hosseinabady, Zainalabedin Navabi, "Selective Trigger Scan Architecture for Reducing Power, Time and Data Volume in SoC Testing", *International Conference on Very Large Scale Integration*, 2003.

Mohammad-Reza Kakoee, Shervin Sharifi, Zainalabedin Navabi, "Generic Synthesis of Digital Designs", *International Symposium on Telecommunications*, 2003.

Masoud Hashempour, Shervin Sharifi, Maziar Gudarzi, Shaahin Hessabi, "Rapid Design Space Exploration of DSP Applications Using Programmable SoC Devices: A Case Study", *IEEE International ASIC/SoC Conference*, 2002.

ABSTRACT OF THE DISSERTATION

Accurate Temperature Sensing and Efficient Dynamic Thermal Management in MPSoCs

by

Shervin Sharifi

Doctor of Philosophy in Computer Science (Computer Engineering)

University of California, San Diego, 2011

Professor Tajana Simunic Rosing, Chair

Constant increase in performance demands, more aggressive technology scaling and higher transistor integration capacity result in continuously increasing power density and temperature in multi-processor System-on-Chip (SoC) devices. Dynamic thermal management (DTM) techniques try to avoid thermal violations by enabling the chip to control its temperature at runtime. To do this, accurate runtime temperature information is necessary, which is typically obtained from on-die thermal sensors. Sensor accuracy can be significantly affected by factors such as sensor degradation and failure, limitations on the number of sensors and their placement, dynamic change of hotspot locations, etc. To improve the accuracy of temperature sensing, which directly affects the efficiency of DTM, two techniques

are proposed. *Accurate direct temperature sensing* is a design time technique for optimum allocation and placement of on-chip thermal sensors. It targets the inaccuracies due to sensor placement and can reduce the number of thermal sensors by 16% on average. *Accurate indirect temperature sensing* is a runtime technique which targets the sources of inaccuracy which cannot be addressed at design time. Based on inaccurate readings from a few noisy sensors, this method accurately estimates the temperature at any location on the die. It also reduces mean absolute error and standard deviation of the errors by up to an order of magnitude.

DTM efficiency can be improved by predicting changes in temperature and proactively controlling them, which reduces DTM's response time and performance overhead. We propose a temperature prediction technique called *Tempo* to accurately evaluate the thermal impact of DTM actions. Compared to previous temperature prediction techniques, *Tempo* can reduce the maximum prediction error by up to an order of magnitude.

Heterogeneous MPSoCs which integrate various types of cores are particularly at a disadvantage from a thermal perspective, due to the inherent imbalance in power density distribution. We present *PROMETHEUS*, a thermal management framework which systematically performs proactive temperature-aware scheduling for heterogeneous (and homogeneous) MPSoCs. *PROMETHEUS* framework provides two alternative temperature - aware scheduling techniques: *TempoMP* which uses online optimization for optimal power state assignment to the cores, and a more scalable technique *TemPrompt*, which is based on a heuristics and has a lower overhead.

Chapter 1

Introduction

MPSoCs are increasingly used to build complex systems for applications such as networking, communications, digital signal processing (DSP) and multimedia. Currently, the majority of modern processors, even in the mobile space, integrate multiple processors on a single chip. Qualcomm's Snapdragon [63] and Texas Instruments' OMAP [78] are examples of embedded MPSoCs. Some MPSoCs, including the above examples, are heterogeneous and integrate cores of various types on the same die.

In modern VLSI systems, technology scaling and transistor integration capacity have constantly increased with each new generation of processors [12]. As shown in Figure 1.1(a), the integration capacity doubles every two years. Frequency and power consumption also increase in an exponential manner as shown in Figure 1.1(b). Due to continuous scaling of technology and increasing integration density, power density significantly increases for each new generation. According to the International Technology Roadmap for Semiconductors (ITRS) estimates, the power density of high performance microprocessors is expected to increase from $50W/cm^2$ at 100nm technology to $100W/cm^2$ at 14nm technology and could be up to $500W/cm^2$ for 3 dimensional (3D) stacked integrated circuits (ICs) [2].

These higher power densities cause higher temperatures and larger temperature variations which in turn lead to degraded reliability and shorter lifetime, increased leakage power, lower performance, timing issues and higher cost. According to [61], more than 50% of all integrated circuit failures are related to thermal

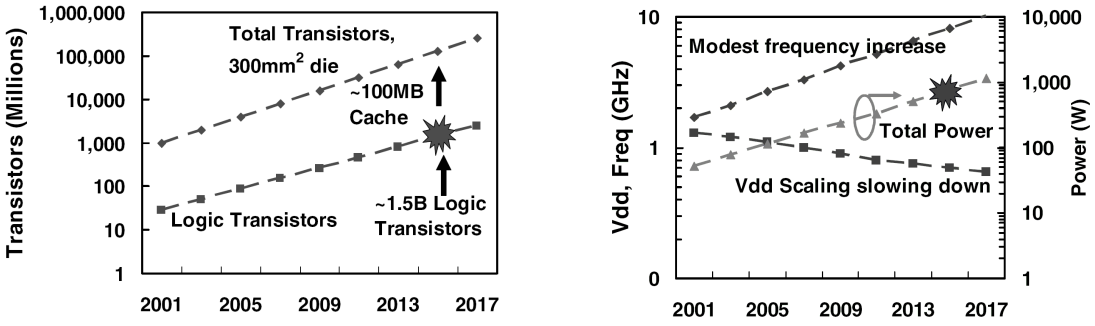


Figure 1.1: Scaling of (a) Transistor integration capacity (2) Frequency, Vdd and power [12]

issues. Gate delay and interconnect resistance increase with temperature, resulting in thermal induced timing and performance issues [68, 9]. Also, leakage power increases almost exponentially with temperature [52]. As a result, controlling the temperature has become one of the main concerns in the electronic system design. In order to efficiently control temperature, thermal considerations need to be taken into account at all stages of design, manufacturing and test of the new generations of VLSI systems [61, 32].

Techniques used to control temperature in the electronic systems are in general called *thermal management techniques*. Temperature and power density can be reduced by reducing the power. However, it should be noted that power-aware design techniques do not necessarily resolve thermal issues since they usually focus on energy efficiency and battery life rather than controlling operating temperature. Figure 1.2 illustrates the fact that while temperature on a chip is correlated with the power density of the heat sources, it also significantly depends on their relative spatial placement. As the figure shows, the location with the highest temperature is not necessarily the one with the highest power density. In other words, controlling the power at one location does not necessarily resolve its thermal issues and power control is not enough to prevent temperature problems.

Table 1.1 provides a broad classification of thermal management techniques

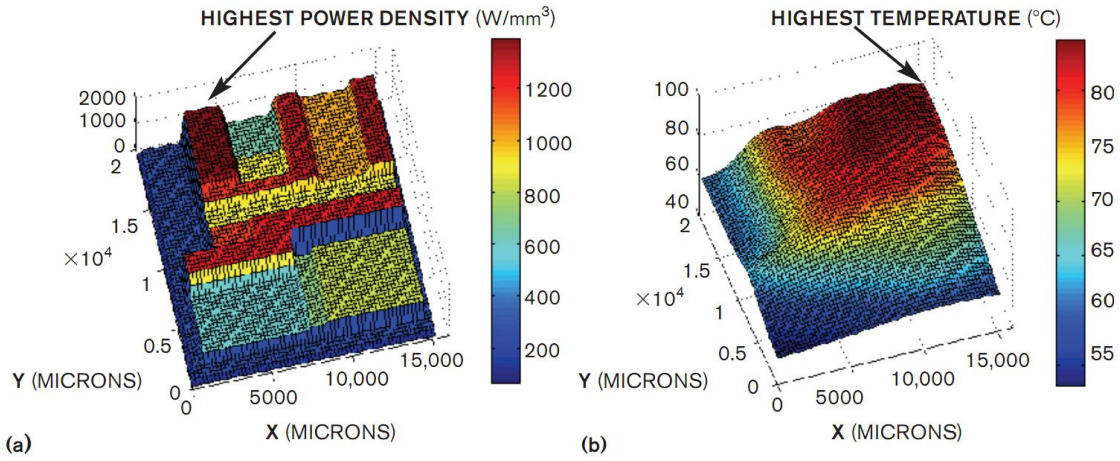


Figure 1.2: Distibution of (a) power density vs. (b) temperature across a chip [67]

Table 1.1: Examples of different classes of thermal management techniques

Classes of Thermal Management Techniques	**Off-chip**	**On-chip**
Design time approaches	Use of better packaging material	Temperature-aware floorplanning
Run time approaches	Dynamic control of fan speed	Temperature-aware scheduling

and one example of each class. As this table shows, thermal management can be applied on or off-chip. Most off-chip techniques work by improving removal of generated heat from the chip, such as by using more complex and advanced cooling techniques. On the other hand, chip-level thermal management techniques usually try to control the temperature by reducing the density of the heat generated by using power control techniques or creating a more balanced distribution of heat generation.

Although off-chip techniques can be very effective, they are usually much more expensive compared to the on chip ones. The ability of the package to remove heat is defined by the junction to ambient thermal resistance (θ_{ja}), expressed as:

$$\theta_{ja} = (T_{chip} - T_{ambient})/P_{chip}$$

where T_{chip} is the on-die junction temperature, $T_{ambient}$ is the ambient (outside package) temperature, and P_{chip} is the maximum power consumption of the chip [76]. Given the maximum power consumption and ambient temperature, the maximum allowable θ_{ja} can be determined by knowing the maximum allowable on-die temperature. Maximum junction temperatures in modern integrated circuits is around 100°C. As P_{chip} rises, θ_{ja} must decrease which means packaging technology must improve to meet the heat dissipation demands. Reduction of thermal junction resistance requires advanced cooling techniques such as larger, more powerful fans, liquid cooling, etc. [76]. In high performance processors, cooling solutions cost $1-3 or more per watt of heat dissipated [74]. Due to the high packaging costs, managing the temperature can allow savings of around hundred dollars for high-end, high power processors [81]. Moreover, in some systems such as mobile embedded systems, using more advanced packaging and cooling is not practical due to cost and space limitations. In such systems, using on-chip thermal management techniques is indispensable. Therefore, on-chip techniques are typically more cost efficient compared to the off-chip ones and might be the only choice for some systems.

Thermal management techniques can be classified into design time or run time techniques. At design time, several thermal-related parameters of the system can be optimized to prevent the potential thermal issues. Different temperature-

aware capabilities are also being added to the EDA (Electronic Design Automation) tools in order to do temperature-related analysis and optimization during the design process. As shown in the table, thermal aware floorplanning is one example of these design time techniques which can reduce processor temperature with minimal performance impact [17, 66].

In contrast, *dynamic thermal management (DTM)* techniques control the temperature of the chip at run time. The goal is to achieve the highest chip performance under a peak temperature limit through adapting the chip's run time behavior when temperatures approach critical levels. Without any run-time thermal management, the packaging and cooling must be designed for worst-case power dissipation which rarely occurs in practice. Designing for worst case is prohibitively expensive. By enabling DTM, packaging and cooling can be designed for power densities exhibited by typical applications, and in the case of thermal emergencies, the chip can dynamically control its temperature. If applications cause the processor to run too hot, the thermal emergency is detected and run time responses are activated to reduce the thermal stress while trying to minimize any associated performance loss. This provides worst-case protection and eliminates the need for expensive packaging and cooling, which makes thermal management essential in modern systems.

To keep the temperature within safe limits, dynamic thermal management techniques generally try to lower or redistribute the heat generation. DTM mechanisms used to do this include migrating activities to other resources on the chip, adjusting the operating parameters of the cores such as voltage and frequency, or stopping the processor's execution completely. All of these actions incur some degrees of performance overhead. Meeting the thermal requirements, while minimizing the performance overhead caused by these mechanisms is the main objectives of DTM techniques.

Effective temperature control needs accurate run time temperature information. The first step in DTM is to capture the variations in the temperature in order to trigger the DTM mechanisms when required. The accuracy of run time temperature information is very important since it directly affects the performance

of CPU and also the effectiveness of DTM in keeping the temperature within limits [74]. Lower or higher temperature estimates cause late or early activation of DTM techniques. Late activation of DTM can result in degraded reliability since the temperature may exceed the designated thresholds. On the other hand, early activation of DTM can have a significant impact on performance [58].

On-chip thermal sensors are the most popular means of obtaining temperature information required for thermal management. However, they can be inaccurate due to a variety of factors including sensor placement, process variation, degradation of sensors, power variations, etc. Often, the sensors cannot be placed exactly at the location they are supposed to monitor. One reason is that hot spot areas on the die are usually also areas where silicon real estate is at premium. Moreover, limitations such as routing may not allow placement of sensors at specific locations. Thus, there can be a significant disparity between sensor readings and the actual temperature at the location of interest [65]. According to [65], there can be temperature differences about 10°C between the sensor and the hotspot. Adding more sensors on the die can partially resolve this problem to some extent, but thermal sensors are costly in terms of hardware, calibration, test, etc.

Design time techniques such as efficient sensor allocation and placement algorithms can increase the accuracy of temperature monitoring while reducing the number and cost of sensors. However, there are other sources of inaccuracy which cannot be addressed at design time. Some of these inaccuracy sources are sensor degradation, process variation, power variations, etc. Other than these factors, the location of hotspots can dynamically change under different workloads. These problems pose several challenges to efficient dynamic management of temperature.

DTM mechanisms can be used reactively or proactively. When used in a reactive manner, the mechanism is engaged after a thermal emergency happens. In contrast, proactive techniques are designed to avoid the occurrence of thermal emergencies. By proactively avoiding thermal emergencies, these techniques may completely prevent thermal violations. Even if a thermal emergency happens, it will be less severe and may be resolved by less aggressive techniques with lower

performance overhead. On the other hand, due to the nature of reactive techniques, they have limited time to respond since they are activated when the emergency has already happened. As a result, to resolve the issues as soon as possible, they need to take more aggressive measures which typically result in higher performance impact.

Proactive thermal management techniques typically rely on temperature predictors in order to estimate the future temperature of the functional units or cores to be able to make more intelligent thermal management decisions. Existing temperature predictors use general signal analysis and prediction techniques and depend only on the temperature history. Therefore, they are not able to accurately evaluate the effect of potential temperature changes which might happen due to future scheduling decisions. Some other predictors need costly run time adaptations. These issues limit their efficiency and effectiveness in predictive decision making for temperature aware scheduling.

Once temperature information can be monitored accurately and future temperature changes can predicted, an effective DTM framework needs intelligent policies to efficiently utilize the available DTM mechanisms. Scheduling is one of the mechanisms which can be used for DTM. The way workload is scheduled on an MPSoC not only impacts the performance, but also has a significant effect on the distribution of power and temperature on the MPSoC. Therefore, it has been extensively used as an effective knob for dynamic thermal management in MPSoCs. In general, task scheduling under thermal constraints is an NP-hard problem [86].

Compared to the single CPU processors, multiple instances of similar processing resources are available on MPSoCs. This creates further opportunities for thermal management by allowing distribution of activities and heat as necessary. Nevertheless, it significantly complicates the thermal management process by increasing the size of the solution space. The complexity of the problem is even worse considering the fact that temperature of any particular point on the die strongly depends on recent temperature and workload history. This complexity is usually overcome by simplifying assumptions and heuristics which compromise the optimality of the solution. The problem is even more complicated for the case

of heterogeneous MPSoCs which integrate more complex cores along with a larger number of simpler cores on the same die in order to provide both single thread performance and parallelism.

In comparison to their homogeneous counterparts, these MPSoCs can achieve significant power and performance advantages [41, 42] if scheduling can take advantage of the heterogeneity by assigning workload to the cores based on their characteristics. Many thermal management techniques have been proposed for homogeneous MPSoCs. However, to the best of our knowledge, heterogeneous MPSoCs have not been studied much from the thermal perspective before.

1.1 Thesis Contributions

The contributions presented in this dissertation can be divided into two main categories. The first category includes techniques for accurate temperature sensing while the techniques in the second category address efficient management of temperature in heterogeneous MPSoCs. The approaches presented here try to apply analytical and formal methods toward thermal management as opposed to the ad-hoc and heuristic methods.

To improve the accuracy of temperature sensing, we propose two techniques. The first one, called *accurate direct temperature sensing* is a design time technique for optimum allocation and placement of on-chip thermal sensors. The other technique, *accurate indirect temperature sensing* is a run time technique which targets the sources of inaccuracy which cannot be addressed at design time. Based on inaccurate readings from a few noisy sensors, it can estimate the temperature at any location on the die .

For efficient dynamic thermal management of heterogeneous MPSoCs, we propose a framework called *PROMETHEUS*. This framework is based on our novel temperature prediction technique called *Tempo*. *PROMETHEUS* provides two proactive temperature-aware scheduling techniques for heterogeneous MPSoCs which use *Tempo* to evaluate the thermal impact of scheduling decisions.

Here we outline the summary of our primary contributions presented in this

dissertation:

Analytical model for upper bound on spatial thermal gradients: We introduce an analytical model for upper bound on temperature difference between any two points on a die. It eliminates the need for simulation overhead to find the maximum temperature differences. It is also able to identify the conditions under which maximum temperature variations might happen, which is useful in generating test data for thermal stress tests and for augmenting benchmarks. As its input it gets the thermal characteristics of the chip and package, and power characteristics of the functional units. Given this information, it finds a close upper bound on temperature difference between any two arbitrary locations on the die. This model is useful in thermal gradient analysis and in applications such as reliability modeling, performance mismatch and clock skew analysis, and thermal sensor placement among other applications. Our experiments show that when using simulations, maximum temperature difference underestimations can be as high as 9°C.

Accurate direct temperature sensing: We propose a design time technique called *accurate direct temperature sensing* which finds the optimum allocation and placement of thermal sensors. It is able to find the minimum number and location of sensors required to monitor a set of locations of interest. To do this, it takes as input the thermal characteristics of the chip and the package, the power characteristics of the functional units, the locations of interest and the desired accuracy for monitoring each location. This technique is based on the analytical model we introduced earlier. Unlike previous sensor allocation and placement techniques, our technique is workload independent and guaranties the accuracy required for monitoring the points of interest. As compared to previously proposed methods, we can reduce the number of sensors needed by 16% on average while guaranteeing the specified sensor accuracy.

Accurate indirect temperature sensing: We propose a run time technique for accurate temperature estimation which we refer to as *accurate indirect temperature sensing* technique. It is able to accurately estimate the temperature at arbitrary locations on the die based on the noisy temperature readings from a

limited number of sensors located further away from the locations of interest. *Accurate indirect temperature sensing* technique requires the thermal characteristics of the die and the package, and power characteristics of the cores in advance. It also needs the locations of available sensors and the locations of interest. At run time, based on the power estimates from the cores and noisy readings from the available thermal sensors, it can accurately estimate the temperature at arbitrary locations of interest on the die where no close by physical sensors are available. A method is also proposed for early detection of new sensor degradations or failures which might happen during the lifetime of the system. This method is used to trigger the calibration of indirect temperature sensing and addressing these new inaccuracies in the technique. Our experimental results show that it is able to reduce the standard deviation and maximum value of temperature estimation errors by an order of magnitude. To the best of our knowledge, *Accurate indirect temperature sensing* is the first technique of this kind.

***Tempo* temperature prediction:** We propose an accurate temperature prediction technique called *Tempo*. Given current temperature and just one sample of the previous temperature of the cores, *Tempo* can accurately predict at runtime what the future temperature of the cores would be for given potential scheduling decisions. It needs the power characteristics of the cores and thermal characteristics of the chip and package in advance, but does not need a training phase or run time adaptation. *Tempo* can be used to evaluate the thermal effects of alternative thermal management decisions to choose the best out of all potential options. Compared to previous state of the art temperature prediction techniques, *Tempo* can reduce the maximum prediction error by up to an order of magnitude.

***PROMETHEUS* framework for proactive thermal management of heterogeneous MPSoCs :** We propose a framework for proactive temperature aware scheduling which is called *PROMETHEUS*. It does not impose any restrictions on size, number and characteristics of the concurrent tasks in the workload which makes it applicable to various classes of workloads. This framework proposes two proactive temperature aware scheduling techniques called *TempoMP* and *TemPrompt*. The ability of these techniques in considering individual perfor-

mance, power and thermal characteristics of the cores in a systematic way makes them applicable to both heterogeneous and homogeneous MPSoCs. Compared to state of the art temperature aware scheduling techniques, our techniques provide better performance, power and energy efficiency while they guarantee meeting a maximum temperature threshold.

Chapter 2 in part, is a reprint of the material as it appears in Proceedings of the Great Lakes Symposium on VLSI, 2008. Sharifi, S. and Rosing, T. S. and IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems, 2010. Sharifi, S. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

Chapter 3 in part, is a reprint of the material as it appears in International Symposium on Quality Electronic Design, 2008. Sharifi, S. Liu, C. and Rosing, T. S. and IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems, 2010. Sharifi, S. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

Chapters 4 and 5 in part, are reprints of the material accepted for publication at Design, Automation and Test in Europe (*DATE*) 2012. Sharifi, S. Ayoub, R. and Rosing, T.S. and and the material under submission at IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems. Sharifi, S. Krishnaswamy, D. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

Chapter 2

Direct temperature sensing

2.1 Introduction

One of the most important aspects of DTM is to capture the run time variations in the temperature caused by the changes in power consumption due to workload changes. This is necessary for accurate and timely response to thermal emergencies. Accuracy of thermal measurements directly affects the efficiency of thermal management as well as the performance of the CPU [74]. Temperature estimates lower (or higher) than the actual temperature may cause late (or early) activation of DTM. Late activation of DTM can result in degraded reliability since the temperature may exceed the designated thresholds. Early activation can have a significant impact on performance, especially in the case of response mechanisms with high invocation time and overhead.

On-chip thermal sensors are the most popular means of obtaining run time temperature information required for dynamic thermal management. Different numbers of sensors are deployed on various modern processors. Cell processor with 9 cores contains 11 thermal sensors [62] and Dunnington Xeon processor with 6 cores contains two thermal sensors per core [43]. An analog on-chip thermal sensor usually consists of a temperature-sensing diode, a calibrated reference current source, and a current comparator. Components other than the thermal diode (e.g. current source) must be placed as far as possible from the hotspot due to their temperature sensitivity. Moreover, the sensed temperature values must be

routed to where they are used. Routing overheads are associated with these can significantly contribute to the overall cost [50].

One of the major problems in direct use of on-chip temperature sensors is the sensor imprecision and noise [58, 65]. There are several factors that cause inaccuracy of temperature measurements. The sensor placement error is one of the most important sources of inaccuracy in values obtained from thermal sensors. Typically sensors can not be placed right at the locations they are supposed to monitor, since hotspots usually happen at high performance and high density areas where silicon is at a premium. This causes a disparity between the actual temperature at the location of interest and the sensor. Increasing the number of sensors can resolve this issue, but the cost of adding a large number of sensors is prohibitive. Moreover, even without considering the cost of sensors, various other limitations such as the need for more channels for routing and I/O may not allow placement of thermal sensors right on the locations of interest.

The technique introduced in this dissertation is a *design time* technique which addresses the problem of efficient placement of on-chip temperature sensors on the die while guaranteeing the desired accuracy at each location of interest. It finds the minimum number of sensors and their locations to cover a number of locations of interest with a maximum acceptable sensor placement error. This technique is based on our analytical model for finding the upper bound of on-chip temperature differences. Our experimental results show that our sensor placement algorithm results in an average of 16% reduction in number of sensors compared to the previous techniques at no cost.

The next section describes the related work. Section 2.3 explains the details of our analytical model for upper bound on temperature difference that is then used in Section 2.4 for the proposed sensor placement method. Experimental results are provided in Section 2.5 and Section 2.6 concludes the chapter.

2.2 Related Work

Different techniques have been proposed for efficient placement of on-chip thermal sensors. These techniques are usually based on identification of the hotspots and placing the minimum number of sensors such that they appropriately cover these hotspots. A sensor placement method is proposed in [46] which is based on the concept of range around a hotspot. This is the maximum distance from the hotspot within which a sensor can be placed while still maintaining the intended accuracy. This technique estimates the maximum temperature difference between a heat source and its surrounding locations based on their distance. The model is based on the assumption that the temperature decays exponentially with this distance from a hotspot. A maximum distance from the hotspot is calculated where temperature difference of all of the points within this distance to the hotspot is less than a maximum acceptable temperature error. Sensors are placed within this distance from the hotspot in order to maintain a desired level of accuracy. Selection of activity factor parameter is not easy and also depends on the application. Therefore the results will not be exact and a pessimistic estimation must be used to guarantee the maximum error. Moreover, when calculating the maximum temperature difference to a hotspot, the result depends only on the distance from the hotspot. In other words, it implies that for all of the points at equal distance from the hotspot, the maximum temperature difference to the hotspot is the same, which is not correct. This can be due to the effect of the location and power consumptions of other power sources on the temperature around a hotspot. Figure 2.1 shows the contour map of maximum temperature difference relative to a point of interest in a multi-processor SoC which is used in our experiments and consists of 6 *XScale®cores* [1]. This figure clearly shows that the maximum temperature differences around the region of interest are not the same for equidistant points from the hotspot.

In [55], the authors introduce a systematic technique for thermal sensor allocation and placement in microprocessors. This technique identifies an optimal physical location for each sensor such that the sensor's attraction towards steep thermal gradient is maximized. However, this approach does not consider the

accuracy of the sensors and does not guarantee the maximum error in the thermal sensor readings.

To the best of our knowledge, no technique has been previously proposed to estimate the temperature difference between various locations on the die. The technique proposed in [46] is a special case of this problem and proposes a model for estimating the temperature at distance d from a heat source. Here, we propose an analytical model for estimating an upper bound for the maximum temperature difference between any two points on the die. Usually in order to estimate the maximum temperature differences and variations, extensive simulations are performed. The upper bound provided by our model is not application dependent and does not involve extensive simulations. Other than sensor placement, the proposed model can be used in analyzing the worst case temperature variations on the chip such as analysis of reliability and performance mismatch. Spatial and temporal temperature variations happen as a result of functional and structural differences and differences between computational activities across the chip and also workload variations during the time. Temperature variations as high as 50°C across the die in a modern microprocessor are reported in [61]. These variations impact reliability and performance of the systems. It is shown in [44] that at moderate temperatures, spatial and temporal temperature gradients determine the device reliability; and to achieve satisfactory reliability, resolving the thermal hotspots alone is not adequate. In [20], a comprehensive framework is proposed for analyzing the effects of temperature and temperature variations on reliability of multi-core systems. Temperature variations may also cause performance mismatch which can lead to performance or functional failures. For example, since wire resistances scale with temperature, difference between temperatures of two regions of the die cause difference between resistances at those two regions which may result in timing issues in interconnects and clock skew problems in clock networks. This makes temperature variations an important factor in clock tree design and optimization [15]. Such issues make analysis of temperature variations and gradients an important issue. Our model which is presented in the subsequent section is a useful tool for analyzing maximum temperature variations across the die.

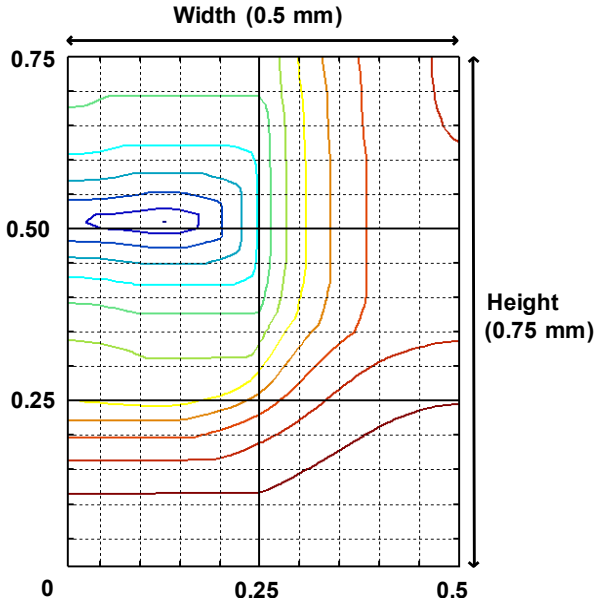


Figure 2.1: Contour map of maximum temperature difference to a point of interest

2.3 Analytical Model for Upper Bound of On-Chip Temperature Differences

Finding the maximum temperature difference between various locations on the die is an important step during the thermal analysis and design as it enables better placement of temperature sensors and helps in evaluation of reliability issues. The maximum temperature difference under potential workloads can be found by extensive simulations, which would incur significant overhead. For systems whose operation depends on interaction with other systems, even the same workload can result in completely different behavior, thus introducing an even higher overhead for temperature estimation. In contrast, our method provides an upper bound with insignificant overhead.

Our techniques here are based on the same thermal model as used in HotSpot. The thermal network generated by the Hotspot model includes thermal resistors and capacitors. Temperature can be modeled at the level of a functional

block, or the die can be divided into regular grid cells (Figure 2.1) to obtain more fine grained estimates. Given the layout and the thermal characteristics of a chip, it is divided into a grid of r rows and c columns as shown in Figure 2.1. The proper size of the grid cells and the number of rows and columns of the grid can be determined by the method proposed in [32]. Our technique works at the granularity of grid cell, therefore when we talk about different locations or points of interest; we actually refer to the corresponding grid cell.

The algorithm starts with the evaluation of the effect each power source has on the temperature differences. This can be done by simulation or with analytical methods. Following this step, the maximum temperature difference between pairs of locations is calculated by exploiting the Linear Time-Invariant (LTI) characteristics of the system. Since the thermal resistors and capacitors are linear components, the thermal network can be considered a linear time-invariant dynamic system. We exploit the LTI characteristics of this system as a basis for calculating an upper bound for the temperature difference. First, we explain it on a simple case of a single input and single output system, and then extend to the thermal networks with multiple inputs and outputs. If we define the power input to the thermal circuit as $p(t)$, the impulse response of the system as $h(t)$, then the temperature output of the system $f(t)$ can be represented as:

$$f(t) = h(t) * p(t) = \int_0^t h(\tau)\, p(t-\tau)\; d\tau \tag{2.1}$$

We assume that minimum and maximum power consumed at each functional unit, p^{Min} and p^{Max}, are known, and that the power consumed at a functional unit is always positive:

$$0 \leq p^{Min} \leq p(t-\tau) \;\; \leq p^{Max} \tag{2.2}$$

$H+$ and $H-$ are the sets of intervals where the impulse response $h(t)$ takes non-negative and negative values respectively. Therefore, the temperature output of the system can be represented as:

$$f(t) = \int_0^t h(\tau)p(t-\tau)\ d\tau = \int\limits_{H+} |h(\tau)|\ p(t-\tau)\, d\tau - \int\limits_{H-} |h(\tau)|\ p(t-\tau)\, d\tau \tag{2.3}$$

Based on equation (2), we know that:

$$p^{Min}\int\limits_{H+} |h(\tau)|\ d\tau\ \leq \int\limits_{H+} |h(\tau)|\ p(t-\tau)\, d\tau \leq\ p^{Max}\int\limits_{H+} |h(\tau)|\ d\tau \tag{2.4}$$

$$p^{Min}\int\limits_{H-} |h(\tau)|\ d\tau\ \leq \int\limits_{H-} |h(\tau)|\ p(t-\tau)\, d\tau \leq\ p^{Max}\int\limits_{H-} |h(\tau)|\ d\tau$$

Equation (4) enables us to then derive the bounds on the value of the output f of a single input single output system based on its impulse response. If we assume the bounds to be f^{Min} and f^{Max} ($f^{Min} \leq\ f(t)\ \leq\ f^{Max}$), they could be calculated as shown below:

$$f^{Min} = p^{Min}\int\limits_{H+} |h(\tau)|\ d\tau\ -\ p^{Max}\int\limits_{H-} |h(\tau)|\ d\tau = p^{Min}\int\limits_{H+} h(\tau)\, d\tau\ +\ p^{Max}\int\limits_{H-} h(\tau)\, d\tau \tag{2.5}$$

$$f^{Max} = p^{Max}\int\limits_{H+} |h(\tau)|\ d\tau\ -\ p^{Min}\int\limits_{H-} |h(\tau)|\ d\tau = p^{Max}\int\limits_{H+} h(\tau)\, d\tau + p^{Min}\int\limits_{H-} h(\tau)\, d\tau$$

We represent a system with m power sources as $p_1(t), \ldots, p_m(t)$, where for each power source maximum and minimum input values are defined as p_i^{Max}, p_i^{Min}.

Assuming $h_i(t)$ is the response of the single output to the impulse on input i, the LTI characteristics of the system imply:

$$f(t) = \sum_{i=1}^{m} f_i(t) = \sum_{i=1}^{m} h_i(t) * p_i(t)$$

Therefore, the minimum and maximum values of the function are:

$$f^{Min} = \sum_{i=1}^{m} f_i^{Min}$$
$$f^{Max} = \sum_{i=1}^{m} f_i^{Max}$$

Equation (7) holds for all outputs of the system. The temperature difference between grid cells a and b is represented by $T_d(a, b)$. Its impulse response to input i, $h_{(a,b),i}(t)$, can be calculated as:

$$h_{(a,b),i}(t) = h_{a,i}(t) - h_{b,i}(t)$$

where $h_{a,i}(t), h_{b,i}(t)$ are impulse responses of temperature at grid cells a and b respectively. Calculation of $T_d(a, b)^{Min}$ and $T_d(a, b)^{Max}$ for each output requires only the knowledge of p_i^{Max}, p_i^{Min} and $h_{(a,b),i}(t)$ as can be seen from equation (5). The impulse response characteristics of the chip, $h_{(a,b),i}(t)$ can be calculated by simulation or by analytical methods.

The process of finding maximum temperature differences is shown in Figure 2.2. Initialization step needs to be done just once, and then its results can be used for the upper bound calculation. To find $h_{j,i}(t)$ during initialization, a step input is applied to power source i while setting all other power sources to 0. Then the step response at grid cell of interest j is used to calculate the impulse response by differentiation. $h_{j,i}(t)$ can also be calculated by analytical methods of linear systems theory.

After initialization, upper bound is calculated for each pair of interest. First the impulse response of temperature difference between the two points caused by power source i is calculated ($h(t) = h_{a,i}(t) - h_{b,i}(t)$). Then based on this impulse response, T_d^{Min} and T_d^{Max} are calculated using equations (5) and (7).

1. Initialization

for all grid cells of interest,

Find $h_{j,i}(t)$ (By simulation-based or analytical methods)

2. Upper bound calculation

for each pair of grid cells of interest (a,b)

Find f^{Min} and f^{Max} using impulse responses initialized in step 1

Figure 2.2: Algorithm for calculating the upper bounds

The calculations can be done only for the pairs of grid cells which are of interest, after completing the simulations for the initialization step (which are done just once). For example, we can use this algorithm to find maximum temperature difference between hotspot a and the set of potential sensor locations around that hotspot $L = \{l_1, l_2, \ldots, l_k\}$. In this case the grid cell pairs of interest are (a, l_1), $\ldots$, (a, l_k). As explained before, many constraints, such as routing, can limit the number of potential sensor locations around a hotspot.

The model can also be used to generate the power trace which results in the maximum temperature difference between any two points. Using equation (5) to maximize the value of $f(t)$ at t_0, we know that $p(t_0 - \tau)$ must take the maximum value at the intervals of τ where$h(\tau) \geq 0$. For example if $h(\tau)$ is non-negative on interval $t_1 < \tau < t_2$, $p(t)$ must take its maximum value on interval $t_0 - t_2 < t < t_0 - t_1$. Similarly it can be shown that for the intervals of τ where $h(\tau) < 0$, $p(t_0 - \tau)$ must take its minimum value. These rules allow us to generate the power trace $p(t)$ which leads to the maximum temperature difference between two arbitrary points. Doing this for all power sources enables us to detect the configurations and scenarios which lead to maximum temperature variations between different points on the die. This information can also be helpful in augmenting the benchmarks

and generating test data for the stress tests.

In the next section, we propose an efficient thermal sensor placement technique which uses our analytical model to minimize the number of thermal sensors while keeping the sensor placement error within acceptable limits.

2.4 Thermal Sensor Placement

The objective of our sensor placement technique is to find the minimum number of sensors and their locations such that the temperature reading errors for each point of interest is within the required accuracy. As shown in Figure 2.1, the chip is divided into a grid. Whenever we specify a point on the die, we are refering to the corresponding grid cell at that location. Let's suppose $Q = \{q_1, q_2, \ldots, q_n\}$ and $E = \{e_1, e_2, \ldots, e_n\}$ are the set of n points of interests and the set of corresponding desired accuracies for these points respectively. We also define a set of potential sensor points $L = \{l_1, l_2, \ldots, l_k\}$ which consists of all of the grid cells around the hotspots where a temperature sensor can be placed. This set is usually determined by other design considerations such as availability of space for locating the sensor, etc. Also, usually sensors could not be placed inside on-chip memory blocks, routing and IO.

The objective is to find the minimum set of sensors (and their locations) $S = \{s_1, s_2, \ldots, s_k\}$ such that for each q_i there exists a s_j for which

$$T(q_i) - T(s_j) < e_i$$

S must be a subset of L.

We introduced the concept of observability area earlier which is defined as the area around a point of interest a within which the maximum temperature difference is always less than the maximum tolerable error. Therefore, if a sensor is placed in the observability area of a point of interest, the sensor placement error would be less than the maximum tolerable error. The observable set of a point of interest a is the set of grid cells in L which completely fall in its observable area. We can define observable set of point of interest q_j as

$$O_j = \left\{ l_i \mid abs(T_d^{\max}(q_j, l_i)) < e_j \ , \ abs(T_d^{\min}(q_j, l_i)) < e_j \right\} \tag{2.6}$$

The inputs to the sensor placement technique are sets Q, E and L mentioned above. In the first step, the observability set of each q_j is calculated using equation (2.6) and our analytical model. Then we find the optimum number of sensors and their locations such that there is at least one sensor in the observable set of each point of interest. This guarantees that the accuracy requirements are satisfied since any sensor placed on a grid cell in the observable set of a point of interest can sense temperature with the required accuracy. This problem is equivalent to the minimum hitting set problem which has been proven to be NP-complete. If $x_a = 1$ when the sensor is placed at grid cell a, and $x_a = 0$ otherwise, then the sensor minimization problem can be formulated as an integer linear program (ILP) as follows:

$$\begin{aligned}
&minimize \quad \sum_{a \in L} x_a \\
&subject\ to \quad \sum_{a \in Oj} x_a \geq 1 \quad i = 1, ..., n \\
&x_a \in \{0, 1\} \quad \forall a \in L \\
&S = \{l_i | x_{l_i} = 1\}
\end{aligned}$$

Minimizing $\sum x_a$ (for $a \in L$) minimizes the total number of sensors while constraint $\sum x_a \geq 1$ (for $a \in O_j$) guaranties that there will be at least one sensor in observable set of hotspot j. For each grid with $x_a = 1$, a sensor is placed in the corresponding grid cell. To solve the ILP problem, we use *lp_solve* [3] which is based on the revised simplex and branch-and-bound methods.

Our proposed sensor placement technique minimizes the cost of sensors while maintaining the desired accuracy. However, our technique may not always be able to place enough sensors in the right locations due to design constraints, and even those sensors that are placed may degrade or fail during the lifetime of the system. These issues call for techniques to compensate for the lack of sensing hardware. Next section presents our accurate indirect temperature sensing technique which is able to estimate the temperature at locations not directly covered by sensors.

2.5 Experimental Results

For our experiments we use two different multi-processor SoCs:

- SoC1 consisting of 6 *XScale*® cores implemented in 90nm process [1] whose layout is shown in Figure 2.1
- SoC2 consisting of two SPARC cores and 4 *XScale*® cores implemented in 65nm process whose layout is shown in 2.3.

A set of programs from the automotive/industrial, network and telecommunications categories of MiBench [28] are selected and run on datasets provided by [24]. Moreover, we have also used a set of programs from Mediabench [45] benchmark suite and run them on SPARC cores.

Pareto distribution is used [73] to introduce idle intervals between the MiBench tasks. A timeout-based dynamic power management policy is applied in order to determine the active and low power states which each core experiences during running these workloads. Power values for *XScale*® are measured on a real system and scaled for the target process. Power values for Alpha cores are calculated using Wattch power estimator [13] integrated with M5 simulator [11]. HotSpot 3.0 [4] grid mode is used for thermal simulations. Each processor is assumed to have its own L2 cache. Parameters used for package are: convection capacitance $140.4 J/K$, convection resistance $0.1 K/W$, spreader thickness 10^{-3}m, and initial temperature of 333°K.

Here we present the experimental results on our analytical model for upper bound on on-chip temperature differences and our sensor placement algorithm.

2.5.1 Maximum Temperature Difference Model

Our proposed model analytically calculates the upper bound on temperature differences on the die eliminating the high overhead of extensive simulations. Very specific combinations of conditions may be required to achieve the maximum temperature difference on the die. Such specific conditions may not be created by the benchmarks, but may happen during the actual operation of the system.

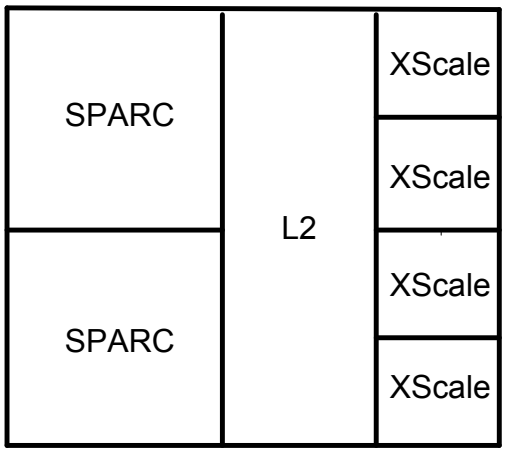


Figure 2.3: Layout of SoC2

Our model can generate a set of input power traces which create such a maximum temperature difference scenario.

Figure 2.4 through Figure 2.6 provide an example of this for SoC1. Figure 2.4 shows a simulation slice in which the temperature difference between points a and b has reached its highest value. The dotted line shows the maximum temperature difference estimated by our model which is clearly higher than the benchmark results. Figure 2.5 shows the situation in which the actual temperature difference found by our method exceeds the maximum temperature difference reported by simulations using standard benchmarks. The first row shows $h_{(a,b),i}(t)$ which is the response of the temperature difference between a and b to the impulse applied at input i.

$h_{(a,b),i}(t)$ is calculated by differentiating the response of the temperature difference to the step input i (since it is in discrete time, this is done by differencing the consecutive samples of step response). To keep the example easy to follow, we considered only three power sources of SoC1 and the rest are turned off. The power traces generating the maximum temperature differences are calculated as explained in section 2.3. Applying each of these traces leads to the corresponding temperature differences shown in the third row. The overall maximum temperature

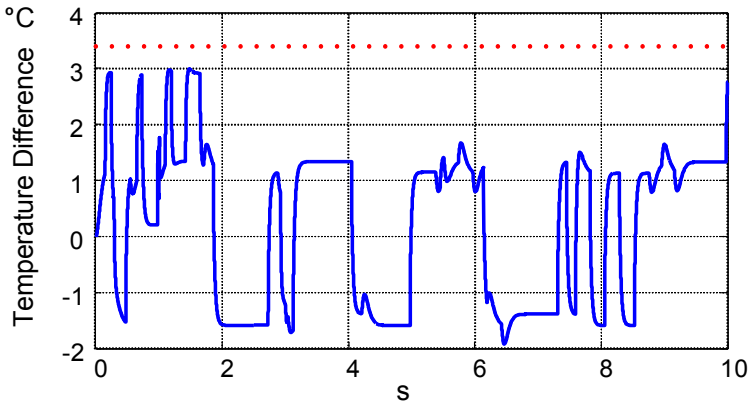


Figure 2.4: Temperature difference between points a and b

difference due to all power sources occurs at time 0.9s (time unit 900) as shown in Figure 2.6.

As shown in this example, the maximum temperature differences can be much larger than what is observed under standard benchmarks. Therefore, when we need to know the maximum temperature differences on the die, running benchmarks may not be enough and models such as ours are needed to get accurate data. The difference between simulations and the model in this particular example is not large because of the low power consumption of *XScale*® cores and the fact that for simplicity we looked at only 3 cores with observation points in proximity of each other.

The error can be much larger in high end processors with higher powered devices. Table 2.1 shows that errors as high as 9°C occur in estimates of temperature differences when relying only on simulations. Even using combinations of different benchmarks does not resolve the problem. Such errors can cause significant functional and reliability issues. For example, if the maximum temperature difference between a sensor and a hotspot is underestimated, it may cause late activation of DTM which can result in serious reliability problems.

The simulation overhead of our method is insignificant. As explained in Section 2.3, this method involves simulating one step input for each power source in the worst case compared to simulating the whole set of benchmarks when standard

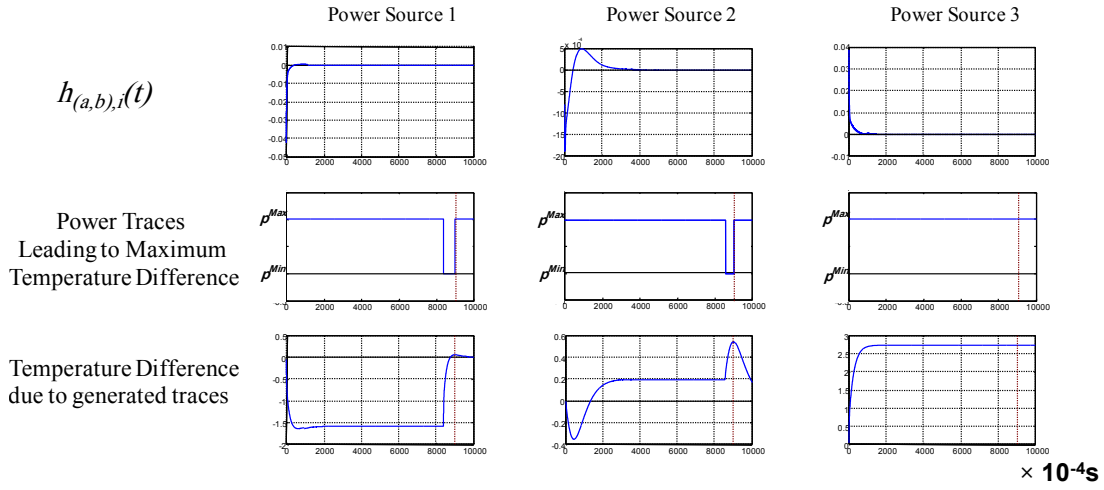


Figure 2.5: Generating the maximum temperature difference by constructing the proper power trace

simulation is used. If linear systems analytical methods are used to calculate the step response, then there will be no simulation overhead.

2.5.2 Sensor Placement

The sensor placement method we developed uses our analytical model for maximum on-die temperature differences. The experimental results show that it reduces the number of required sensors as compared to previous work.

Previous techniques such as [46] and [54] depend on calculation of the range of the hotspot which is the maximum distance r from the hotspot that a sensor can be placed while still maintaining the intended accuracy. This range has a circular form and is centered at the point of interest which limits the accuracy and the efficiency of such techniques. Some points which meet the accuracy requirements might be missed, as demonstrated on SoC with 6 *XScale*® cores shown in Figure 2.7. The two X's represent the points of interest to be monitored. The observability areas of the points of interest are shown by solid lines. Circular ranges of the hotspots are shown by dotted circles. Our sensor placement technique considers the observability area of a point of interest instead of its circular range as the

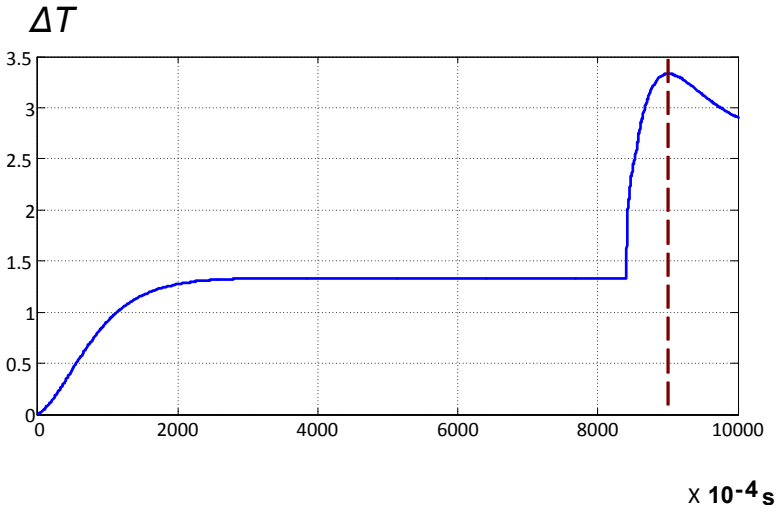


Figure 2.6: Temperature difference in the example of figure 2.5

potential location of a sensor. Therefore it can identify the point marked by * at the overlapping part of the observability areas to place a single sensor to monitor both points of interest. When using the circular ranges, this sensor location would not be identified since the ranges do not overlap; and therefore two separate sensors would be required. We evaluate our sensor placement algorithm for different values for the desired accuracies and compared it with techniques that uses circular ranges [46] . In [46] , only the sensor placement model is proposed and actual results on sensor placement are not provided. Moreover, the activity factor parameter (f_a) which is one of the important parameters in the proposed model in that work depends on the workload and the paper does not explain in enough details that how this parameter is calculated. Due to these reasons, we were not able to reproduce the actual sensor placement results of this work for comparison. Therefore, in order to be fair, we assumed the best results which this model could achieve. We found the largest possible circular range this model could have found. Then we used the outcomes of this model in our sensor placement algorithm. In other words, we are comparing our techniques to the best possible results the other model could achieve under the best conditions.

Table 2.2 shows the number of sensors required to monitor 8 locations of

Table 2.1: Errors in temperature difference simulations (°C)

	Positive			**Negative**		
Benchmarks	**Mean**	**Std. Dev.**	**Max.**	**Mean**	**Std. Dev.**	**Max.**
MiBench (Automotive)	1.09	0.78	4.34	1.01	0.72	3.77
MiBench (Network)	6.89	2.60	9.39	6.96	0.55	9.55
MiBench (Telecomm.)	6.02	2.53	9.16	6.36	2.47	9.29
MiBench (Mixed)	1.15	1.15	8.05	0.59	0.55	7.08
MediaBench (Mixed)	3.13	2.68	8.36	2.75	2.27	7.41

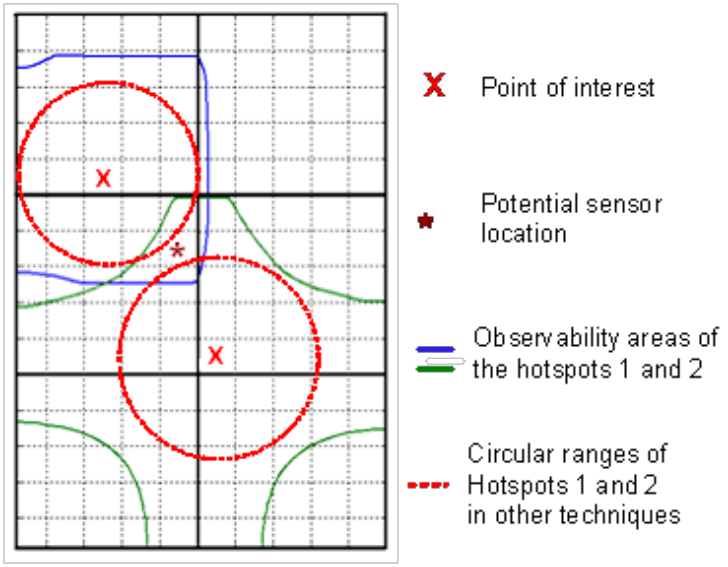


Figure 2.7: Using observability area vs. circular range

Table 2.2: Number of sensors needed by our technique and range-based methods

	SOC1						
Accuracy (°C)	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Our technique	7	7	6	5	5	5	4
Circular range [21]	7	7	7	7	6	6	6
	SOC2						
Accuracy (°C)	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Our technique	8	8	7	6	6	5	5
Circular range [21]	8	8	8	7	7	6	6

interest on SoC1 and 8 locations on SOC2 for specified maximum tolerable error between the temperature sensor and the actual temperature at the point of interest (Accuracy in Table 2.2).

As these tables show, our sensor placement technique is able to reduce the number of sensors needed more aggressively with lower desired accuracy than previous work [46] .

It should be mentioned that increasing the number and locations containing the points of interest will improve the efficiency of our technique. Our sensor placement technique minimizes the number of sensors by sharing a sensor among multiple points of interest. This technique takes advantage of the overlapping observability regions of the locations of interest in order to find the possible sharing of a sensor among various locations of interest. Increasing the number of the locations of interest increases the chances that observability regions of these locations overlap, so more sensors could be shared among various points of interests. This would help reduce the number of sensors.

2.6 Conclusion

Having a good estimate of maximum temperature variations across die is necessary for a number of applications such as placement of thermal sensors and

Table 2.3: Error statistics for limited number of sensors

SOC1				
	Temperature Measurement Error (°C)		**Temperature Estimation Error (°C)**	
Sensor No.	**Mean Absolute Error**	**Std. Dev.**	**Mean Absolute Error**	**Std. Dev.**
2	3.74	4.72	0.77	1.28
3	3.72	4.60	0.76	1.27
4	4.41	5.50	0.75	1.27
5	3.29	3.94	0.76	1.27
SOC2				
	Temperature Measurement Error (°C)		**Temperature Estimation Error (°C)**	
Sensor No.	**Mean Absolute Error**	**Std. Dev.**	**Mean Absolute Error**	**Std. Dev.**
2	5.61	6.02	1.26	2.35
3	4.17	6.41	1.35	2.26
5	4.34	5.76	1.12	1. 50
7	5.03	6.60	0.85	1.23

Table 2.4: Error statistics for different time steps

	SoC1					
	Sensor Measurement Errors (°C)			**Indirect Temperature Sensing Errors (°C)**		
Step Size (10^{-4}s)	**Mean Absolute Error**	**Std. Dev.**	**DTM Slowdown**	**Mean Absolute Error**	**Std. Dev.**	**DTM Slowdown**
128	2.91	4.04	40%	0.13	0.25	6%
512	3.09	4.33	41%	0.49	0.89	24%
1028	4.29	4.27	44%	0.97	1.24	28%
2056	3.89	5.08	43%	1.74	1.39	38%
	SoC2					
	Sensor Measurement Errors (°C)			**Indirect Temperature Sensing Errors (°C)**		
Step Size (10^{-4}s)	**Mean Absolute Error**	**Std. Dev.**	**DTM Slowdown**	**Mean Absolute Error**	**Std. Dev.**	**DTM Slowdown**
128	3.94	4.75	55%	0.54	0.62	9%
512	3.66	4.82	52%	1. 12	1.64	22%
1028	4.04	4.38	49%	1.77	2.15	34%
2056	3.87	4.51	51%	2.66	3.34	46%

analysis of reliability and thermal-induced clock skew and performance mismatch. Normally extensive simulations are used in order to determine these variations, thus incurring significant overhead. The method proposed here removes the simulation overhead by providing a model for accurate estimation of maximum temperature differences over various points across the die. It also identifies the conditions under which the maximum temperature differences and variations occur. This aspect of the model is helpful for generation of test data for stress tests and augmenting the benchmarks to check when maximum temperature differences occur in real life situations. Our experiments show that when using simulations, maximum temperature difference underestimation error can be as high as 9°C. Based on this analytical model, a novel design time technique for allocation and placement of thermal sensors is proposed which is able to guarantee a maximum sensor placement error. As compared to previously proposed methods, this technique can reduce the number of sensors needed by 16% on average while guaranteeing the specified sensor accuracy.

Chapter 2 in part, is a reprint of the material as it appears in Proceedings of the Great Lakes Symposium on VLSI, 2008. Sharifi, S. and Rosing, T. S. and IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems, 2010. Sharifi, S. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

Chapter 3

Indirect Temperature Sensing

3.1 Introduction

Previous chapter explained our design time technique for addressing inaccuracies due to sensor placement error. However, there are factors affecting the accuracy of sensors which cannot be addressed at design time. Variations in process parameters introduced during manufacturing can result in sensor reading inaccuracies (e.g. threshold voltage variation on the die) [65]. Errors are also introduced in the process of analog to digital conversion due to quantization, and due to limitations of design and technology. Changes in power supply voltage can affect sensor readings. Finally, the statistical characteristics of sensor inaccuracy change during the lifetime of the chip [83].

Recent work reports thermal sensor accuracy of around 1°C [50], however, this is achievable only through accurate calibration [64]. Many microprocessors use un-calibrated thermal sensors [64]. This is mainly due to the high cost and overhead associated with the thermal sensor calibration, particularly in the systems featuring multiple sensors. The calibration is usually done at test time and thus incurs overhead in design and test cost and silicon area [65]. It requires pre-heating and testing the sensors to detect various errors. Once these errors and sources of inaccuracy are known, the sensing unit is calibrated using A/D (analog to digital) converters and look-up tables.

Even well calibrated sensors are not capable of accurately reporting the

actual hot spot temperature on the die due to their location. Various sources of noise such as process variation and power variation further increase the inaccuracy of thermal sensors. Even when the average sensor error is low, the sensor might deliver single readings that are quite different from the actual temperature. If the readings from the sensors are directly used with complete trust, such variations may cause problems such as performance degradation due to early activation of DTM, or reliability degradation due to its late activation.

This chapter introduces *indirect temperature sensing*, a *run time* technique to address issues such as unavailability of enough sensors, degradation or failure of existing sensors and dynamic change of hotspot locations. Our technique accurately estimates the temperature at different locations on the die using noisy readings obtained from a few available sensors and power estimates of functional units. It also complements our design time technique by allowing a trade-off between hardware cost and computation cost. Now we can use fewer sensors at the cost of slightly increased computation at run time. Due to its low overhead it can be used for temperature aware scheduling and other online thermal management techniques. Indirect temperature sensing can be activated as needed rather than continuously; for example when the temperature at a unit on the chip is approaching a threshold. In this way it provides an easy trade off between accuracy and overhead. Finally, it can adapt to changes in measurement noise characteristics, which is very important since mean time to failure of thermal sensors is shorter than that of assets they are supposed to protect. Our indirect temperature sensing technique also shows an order of magnitude reduction in the standard deviation and maximum value of temperature estimation error relative to measured temperature values.

3.2 Related Work

One of the most widely used models for temperature estimation at micro-architectural level is HotSpot [32], which is based on building a multilayer thermal RC network of the given chip. The differential equations used to describe the heat

flow have a form similar to that of electrical current. This duality is the basis for the micro-architectural level thermal model of HotSpot which was proposed in [74] and further described in [33] and [32]. For more details on this model, please refer to appendix A.

Temperature modeling techniques such as HotSpot incur too high computation cost during run time. In [48], a technique is proposed which performs run time thermal simulation based on the observation that the average power consumption of architecture level modules in microprocessors is the major contributor to the variations in the temperature. Therefore piecewise constant average power inputs can be used to speed up the thermal analysis. Techniques such as those introduced in [48] need to continually perform temperature estimation, thus causing significant overhead. In addition, since these on-chip temperature estimation techniques are not combined with thermal sensor measurements, the estimates can easily deviate from the actual values. Due to these issues, temperature information for dynamic thermal management is usually obtained from thermal sensors at run time.

In [88], the authors address the problem of estimating the accurate sensor temperatures given noisy sensor readings. This work focuses on steady state temperature and doesn't consider transient temperature changes, which is necessary for dynamic thermal management techniques. Temperature is assumed to depend only on the current power consumption of the functional blocks and dependence of current temperature on previous temperature is ignored. In [16], the authors present a thermal characterization approach which reconstructs the thermal map of the die based on limited sensor measurements using spectral Fourier analysis techniques. This approach is able to reconstruct the thermal map using limited sensor data, but the effect of inaccurate and noisy sensors is not considered. The information about the thermal map is extracted only based on the thermal sensor readings and thermal dynamics of the chip. Power consumption of different cores are not exploited. None of the previously published techniques provide any results on their execution times to show practicality of these techniques for thermal estimation at run time.

The next section describes the details of our *indirect temperature sensing*

technique.

3.3 Components of Indirect Temperature Sensing

There are multiple sources of errors which cannot be effectively addressed at design time; sensors fail over time, their output is subject to noise, and even if the system did not have sensor failures and noise, the location of hotspots dynamically changes under different workloads. To address such run time issues, we propose indirect temperature sensing based on Kalman filtering which can accurately estimate the temperature at various locations on the die using inaccurate readings obtained from a limited number of noisy sensors. Our technique consists of a set of off-line setup steps shown in Figure 3.1.a., followed by run-time procedure shown in Figure 3.1.b.

The setup phase (Figure 3.1.a.) starts by creation of chip's equivalent thermal RC network using models described in [74, 33]. The linear dynamic system generated in this way is usually too large and too complex for an on chip software implementation. Model order reduction is used to generate a much smaller yet accurate system. Calibration is performed by applying KF to the reduced order model of the system. The calibration ends when the KF reaches its steady state. The resulting steady state KF is used during the normal operation to actually perform the temperature estimation (Figure 3.1.b.).

The KF estimates the temperature in a predict-correct manner based on inaccurate information of temperature and power consumption. Time update equations project forward in time with the current temperatures and the error covariance estimates to obtain a priori estimates for the measurement step. The measurement update equations incorporate the new measurements into the a priori estimate to obtain an improved a posteriori estimate of the temperature. Details of the technique are explained in the next subsection.

3.3.1 KF-based Temperature Estimation

Temperature values at different locations on the die depend on various factors, such as power consumptions of functional units, layout of the chip and the package characteristics. Analysis and estimation of temperature requires a thermal model which represents the relation between these factors and the resulting temperature. We use the same thermal RC network model as described in the appendix A.

The lumped values of thermal R and Cs represent the heat flow among units and from each unit to the thermal package. We model the temperature at grid cell level [32] which enables more accurate and fine grained temperature estimates. An analytical method is proposed in [32] to determine the proper size of a grid cell. The thermal network is represented in state space form with the grid cell temperatures as states and the power consumption as inputs to this system. The outputs of this state space model are the temperatures at the sensor locations which can be observed by sensor readings ($S(t)$). We define C_t and G_t as thermal capacitance and thermal conductance matrices, D as the input selection matrix which identifies the effect of power consumptions at current time steps on the temperature at next time step and F as the output matrix which identifies the sensor grid cells at which temperatures are observable. u is the vector of power consumption values for different components on the die and T is the vector of temperature values at different grid cells. The units for temperature and power are centigrade degree and Watt.

As explained in the appendix A, the system is described by equation (A.2). Based on this equation, the system with the thermal sensors can be represented as:

$$\begin{aligned} &\tfrac{d}{dt}T(t) = -C_t^{-1}G_tT(t) + C_t^{-1}Du(t) \\ &S(t) = FT(t) \end{aligned} \tag{3.1}$$

Since sensor measurements can be inaccurate and exact power consumption of each functional unit at run time is not available, we use Kalman filter (KF) for temperature estimation. The KF uses a form of feedback control to estimate a

process in a predict-correct manner with time and measurement update phases. Time update equations project forward in time the current state of the system and the error covariance estimates to obtain a priori estimates for the measurement step. The measurement update equations incorporate the new measurements into the a priori estimate to obtain an improved a posteriori estimate.

We use Kalman filtering to both estimate the temperature and to filter out any thermal sensor noise. In order to apply the KF to our model, we convert the continuous time differential equations in (3.1) to corresponding discrete time equations in (3.2). Here H, J and F are the state matrix, input matrix and output matrix of the system respectively. Furthermore, at time *n, T[n], u[n]* and $S[n]$ are the state vector representing temperatures at different grid cells, input vector of functional block power consumption and output vector of temperatures at sensor locations respectively.

$$\begin{aligned} T[n+1] &= H\,T[n] + J\,u[n] \\ S[n] &= F\,T[n] \end{aligned} \tag{3.2}$$

Accurate estimation of power consumptions of each component at each time step is not practical in run time. On the other hand, [48] shows that most of the energy in the power traces is concentrated in the DC component. The trend of temperature variations is determined by the average power over a period of time. This is especially true for power traces with very large DC components and smaller high frequency harmonics [48]. Based on this fact, we use the average power consumption of each component as an estimation of the actual power consumption at that time.

Introduction of noise due to inaccuracies of modeling the process, $w[n]$, and the measurement noise, $v[n]$, enables us to rewrite the system formulation as:

$$\begin{aligned} T[n+1] &= H\,T[n] + J\,u[n] + Gw[n] \\ S_v[n] &= F\,T[n] + v[n] \end{aligned} \tag{3.3}$$

The time-update equations for our system are given below. Here $\check{T}$*[n|n-1]* represents the estimate of $T[n]$ given the past measurements up to S_v*[n-1]*, $\check{T}$*[n|n]*

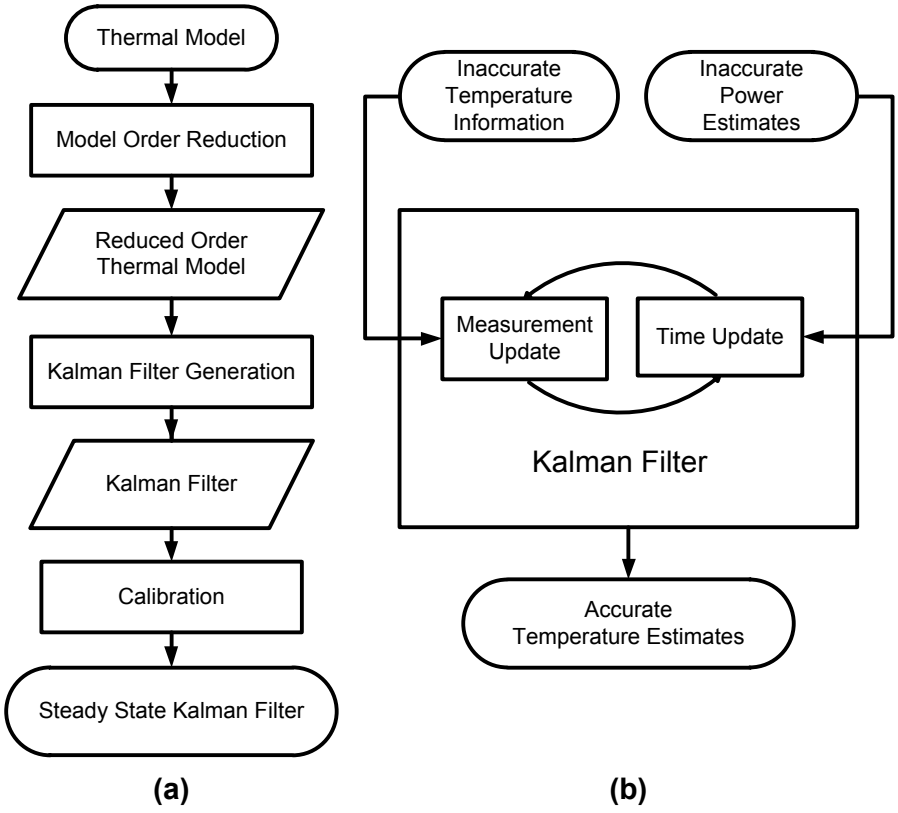


Figure 3.1: Proposed technique. (a) Offline setup (b) Run time temperature estimation by KF

is the updated estimate based on the last measurement $S_v[n]$ and P is the error covariance matrix.

$$\begin{aligned} \breve{T}[n+1|n] &= H\,\breve{T}[n|n] + J\,u[n] \\ P[n+1|n] &= HP[n|n]H^T + GQ[n]G^T \end{aligned} \tag{3.4}$$

Given the current estimate $\breve{T}[n|n]$, the time update predicts the state value at the next sample $n+1$ (one step ahead). Then the measurement update adjusts this prediction based on the new measurement $S_v[n+1]$. The measurement update equations for this system are:

$$\breve{T}[n|n] = \breve{T}[n|n-1] + M[n](S_v[n] - F\breve{T}[n|n-1])$$

$$M[n] = P[n|n-1]\,F^T(R[n] + FP[n|n-1]F^T)^{-1} \tag{3.5}$$

$$P[n|n] = (I - M[n]F)\,P[n|n-1]$$

M is called Kalman gain or innovation gain. It is chosen to minimize the steady state covariance of the estimation error given the noise covariance $Q = E(w[n]w[n]^T)$ and $R = E(v[n]v[n]^T)$. Computational complexity of the KF is $O(k^3)$ due to the matrix inversion in calculating Kalman gain $M[n]$,where$k$is the size of the dynamic model. The next subsection describes the methods we exploit in order to speed up the run time computation.

3.3.2 Reducing Computational Complexity

We introduce two techniques that significantly reduce the computational complexity of our model. One of the techniques reduces the size of the model used in KF, while the other reduces the number of computations required for KF.

Steady State Kalman Filtering

The time scale at which the sensor noise characteristics change is much larger than the time scale at which we study the system (months or years compared to seconds). Thus we assume the system and noise covariances are time-invariant.

As a result, we can use steady state KF in which it is not necessary to compute the estimation error covariance or Kalman gain in real time [72]. The steady state KF reduces the computational overhead from $O(k^3)$ to $O(k^2)$ while still providing good accuracy [72]. A calibration step is needed prior to run-time operation in order to get the KF to steady state. Running the KF during the calibration step makes gain and covariance matrices converge to constant values. In our experiments, we show that use of steady state KF reduces the computational complexity to several orders of magnitude without significant effect on accuracy.

Model Order Reduction

The model order reduction enables us to find a low-dimensional but accurate approximation of the thermal network which preserves the input-output behavior to a desired extent. We use a projection based implicit moment matching method (PRIMA) [59] which is used to find a mapping from the high-dimensional space of the given state-space model to a lower dimensional space. In this technique, Krylov subspace vectors are used instead of moments. For a square matrix A of dimension N and a vector b, the subspace spanned by the vectors $[b, Ab, \ldots, A^{q-1}b]$ is called a Krylov subspace of dimension m generated by $\{A, b\}$ and is denoted by $Kr(A, b, q)$.

With thermal capacitance and conductance matrices represented by C_t and G_t respectively, the circuit formulation shown in equation (A.2) can be represented in this form:

$$sC_tT = -G_tT + Du \tag{3.6}$$

The reduced order model is generated using congruence transformation, where $C_r = V_q^T C_t V_q, G_r = V_q^T G_t V_q, D_r = V_q^T D, X_r = V_q^T X$:

$$sC_rT = -G_rT + D_ru \tag{3.7}$$

The projection matrix $V_q = \{V_1, V_2, \ldots, V_q\}$ is obtained by Arnoldi process such that

$$Span\{V_1, V_2, ..., V_q\} = Kr\{-G_t^{-1}C_t, DU\} = \\ Span\{DU, -G_t^{-1}C_tDU, ..., (-G_t^{-1}C_t)^{q-1}DU\} \tag{3.8}$$

and

$$v_i^T v_j = 0 \quad for\, all\, i \neq j \\ v_i^T v_i = 1 \quad for\, all\, i$$

This approach matches moments up to order q. The larger the number of matched moments, the closer is the behavior of the reduced order model to the original system, but at the cost of higher processing time. Because of the moment-matching properties of Krylov-subspaces, the reduced transfer function will agree with the original up to the first q derivatives on an expansion around some chosen point in the complex plane (usually $s = 0$). In addition, due to the congruence transformation, the reduced model inherits the structure of the original model, which means the passivity is preserved. Interested reader can refer to [59] for a detailed discussion of the technique.

There are other model order reduction techniques which are designed for linear circuits with multiple sources. For example, EKS [82] can match higher moments compared to PRIMA in multiple-input multiple-output systems, but imposes limitations on the inputs and, more importantly, incurs higher computational overhead. Our experimental results show that PRIMA works well for our applications since our network consists of only simple linear elements (Rs and Cs). Moreover, the topology of the network is such that it operates as a low pass filter which eliminates the high frequencies of the inputs. Therefore PRIMA with a few moments around frequency $s = 0$ provides sufficient accuracy and acceptable overhead compared to RHS-model order reduction methods like EKS [82]. The effectiveness of PRIMA is shown in Table 3.1.

3.3.3 Detecting Sensor Failure and Degradation

As explained in the previous sub-sections, one of the techniques we use for real time realization of the technique is the use of steady state Kalman filters.

Steady state Kalman filter is optimal when assuming stationary noise. The fact that the noise characteristics of the thermal sensors do not change rapidly, makes steady state Kalman filter applicable in our problem. Sensor degradations or failures which cause these changes usually happen at the order of months. While steady state Kalman filtering works well for the stationary noise characteristics, it is not guarantied to work well under non-stationary noise. Therefore the changes in the characteristics of the sensor noise affect the accuracy of indirect temperature sensing. Here we address this problem by proposing a technique to detect the variations in the characteristics of sensor noise in order to detect the sensor degradation or failure before they affect the accuracy of the technique. Then these variations are addressed by adapting the indirect temperature sensing to these variations.

The steady state Kalman filter is completely dependent on the characteristics of the noise. Therefore, a Kalman filter generated for a certain set of noise characteristics might not work well for a different set. When a change is detected, the calibration phase of the technique is performed again and a new steady state Kalman filter is generated based on the current noise characteristics. In order to detect these variations, we use a hypothesis test called sequential probability ratio test (SPRT) which is specifically designed for sequentially collected data. It allows us to detect the variations in the statistical characteristics of the estimation error. An alternative would be setting thresholds on mean of the errors, but this can detect the variations only after they have happened. SPRT is able to detect the abrupt changes as threshold-based techniques, but it is also able to detect slow gradual variations which evolve over a long period of time [39]. It has been shown in [23] that a hypothesis test based on the SPRT is optimal in the sense that for given false and missed alarm probabilities α and β, it results in the lowest possible false or missed alarms. As stated before, α and β are the probabilities of false positives and false negatives respectively.

SPRT is based on a pair of hypotheses, H_0 and H_1 which are null hypothesis and alternative hypothesis respectively. The null hypothesis (H_0) states that the estimation errors are drawn from a distribution with mean zero and standard

deviation of σ. The alternate hypothesis states that the estimation errors are drawn from a distribution with mean μ and standard deviation of σ. In other words, H_0 assumes no change in the characteristics of the noise, while H_1 implies variations in these characteristics. The decision between these two hypotheses is based on the cumulative sum of the log-likelihood ratio (Λ_{i}):

$$S_{i+1} = S_i + \ log\ \Lambda_i \tag{3.9}$$

H_1 is accepted if $S_i \geq b$, H_0 is accepted if $S_i \leq a$, otherwise the monitoring continues, where a and b are chosen as:

$$a = \log(\frac{\beta}{1-\alpha})$$

and

$$b = \log(\frac{1-\beta}{\alpha})$$

where α is the false alarm probability, which is the probability of accepting H_1 when H_0 is true, and β is the missed alarm probability, which is the probability of accepting H_0 when $H_1$is true. α and β are decided in advance by the user.

Likelihood ratio is the ratio of the maximum probability of a result under two different hypotheses. In other words, likelihood ratio can be calculated as the maximum probability of H_0 (the estimation errors are drawn from a distribution with mean zero and standard deviation of σ) to H_1 (estimation errors are drawn from a distribution with mean μ and standard deviation of σ) given the current observations.

$$\Lambda_n = \frac{\Pr(\varepsilon_1, ..., \varepsilon_n|H_1)}{\Pr(\varepsilon_1, ..., \varepsilon_n|H_0)} \tag{3.10}$$

when n is the number of observations and

$Pr(\varepsilon_1, \ldots, \varepsilon_n|H_0)$ and $Pr(\varepsilon_1, \ldots, \varepsilon_n|H_1)$ are the probability of observing sequence $\varepsilon_1, \ldots, \varepsilon_n$ under H_0 and H_1 respectively.

Assuming independent observations, we can write:

$$\Lambda_n = \frac{\prod_{i=1}^{n} \Pr(\varepsilon_i|H_1)}{\prod_{i=1}^{n} \Pr(\varepsilon_i|H_0)} = \prod_{i=1}^{n} \frac{\Pr(\varepsilon_i|H_1)}{\Pr(\varepsilon_i|H_0)}$$

and

$$\log \Lambda_n = \sum_{i=1}^{n} \log \frac{\Pr(\varepsilon_i|H_1)}{\Pr(\varepsilon_i|H_0)}$$

When operating at run time mode, the technique monitors the temperature estimation error ε (difference between the estimated and observed temperature). With every new observation, equation (3.10) is calculated and the new log likelihood ratio is compared against the thresholds $a$and b. If the new log likelihood ratio exceeds b, SPRT reports degradation, while if the likelihood ratio gets under a, SPRT assumes the sensor is working fine. The changes are detected before they are large enough to affect the indirect temperature sensing results. Due to extremely low rate of sampling required in this technique (due to the extremely slow changes in the noise characteristics), we are not concerned about the overhead of the technique, but for estimation errors which are normally distributed, [39] proposes a compact expression with very low computational overhead. As stated previously, the probabilities α and β are defined by the user. When a decision is made about the H_0 or H_1, the technique starts at $n = 1$ with current observation. For our experiments we used $\alpha = 0.01$, $\beta = 0.0001$ and sampling interval of 1 minute.

In the next section we verify the proposed technique for indirect temperature sensing and also detection of sensor failure and degradation.

3.4 Experimental results

Our experimental setup for the *indirect temperature sensing* method and the technique for detection of sensor degradation and failure is the same as the setup explained in section 2.5.

3.4.1 Indirect temperature sensing

Unlike design time techniques discussed above, our method for accurate indirect temperature sensing addresses run time issues such as limited number of available sensors, their degradation or failure, dynamic changes in location of hotspots, etc. Indirect temperature sensing requires an off line setup step which has been implemented in Matlab, while the run time part of the algorithm is implemented in $C++$ on XScale or SPARC processors. Temperature values are obtained by running HotSpot [13] in grid mode with XScale power measurements as inputs. The temperature values of the grid cell containing the sensors are observable, while the temperature at other grid cells are assumed to not be observable and must be estimated using our technique. For our experiments, we used a 18×12 grid for SoC1 and a 20×20 grid for SoC2.

One of the advantages of using PRIMA model order reduction technique for indirect sensing is that the size of reduced model depends only on the number of power sources and the number of matched moments, not the number of grid cells. Therefore, increasing the granularity of the grid in order to increase the accuracy does not result in higher computational overhead.

To find the appropriate number and size of the grid cells, the analytical method presented in [32] is used .

No specific sensor technology is assumed in this work. The readings from the temperature sensors are used as starting temperature values for our model. Gaussian noise has been superimposed on the actual temperature values to model the inaccuracies of real thermal sensors. Processes generating noise are assumed to be stationary between off-line calibrations.

Figure 3.2 shows that the estimated temperature values closely follow the actual temperature at the location of interest on SoC1. Accurate estimates of the temperature are important to prevent early or late activation of DTM techniques due to sensor noise and errors. Indirect temperature sensing is very accurate in estimating temperature at locations far away from a limited number of sensors available on the die as shown in Table 2.3. It should be noted that the quality of estimation using Kalman filter depends on many factors such as workload charac-

teristics, number and location of the sensors and the characteristics of noise. For example, having more sensors located closer to the locations of interest would result in better estimation. For our application, as our experiments in Table 2.3 show, even with a very few sensors, our technique is able to estimate the temperature with reasonable accuracy.

For SoC1, we estimate the temperature at 6 locations of interest using only 2, 3, 4 and 5 sensors. For SoC2, we estimate the temperature at 8 locations of interest using 2, 3, 5 and 7 sensors. Each sensor is equidistant to the hotspots it must cover. In order to reduce the inaccuracies due to step size and model reduction, the step size is chosen to be $10ms$ and 3 moments are matched. The mean absolute error and its standard deviation are reduced by up to an order of magnitude.

Another important parameter affecting both accuracy and computational requirements of our technique is the time step at which temperature sensors are read and KF is applied. Table 2.4 shows statistics of measurement and estimation errors for different sizes of time steps used on SoC1 and SoC2. The basic time step is chosen at $10^{-4}s$ and multiplied by powers of 2. For this experiment, in order to reduce the inaccuracies due to sensor model order reduction, three moments are matched. One sensor monitors each location of interest, but this sensor is placed at an arbitrary location around the hotspot to show that the technique does not depend on the relative position of the sensor and the location of interest. Step size between $10ms$ and $100ms$ provide reasonable accuracy.

Table 3.2 also shows the effect of the indirect temperature sensing technique on the performance impact of dynamic thermal management techniques. It compares the relative slowdown of a DTM technique when it is driven by temperature values read directly from the sensors and temperature estimates of our indirect temperature sensing. The DTM technique used here is a threshold based DTM technique which reduces the voltage/frequency of a core when its temperature exceeds the threshold of 75°C.

Slowdown is compared by looking at the number of instructions executed per second at high utilizations. As this table shows, our indirect temperature

Table 3.1: Effect of number of matched moments on temperature estimation Error

Sensor	**No. of Matched Moments**	**SoC1**		
		Model Size	**Mean Absolute Error**	**Std. Dev.**
Measurement Error (°C)	-	-	3.38	4.82
Estimation Error (°C)	1	6	0.88	1.24
	2	12	0.75	1.05
	3	18	0.68	0.90
	4	24	0.48	0.88
Sensor	**No. of Matched Moments**	**SoC2**		
		Model Size	**Mean Absolute Error**	**Std. Dev.**
Measurement Error (°C)	-	-	5.64	6.60
Estimation Error (°C)	1	7	1.78	2.28
	2	14	1.53	1.79
	3	21	1.24	1.67
	4	28	0.69	0.92

Table 3.2: Effects of sensor degradation and failure

	SoC1			
	Estimation Error		**DTM Slow-down**	**Detection time by SPRT**
Sensor Degradation Type	**Mean Absolute Error**	**Std. Dev**		
Gradual mean change (2°C/Year)	0.59	0.61	19%	˜15 Weeks
Gradual mean change (1°C/Year)	0.53	0.67	24%	˜8 Weeks
Bias (1°C)	1.15	1.60	40%	˜1.5 Hours
Bias (0.5°C)	0.85	1.10	26%	˜2 Hours
Stuck to constant value	1.47	2.63	54%	<1 Hour
	SoC2			
	Estimation Error		**DTM Slow-down**	**Detection time by SPRT**
Sensor Degradation Type	**Mean Absolute Error**	**Std. Dev**		
Gradual mean change (2°C/Year)	1.35	1.55	25%	˜17 Weeks
Gradual mean change (1°C/Year)	1.33	1.45	24%	˜10 Weeks
Bias (1°C)	1.86	2.54	36%	˜2 Hours
Bias (0.5°C)	1.79	2.35	35%	˜3 Hours
Stuck to constant value	2.89	3.47	48%	<1 Hour

sensing technique can reduce the slowdown due to DTM by more than 6X. This is due to the fact that more accurate temperature estimates reduce the number of false positives in triggering the DTM technique which reduces the slowdown due to DTM.

This figure also shows that since larger step sizes reduce the accuracy of the indirect temperature sensing, the reduction in DTM slowdown caused by indirect sensing reduces as step sizes get larger.

Table 3.1 shows how matching different number of moments affects the accuracy of our technique. The step size is set at $50ms$. Size of the reduced model is the number of functional units times the number of matched moments. For SoC1 example, since there are 6 power consuming functional units, matching 4, 3, 2 and 1 moments reduces the model size to 24, 18, 12 and 6 respectively. As it is shown in the table, after matching the second moment, further moment matching does not significantly improve accuracy. Matching two or three moments provides sufficient accuracy for most applications. As explained in the previous sections, an important step toward online realization of indirect temperature sensing is using steady state Kalman filtering.

We next analyze the overhead of our indirect temperature estimation technique. During the calibration process, i.e. before the KF reaches its steady state, we use general KF. Since calibration can be performed offline, the overhead of the general KF does not affect performance. The number of calibration steps, in our case less than 100, depends on the thermal network and noise characteristics. The run time overhead is shown in Figure 3.3. The cost of using regular KF is compared with the steady state KF used by online portion of our indirect temperature sensing technique. As can be seen in Figure 3.3, the performance overhead of steady state KF is significantly lower. This difference grows for larger models. Using model order reduction and steady state KF allow performing indirect temperature sensing even on *XScale*® on which floating point instructions have to be emulated due to the lack of floating point units.The run time overhead on *XScale*® can be further reduced by using a fixed point implementation, or running the technique on a processor that has a floating point unit. The reduction in overhead due to use

of floating point unit is clearly shown in Figure 3.3(b) where the execution times are reported on a SPARC processor. As the figure shows, for the MPSoCs used in our experiments, the run time overheads are in the order of 100us.

3.4.2 Detecting sensor failure and degradation

Here we show how using a steady state Kalman filter while the noise characteristics are not stationary can affect the accuracy of the results. To prevent such inaccuracies, we use a sequential hypothesis test called SPRT which was described in the previous sections. This allows us to detect the meaningful changes in the sensor noise characteristics before they affect the accuracy of the technique.

Figure 3.4 shows how SPRT technique detects the meaningful changes in the estimation errors due to the changes in the characteristics of the noise and triggers adaptation of the technique to address the new changes.

Thermal sensors are usually based on temperature sensitive diodes, and according to [38], the most common faults in diodes are open, short and degradation. Open and short failures in the diode cause constant sensor readings which according to test community, we call it stuck at fault. We tried stuck at faults with various values around the average expected sensor reading. Using this technique, all of these failures were detected in less than one hour.

Another common fault in these diodes is degradation which means the deviation of parameters from the expected values. In order to emulate subtle incipient sensor degradations, we apply a gradual change in the average of the error which changes this average by a couple of degrees in a year. α and β values are set to 0.01 and 0.0001 respectively. This will result in a threshold of 100 for the likelihood ratio. The y axis shows the likelihood ratio (Λ_i) and the threshold for deciding about the degradation.

As shown in the figure, the likelihood ratio has passed the threshold at about two months. This triggers an alarm showing degradation of the sensor and the indirect sensing is signaled to adapt itself to this change and address this degradation by regenerating the steady state Kalman filter.

We assume different cases of sensor degradation or failure by adding some

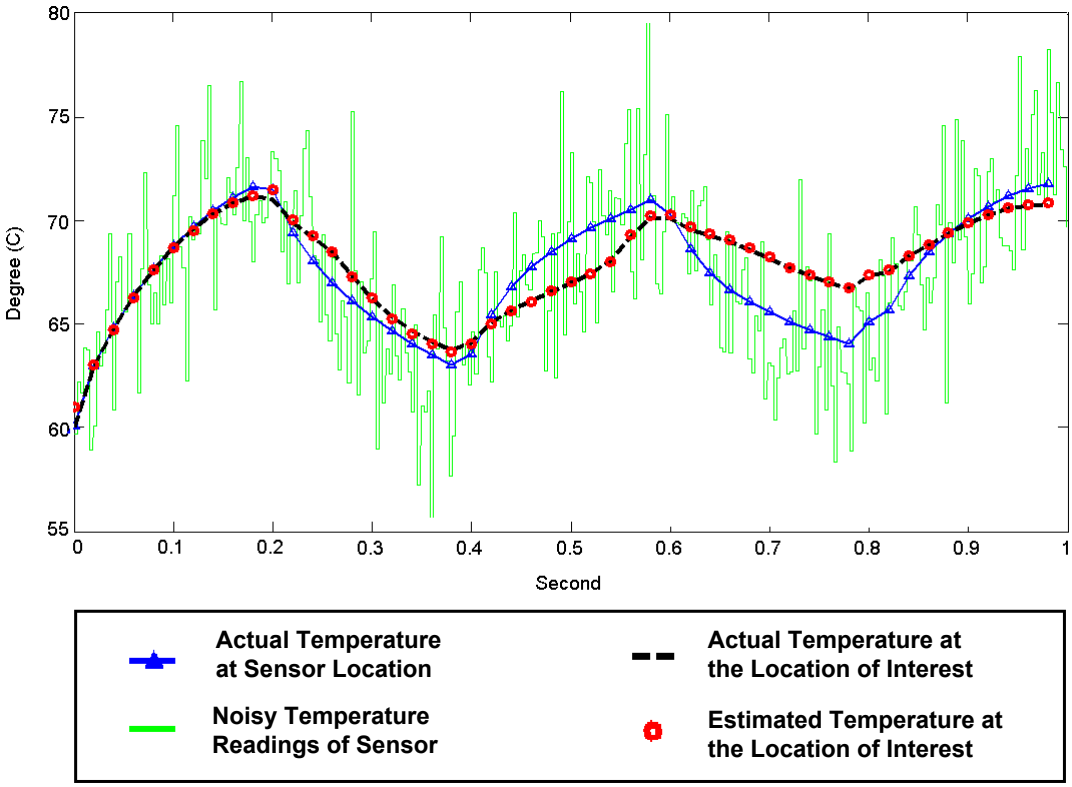


Figure 3.2: Comparison of sensor, actual and estimated temperatures

error to the sensed temperature.

These error types include an error signal whose mean changes gradually and very slowly from 0 to 1 or 2°C in a year. Another error introduced is a constant bias (0.5°C and 1°C) in the sensor readings. Another type of failure considered is stuck at error which means reading the same value from the sensor irrespective of the actual temperature.

Table 2.4 shows how such sensor degradations will affect the estimation error and DTM slowdown before these sensor degradations are detected and addressed by regenerating the steady state KF. It also shows the time it takes to detect these degradations. As in the example shown in Figure 3.4, the sampling time of the sensor degradation technique is 1 minute and α and β values are set to 0.01 and 0.01. As the table shows, for the gradual changes in the error characteristics, the error is detected early before it affects the accuracy of the technique. For example, for a gradual mean change of 1°C/Year, the error is detected after about two months before it can cause significant DTM slowdown. For more significant and abrupt changes, such as bias values or stuck at errors which cause significant effect on DTM, the technique is able to detect the error in a matter of hours or so. This quick detection of more severe degradations minimizes the effect of these inaccuracies on the DTM slowdown. Another important point to note is that these detection times can be reduced with more frequent sampling. The cost will be more frequent computations for SPRT which usually are not significant since the technique can be implemented by accessing lookup tables. Using equation (24) in general cases implies two lookups in the look up table, one division, one logarithm operation and one addition. For the normally distributed observations, [29] suggest a compact expression which can be implemented by just two additions for each sample.

3.5 Conclusion

In this section, a method is proposed for accurate software estimation of temperature at different locations on the chip based on the inaccurate values ob-

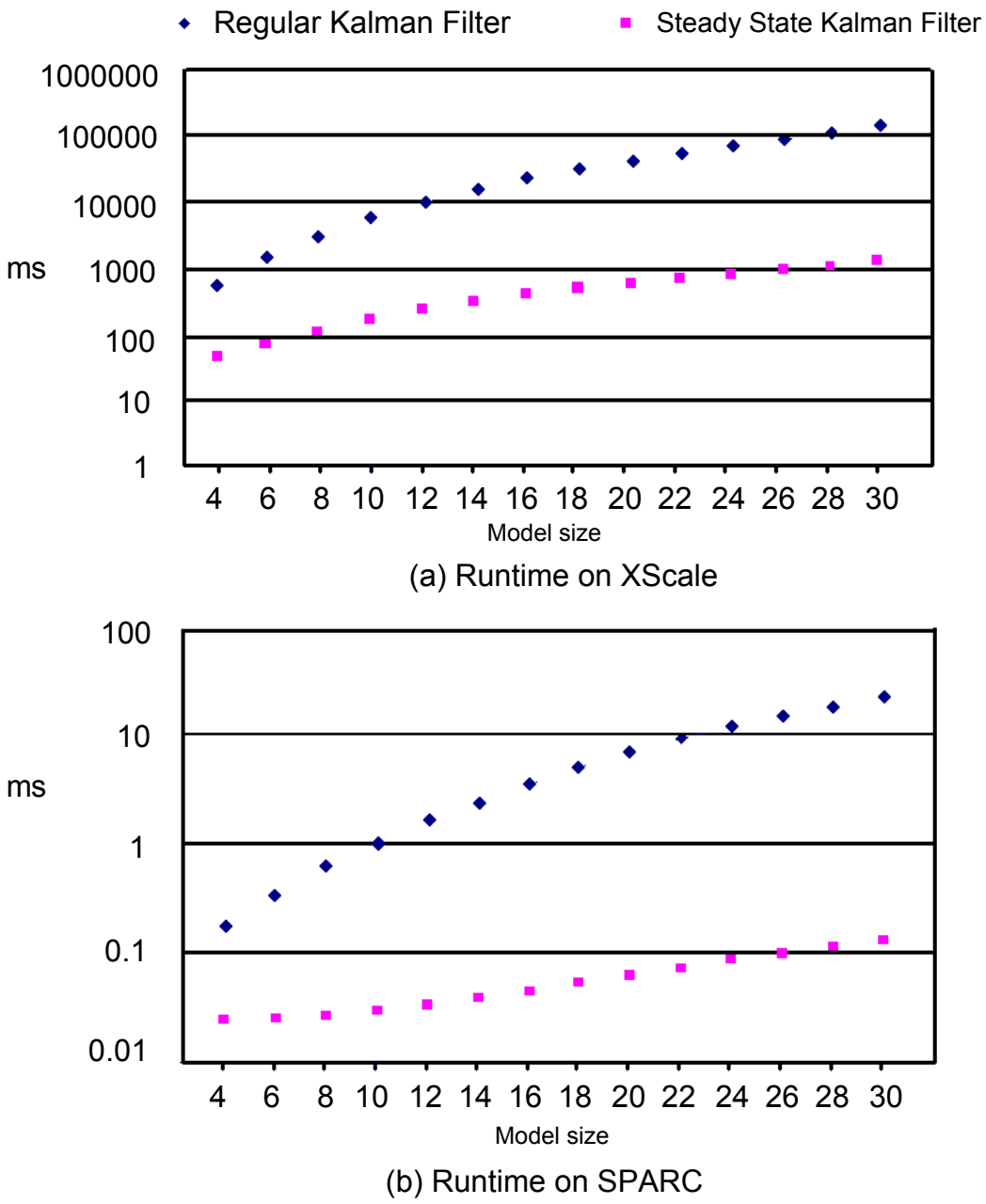


Figure 3.3: Run time of the technique on (a) *XScale*® (b) SPARC®

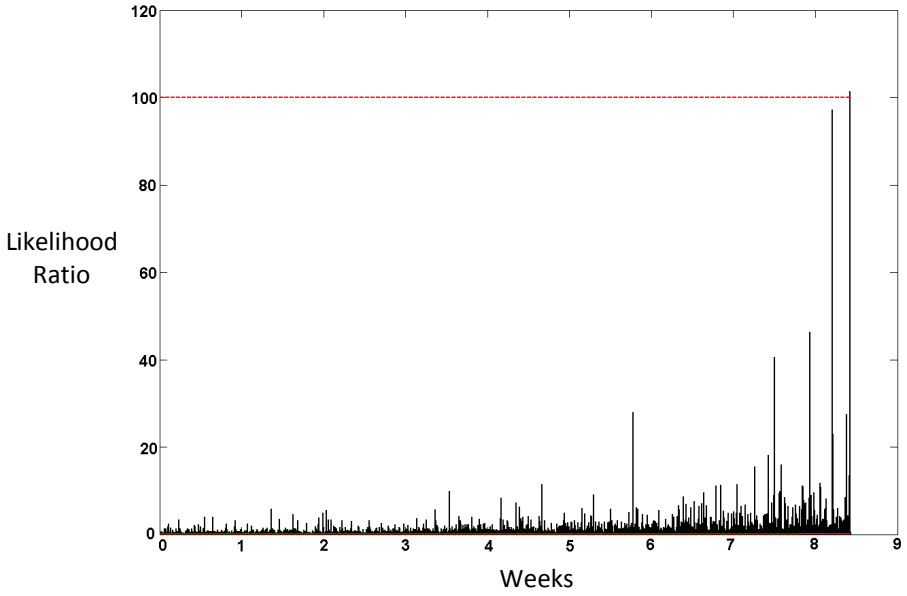


Figure 3.4: SPRT technique to detect sensor degradation

tained from a few on-chip temperature sensors. Most importantly, our technique can be used efficiently to estimate the temperatures at the locations of interest where no physical sensor is available. One important advantage of this method is that it can be activated only when the temperature is approaching a limit to provide more accurate estimates. Kalman Filter is used for state estimation and to eliminate the noise of on-chip temperature sensors. In order to reduce the complexity, a model order reduction technique is applied. To further improve the efficiency, steady state KF is used that is calibrated off-line. Our temperature estimation technique incurs very low run-time overhead in orders of hundreds of microseconds which is needed in OS level schedulers that run in millisecond time scales. Although this technique addresses sources of inaccuracy at runtime, new sensor degradation and failure can still introduce inaccuracy in the results. To prevent this, a method is proposed for early detection of sensor degradation and failure which triggers the calibration of indirect temperature sensing. The experimental results show that using *indirect temperature sensing*, the mean absolute

error and the standard deviation of the error are minimized about an order of magnitude as compared to information obtained from direct reading of sensors.

Chapter 3 in part, is a reprint of the material as it appears in International Symposium on Quality Electronic Design, 2008. Sharifi, S. Liu, C. and Rosing, T. S. and IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems, 2010. Sharifi, S. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

Chapter 4

Tempo Temperature Prediction

4.1 Introduction

A large number of thermal management techniques proposed in the past have been reactive in the sense that they take action when temperature of a unit rises above a given threshold. This might cause significant performance loss and reliability issues [75]. Therefore, proactive temperature management techniques have been proposed which try to predict and prevent thermal emergencies before they happen. Even if a thermal emergency happens, it may be less severe and can be resolved by less aggressive techniques with lower performance overhead. Proactive thermal management techniques typically rely on temperature predictors in order to estimate the future temperature of the functional units or cores to be able to make more intelligent thermal management decisions.

In this chapter, we propose a temperature prediction method which can be used to evaluate the thermal impact of the selecting among a set of power states. This technique, which is called *Tempo*, has various advantages over previous temperature predictors First, it does not need any kind of runtime adaptation to assure accurate prediction. Moreover, because most previous temperature prediction methods rely on general signal analysis and estimation techniques, they do not do well with varying temperature trends due to changes in the power state of a core. They need to wait until the thermal effect of the power state change appears in the temperature so that the predictor can detect the new trend. Therefore,

they are not able to evaluate the thermal impact of future power state changes before they happen. In contrast, *Tempo* predicts the temperature based on the characteristics of the thermal model and is able to predict temperature of any set of future power states before they are applied to the system. This capability enables accurate upfront evaluation of future thermal impact of potential power state changes caused by scheduling decisions in an MPSoC. Moreover, methods relying on signal estimation techniques typically treat temperatures of different cores as independent signals. Therefore, in the cases where mutual thermal effects of the cores are significant, inaccuracies in the estimates might be significant.

This chapter describes details of *Tempo* predictor, while the next chapter explains how we use *Tempo* in our framework for proactive thermal management of heterogeneous MPSoCs.

4.2 Related work

Several temperature predictors have been proposed to be used for proactive thermal management techniques. In [19] a predictive temperature balancing technique based on autoregressive moving average (ARMA) modeling has been proposed for general purpose systems. In order to avoid inaccuracy, the changes in the workload and temperature dynamics are detected using a sequential probability ratio test (SPRT) so that the ARMA model can be updated in a timely fashion. In systems with highly dynamic workloads, continuously performing SPRT-based detection and updating the ARMA model incurs overhead.

In [40], a temperature prediction technique for chip multiprocessors is proposed which assumes future temperature of a cores as the linear extrapolation of its previous temperature readings. Although this predictor does not need any kind of runtime adaptation, because it implicitly assumes that the gradient of temperature stays the same for the whole prediction interval, it may result in inaccurate estimates as it does not consider power state changes.

In [10] a proactive dynamic thermal management is introduced for chip multiprocessors based on the band-limited property of temperature frequency spec-

trum. Although this method does not need any adaptation either, it is not able to accurately predict temperature changes due to power changes before they actually happen. This technique predicts the temperature by observing the previous samples and projecting them into future assuming the trend does not change. Therefore, when the power state of a core changes, this technique needs to observe at least a few samples before being able to estimate the future temperature. Thus it cannot be used to evaluate potential thermal impact of alternative future scheduling decisions.

Previous temperature prediction techniques either need costly runtime adaptation (e.g. [19]) or are not able to accurately predict the thermal impact of transition to new power states before before the power change happens (e.g. [10]). The general signal analysis and prediction approaches used in these techniques are more suitable for the cases where the underlying physical model of the system is not well known. Being oblivious to this knowledge of the system, the advantages of these techniques are limited when the underlying model of the system is known because they are not getting the full benefit of this valuable information at hand.

In contrast to previous work, we introduce *Tempo*, a novel temperature prediction technique which allows accurate evaluation of future thermal impact of potential scheduling decisions without having to measure their effect on temperature after applying them. In contrast to previous techniques, our prediction method takes advantage of the knowledge of the dynamics of the system which is provided by the thermal model. Unlike previous techniques, *Tempo* can accurately predict what the future temperature of the cores can be for any future power states of the cores without the need to apply them and wait to see their effect on temperature. Therefore, it can be used to evaluate the thermal effects of alternative decisions for thermal management to choose the best out of potential options. Moreover, our proposed model does not need any runtime adaptation and also is fully linear.

Tempo can be used within model predictive control techniques as well. Thermal management techniques using model predictive control such as [84] typically assume the temperature at all of the nodes of the thermal network are observ-

able by thermal sensors, which is not true for majority of designs because thermal sensors are not placed in thermal interface material and heat sink. Instead, *Tempo* can be used in such cases in order to avoid such assumptions, especially because it is completely linear so there is no need for non-linear optimization techniques. It can also be easily integrated with multi-parametric optimization techniques, as we have used in *TempoMP* which is explained later. *Tempo* can also be used with phase detection and prediction techniques such as [35], [36] which are able to monitor and predict power consumption of the cores. Although availability of power estimates increases prediction accuracy of *Tempo*, even when there are no accurate values of future core power consumptions, *Tempo* is able to estimate future temperature based on only previous temperature information.

4.3 Temperature Prediction

In this section, we first explain the basic ideas behind our *Tempo* temperature predictor and then we describe how *Tempo* can take into account the leakage and its temperature dependence. The objective of our prediction method is to accurately predict future temperature of the cores based on the available temperature and power information. More specifically, we estimate the temperature of the cores at the end of the scheduling tick $(k+1)$ based on the temperature of the cores at the beginning of scheduling tick $k+1$ and the one before that (k). *Tempo* takes into account the power state changes between scheduling ticks $k+1$ and k as well.

Our work is based on compact thermal model of the chip [32] which is described in detail in Appendix A. It leverages the well-known duality of thermal and electrical phenomena. The heat flow among the functional units is modeled using a corresponding network of thermal capacitances and resistances as shown in Figure A.1. The dynamics of the temperature and the relation between the temperature, power consumption of the cores and thermal characteristics of the system is described as:

$$C_t \frac{d}{dt} T(t) = -G_t T(t) + P(t) \tag{4.1}$$

where the vectors and matrices are defined as:

T	Temperature at all the nodes of the thermal network
P	Power consumptions of the nodes of thermal network
G_t	Thermal conductance matrix
C_t	Thermal capacitance matrix

Assuming the number of the cores to be n and the number of nodes in the thermal RC network to be m, P and T are vectors of length m both and G_t and C_t are matrices of size $m \times m$ both.

Given the temperature at all the nodes of the thermal network, estimating the future temperature based on the power is not difficult. However, usually this is not the case. At runtime, temperature is usually obtained via thermal sensors within the silicon layer. If each core does not have its own sensor, the technique in [70] can be used to estimate the core temperatures using the available sensors. However, thermal sensors cannot be placed within internal layers (thermal interface material, heat spreader, etc.), so the available temperature information is limited to the temperature of the cores. This lack of thermal information of internal nodes makes it challenging to predict or evaluate temperature at runtime. The temperature of the internal nodes can be obtained by simulating the thermal model at runtime, which is not computationally practical. Moreover, without feedback from the thermal sensors, the results may deviate dramatically from the actual values.

In our formulation, to reflect this lack of thermal information of the internal nodes, we break the vector of temperature values (T) into two sub-vectors. Sub-vector T_o represents the temperatures observable by thermal sensors (core temperatures), while T_u represents the internal nodes of the thermal network whose temperatures are unobservable (as shown in Figure A.1). The size of these vectors are n and $m - n$ respectively.

The analytical solution to the non-homogeneous system of differential equa-

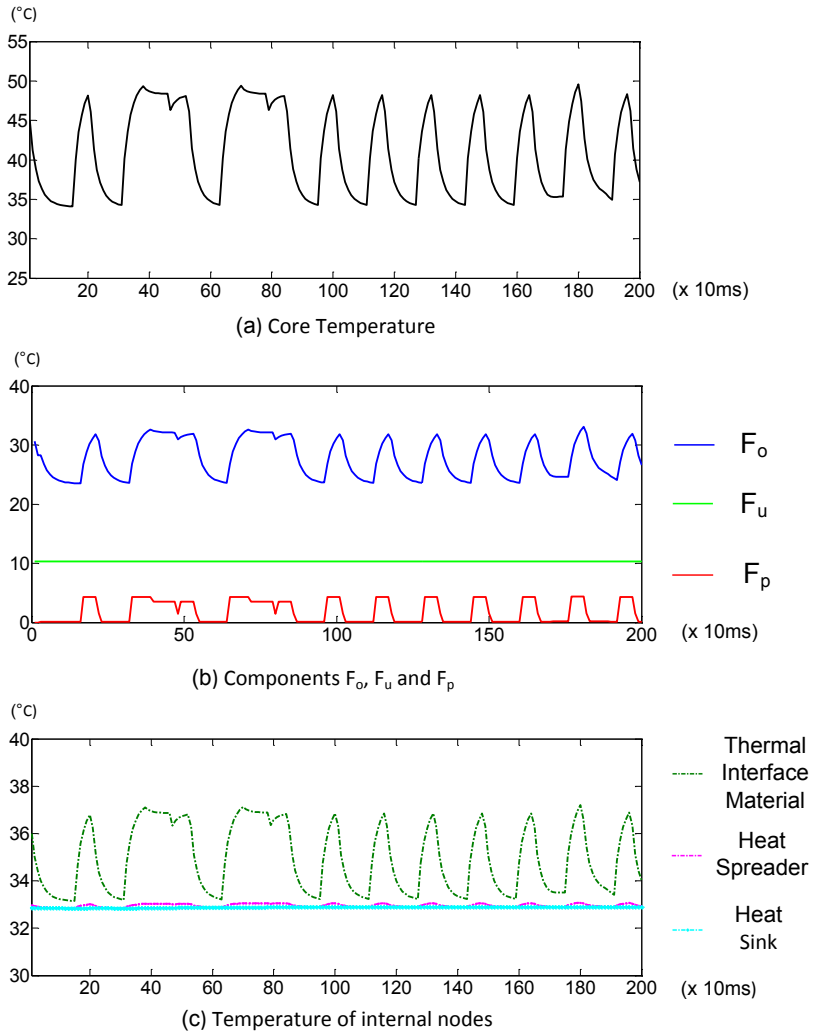


Figure 4.1: (a) Temperature of the core (b) Breakdown of temperature into components of equation (4.7) (c) Temperature of corresponding nodes in thermal interface material, heat spreader and heat sink, all relative to ambient

tions in equation (4.1) can be calculated as:

$$T(t) = e^{-\Gamma(t-t_0)}T(t_0) + \int_{t_0}^{t} e^{-\Gamma(t-\tau)}C_t^{-1}P(\tau)\,\mathrm{d}\tau \tag{4.2}$$

where $T(t_0)$ is the starting temperature at time t_0 and

$$\Gamma = C_t^{-1}G_t \tag{4.3}$$

Discretizing the equation (4.2) for a scheduling tick of t_s, the temperature at consequent scheduling ticks can be described as:

$$\begin{bmatrix} T_o[k+1] \\ T_u[k+1] \end{bmatrix} = \Psi \begin{bmatrix} T_o[k] \\ T_u[k] \end{bmatrix} + \Phi \begin{bmatrix} P[k+1] \\ P_u[k+1] \end{bmatrix} \tag{4.4}$$

where

$$\Psi = e^{-\Gamma t_s}, \Phi = \Gamma^{-1}(I - e^{-\Gamma t_s})C_t^{-1} \tag{4.5}$$

The first term in equation (4.4) is the contribution of initial conditions and the second term is the contribution of power consumption during this scheduling tick. We divide the matrices Ψ and Φ into sub-matrices as shown here:

$$\Psi = \left[\begin{array}{c|c} \Psi_{oo} & \Psi_{uo} \\ \hline \Psi_{ou} & \Psi_{uu} \end{array}\right], \Phi = \left[\begin{array}{c|c} \Phi_{oo} & \Phi_{uo} \\ \hline \Phi_{ou} & \Phi_{uu} \end{array}\right] \tag{4.6}$$

where sizes of the matrices Ψ_{oo}, Ψ_{uo}, Ψ_{ou} and Ψ_{uu} are $n \times n$, $n \times m-n$, $m-n \times n$ and $m-n \times m-n$ respectively. Each matrix Ψ_{xy} shows the effect of initial temperature of set x of nodes on the current temperature of the set y. For example, Ψ_{uo} models the effect of initial temperature of *unobservable(u)* nodes on the current temperature of *observable(o)* nodes. Similarly, each matrix Φ_{xy} shows the effect of current power of the set x of nodes on the current temperature of the set y.

The internal nodes do not consume any power ($P_u[k+1] = 0$), so:

$$T_o[k+1] = \underbrace{\Psi_{oo}T_o[k]}_{F_o} + \underbrace{\Psi_{uo}T_u[k]}_{F_u} + \underbrace{\Phi_{oo}P[k+1]}_{F_p} \tag{4.7}$$

where the first and second terms (F_o and F_u) are respectively the contributions of initial temperature of the observable and unobservable nodes on the temperature at the next scheduling tick. The third term (F_p) is the contribution of the power consumption of the cores during the incoming scheduling tick. At each scheduling tick, the term F_o can be calculated based on the current temperature of the cores which are observable by thermal sensors. The term F_p also can be calculated given the current power consumption of the cores. But due to lack of knowledge about the temperature of unobservable nodes ($T_u[k]$), term F_u is unknown. This term

represents the contribution of initial temperature of the unobservable nodes of the thermal network (internal nodes) on the temperature at the next scheduling tick. Because F_u is not known at runtime, equation (4.7) is not enough to calculate future temperature of the cores.

We show that fast changes in the temperature are produced by components F_o and F_p while component F_u does not change quickly. We find the upper bound on the rate of change of F_u. Figure 4.1 shows an example of breakdown of the temperature of a SPARC core in floorplan of Figure 4.3 into three components of the equation (4.7). It should be noted that while the core temperature shown in part (a) of the figure is the sum of the components in Figure 4.1(b). All values are relative to the ambient temperature (40°C in this case). As can be seen in this figure, although the core temperature changes significantly, the term F_u changes very slowly. The quick change of temperature between two scheduling ticks is mainly due to changes on the other two terms (F_o and F_p) as shown in the figure. Figure 4.1(c) shows the temperature of the corresponding internal nodes. It should be noted that as this figures shows, although some of the unobservable internal nodes might change quickly (e.g. thermal interface material), the changes of term F_u are very slow. Later in this section we will explain the reason for this phenomenon in detail.

Due to the limited rate of change on F_u, we assume that this term remains constant between two consecutive scheduling ticks:

$$F_u[k+1] \approx F_u[k] \tag{4.8}$$

Temperature evaluation equation for the previous scheduling tick is then:

$$T_o[k] = \Psi_I T_o[k-1] + \Psi_{II} T_u[k-1] + \Phi_I P[k] \tag{4.9}$$

Based on equation (4.8), $\Psi_{II} T_u[k] \simeq \Psi_{II} T_u[k-1]$. $\mathbb{T}$, the temperature prediction by *Tempo* is then calculated as:

$$\mathbb{T}[k+1] = (\Psi_{oo} + I) T_o[k] - \Psi_{oo} T_o[k-1] + \Phi_{oo}(P[k+1] - P[k]) \tag{4.10}$$

We use equation (4.10) to predict the temperature at the beginning of scheduling tick $k+1$ based on the temperature of the cores at the beginning of scheduling

tick k and their power state during this scheduling tick. We also define *Tempo*'s thermal state of a core, $\mathcal{T}[k+1]$ in equation (4.11) as the predicted temperature at the beginning of scheduling tick $k+1$ if the power state of the cores do not change in scheduling tick k.

$$\mathcal{T}[k+1] = (\Psi_{oo} + I)T_o[k] - \Psi_{oo}T_o[k-1] \tag{4.11}$$

As equation (4.11) shows, $\mathcal{T}[k+1]$ can be calculated based on the previous and current core temperatures. Later we use *Tempo*'s thermal state within our scheduling techniques. Based on equation (4.11) we rewrite the equation (4.10) as:

$$\mathbb{T}[k+1] = \mathcal{T}[k+1] + \Phi_{oo}(P[k+1] - P[k]) \tag{4.12}$$

The second term on the right hand side reflects the effect of power changes on the temperature of the cores.

Now we describe how we incorporate the leakage power and its temperature dependence into *Tempo* prediction. We use the linear approximation of the leakage as suggested in [49] with an approximate estimation error of up to 5%. Using this model, the leakage power of a core can be estimated as sum of a constant term and a term linearly dependent on the cores temperature:

$$P_{leak}(t) = LT(t) + Q \tag{4.13}$$

where L is a diagonal matrix containing the coefficients for the linear terms and Q is a vector of constant terms for different cores. It should be noted that elements of L and Q which correspond to the nodes in any layer other than silicon are zero because these nodes do not consume any power, including leakage power. Therefore, the equation (4.1) is transformed to:

$$C_t \frac{d}{dt}T(t) = -(G_t - L)T(t) + (P_{dynamic}(t) + Q) \tag{4.14}$$

The next steps would be similar to deriving equations (4.2) to (4.12) based on equation (4.1). Predicted temperature considering leakage is:

$$\begin{aligned} \mathbb{T}[k+1] = \; & \big((\Psi'_{oo} + I)T_o[k] - \Psi'_{oo}T_o[k-1]\big) \; + \\ & \Phi'_{oo}(P_{dyn}[k+1] - P_{dyn}[k]) \end{aligned} \tag{4.15}$$

where $\Psi = e^{-C_t^{-1}(G_t - L)t_s}$, $\Phi = ((G_t - L)^{-1}C_t)(I - e^{-C_t^{-1}(G_t - L)t_s})C_t^{-1}$.

Before describing how we leverage *Tempo* in our scheduling framework, which is explained in Section 5.3, in the following subsection we show why *Tempo* is an accurate predictor and provide a theoretical upper bound on the prediction accuracy of *Tempo*.

4.3.1 Theoretical Analysis of *Tempo*

Tempo is based on the premise that the changes on component F_u of temperature are very slow, or $F_u[k+1] \approx F_u[k]$. This phenomenon can be explained by the structure of the thermal RC circuit and thermal characteristics of the chips. To simplify the explanation, we assume that the length and width of heat spreader and heat sink exactly match those of the silicon (and thermal interface material).

As shown in Figure A.1, each node in the thermal network is connected to its neighbor nodes in the same layer through lateral thermal conductances and to the corresponding nodes in the top and bottom layers through vertical thermal resistances. If the nodes at the bottom and top of node i are respectively represented by $bottom(i)$ and $top(i)$ and the set of lateral neighbors of the node i are represented by $LN(i)$, then thermal conductance matrix G_t could be described as:

$$G_t(i,j) = \begin{cases} i = j & \sum_{l \in LN(k)} g(i,l) + g(i, top(i)) + g(i, bottom(i)) \\ i \neq j & -g(i,j) \end{cases} \tag{4.16}$$

where $g(i,j)$ is the conductance and capacitance between nodes i and j in the thermal circuit. Capacitance matrix C_t is also defined as:

$$C_t(i,j) = \begin{cases} i \neq j & 0 \\ i = j & c(i) \end{cases} \tag{4.17}$$

where $c(i)$ is the thermal capacitance of node i. As can be seen, matrix G_t is symmetric and diagonally dominant, and matrix C_t is diagonal. Based on equations

(4.16) and (4.17):

$$\Gamma(i,i) = c(i)^{-1}\left(\sum_{l\in LN(k)} g(i,l) + g(i,top(i)) + g(i,bottom(i))\right) \tag{4.18}$$

Considering matrix Γt_s, its eigendecomposition is:

$$\Gamma t_s = V\Lambda V^{-1} \tag{4.19}$$

where V is a square matrix whose jth colum is the jth eigenvector (v_j) of Γt_s and the Λ is a diagonal matrix whose diagonal elements are the corresponding eigenvalues of matrix Γt_s; in other words: $\Lambda(j,j) = \lambda_j$. From equations (4.5) and (4.19), we have:

$$\Psi = e^{-\Gamma t_s} = Ve^{-\Lambda}V^{-1} = VXV^{-1} \tag{4.20}$$

Since Λ is diagonal, X is also a diagonal matrix with diagonal elements:

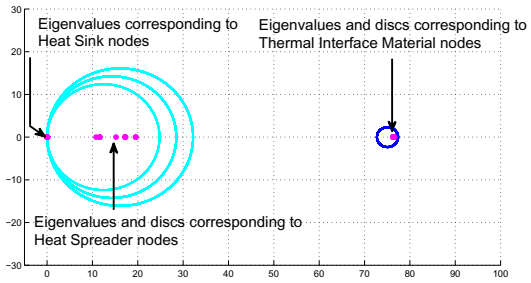
$$X(j,j) = e^{-\lambda_j} \tag{4.21}$$

According to Gershgorin's circle theorem [26], the eigenvalues of matrix Γt_s lie within a set of discs called Gershgorin discs. Gerschgorin disc i is centered at $\Gamma(i,i)t_s$ with radius $\rho(i)$. $\rho(i)$ is defined as $\rho(i) = min(\rho_R(i), \rho_C(i))$ where

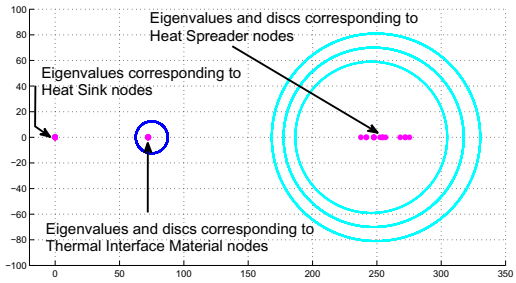
$$\rho_R(i) = \textstyle\sum_{j\neq i} |\Gamma(i,j)t_s| \quad \rho_C(i) = \sum_{j\neq i} |\Gamma(j,i)t_s| \tag{4.22}$$

For diagonally dominant matrices, usually the eigenvalues tend to be closer to the centers of the discs. For example, in the extreme case of a diagonal matrix, the eigenvalues fall right onto the center of Gershgorin discs. Using equations (4.18) and (4.22), Gershgorin discs and the location of the eigenvalues can be estimated directly based on the chip and package parameters without the need to calculate the exact eigenvalues.

For an MPSoC composed of a 3×3 grid of cores of size $1 \times 1mm$ each, Figure 4.2 shows the eigenvalues and Gershgorin discs of matrix Γt_s corresponding to the nodes in thermal interface material, heat spreader and heat sink for two different types of packages. The first one is the default package in Hotspot [4] which is a high end package with thermal interface material thickness of $0.02mm$,



(a) A high end package



(b) An embedded-type package

Figure 4.2: Gershgorin discs of matrix Γt_s in complex plane for (a) a high end package and (b) an embedded-type package

heat spreader thickness of $1mm$, heat sink thickness of $6.9mm$ and convection resistance of $0.1K/W$. The other one resembles a lower end and less expensive package which can be found in embedded type devices. It has the same thermal interface material thickness, but with heat spreader thickness of $0.1mm$, heat sink thickness of $1mm$ and convection resistance of $5.0K/W$.

In both cases, the eigenvalues of matrix Γt_s corresponding to heat sink (shown in Figure 4.2) are very close to zero and their corresponding Gershgorin discs are too small to be seen in the figure. In both types of packages, the eigenvalues corresponding to the thermal interface material and heat spreader are much larger compared to those of heat sink, although the heat spreader eigenvalues are very different in these two cases. As a result, in matrix Ψ, according to equation (4.21) the eigenvalues corresponding to heat spreader and thermal interface mate-

rial are negligible compared to the ones of the heat sink. Therefore, the effect of the temperature of these nodes on component F_u is negligible relative to the effect of the heat sink. Consequently, although the temperature of thermal interface material and heat spreader can change quickly and significantly, the effect of these changes on the component F_u is negligible. In contrast, the temperature of heat sink which is the dominant component of F_u changes very slowly. Its changes are negligible in the milliseconds range. Therefore, F_u's changes between two consecutive scheduling ticks can be neglected.

Upper bound of* Tempo*'s prediction error: The analytical bound on *Tempo*'s prediction error is found based on the characteristics of the thermal RC network and the power characteristics of the cores. The thermal network described by equation (4.1) is a linear time invariant (LTI) system since thermal resistances and capacitances are linear components. Equation (4.7) represents accurate temperature for the next scheduling tick while equation (4.10) is *Tempo*'s estimate of the temperature at the next scheduling tick. Temperature prediction error is the difference between these two values. Our goal is to find an upper bound on $\mathcal{E}$ defined below:

$$\mathcal{E} = \mathbb{T}[k+1] - T_o[k+1] = \Psi_{uo}(T_u[k] - T_u[k-1]) \tag{4.23}$$

The term $(T_u[k] - T_u[k-1])$ is the difference between value of $\hat{T}$ at times kt_s and $(k-1)t_s$ where t_s is the length of the scheduling interval. The difference between temperature of the nodes of thermal network at times t_0 and t_1 is $\Delta T(t_0, t_1) = T(t_1) - T(t_0)$ which can be calculated based on the equation (4.2) as:

$$\Delta T(t_1, t_0) = -(I - e^{-\Gamma(t_1 - t_0)})T(t_0) + \int_{t_0}^{t_1} e^{-\Gamma(t_1 - \tau)}\Lambda P(\tau)\,\mathrm{d}\tau \tag{4.24}$$

ΔT is contributed by two components which we call ΔT_i and ΔT_p which are defined as:

$$\Delta T_i(t_1, t_0) = -(I - e^{-\Gamma(t_1 - t_0)})T(t_0), \tag{4.25}$$

$$\Delta T_p(t_1, t_0) = \int_{t_0}^{t_1} e^{-\Gamma(t_1 - \tau)}\Lambda P(\tau)\,\mathrm{d}\tau. \tag{4.26}$$

These two components are completely independent because ΔT_i depends only on the initial temperature at the scheduling interval while ΔT_p only depends on the power consumption in that scheduling interval. The first component, ΔT_i, depends on the initial temperature of the thermal RC network and has a negative contribution to the ΔT. Intuitively, if no power is applied to a system with an initial temperature higher than ambient, its temperature will decrease gradually. The higher the initial temperature is, the larger the decrease. On the other hand, the second component, ΔT_p, is always non-negative and has a positive or zero contribution to the ΔT because the power consumption of a core is always non-negative. If the system's initial temperature is equal to ambient temperature and the system is running, its temperature will increase. Because of the negative contribution of ΔT_i and non-negative contribution of ΔT_p, in order to maximize the positive value of ΔT, we should maximize ΔT_p while minimizing the absolute value of ΔT_i which can be done independently.

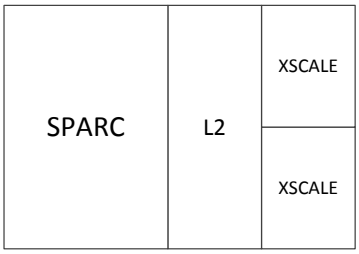
The only variable part of equation (4.26) is P. Therefore, in order to maximize ΔT_p, we should maximize P: $P = P_{max}$. At the same time we should minimize the absolute value of ΔT_i. The lower the initial temperature is, the lower absolute value of ΔT_i will be. The minimum value of $T(t_0)$ will result in the minimum absolute value of ΔT_i. We can set the lowest value of initial temperature to ambient temperature. Similarly, to maximize the value of ΔT in negative direction, we should minimize ΔT_p while maximizing the absolute value of ΔT_i. ΔT_p can be minimized by setting the power to the minimum ($P = P_{min}$) and ΔT_i can be maximized by using the maximum possible initial temperature $T(t_0)$. In this way, we have the upper bound on the absolute value of the temperature difference between the two consecutive scheduling ticks. Then, given equation (4.23), the upper bound on the absolute value of *Tempo*'s prediction error can be found. In our experiments, the maximum error observed is less than 0.5°C.

4.4 Experimental results

The cores used in our experiments are a low-power in-order architecture similar to the SPARC cores in UltraSPARC T1 [47], and very low power cores designed for embedded systems, similar to Intel's XScale [34]. Power, performance, and area characteristics of the cores are shown in Figure 4.3. We assume that the MPSoC is implemented in 65 nm technology. The areas of the cores are derived from published photos of the dies after subtracting the area occupied by I/O pads, interconnection wires, interface units, L2 cache, and control logic as in [41], and scaled to 65nm. Each L2 cache has 1MB size, 2 banks, 64-byte lines, and is 4-way associative. Using CACTI [31], the area and power consumption of the caches at 65nm are estimated as $14mm^2$ and 1.7W, respectively. The cache power consumption value includes leakage.

For performance and power data, the M5 Simulator [11] is used with Wattch [13] power model updated with 65nm model parameters. The in-order pipelines of SPARC and Xscale are modeled by modifying M5's execution engine. We assume the same three voltage settings for the XScale and SPARC cores. For XScale, we use the existing available frequency levels (as reported in [34]), and for SPARC we set the default frequency to 1.2GHz (as reported in [47]), and scale frequency using the 95% and 85% settings as in [36]. The overhead of switching to a new voltage/frequency is set to $50\mu s$. These power values are then utilized in the temperature simulations. We compute the leakage power of CPU cores based on structure areas and temperature. We compute temperature dependence using the model introduced in [30] with the same constants mentioned in the paper for 65nm. For power overhead during frequency scaling, we use the power of the higher power state. The overhead of migration of the tasks between the cores is assumed to be $10\mu s$ [27].

We use HotSpot Version 4.2 [4] for thermal modeling with a sampling interval of $100\mu s$ to ensure sufficient accuracy. In many embedded systems such as cell phones there is no heat sink or spreader. To model this within HotSpot, we set the spreader thickness to be very thin- $0.1mm$. The heat sink is replaced by a package



(a) Floorplan of MPSoC

Thermal Parameters	
Die Thickness	0.5 mm
Convection Resistance	7 °K/W
Convection Capacitance	100 J/°K

(b) Thermal parameters

Processor		XScale-like	SPARC-like
Issue width		1	2
Area (mm^2)		1.4	5
Frequency Settings (MHz)	1.2 V	624	1200
	1.18 V	416	1140
	1.06 V	208	1000
Average Dynamic Power (W)	1.2 V	0.34	2.40
	1.18 V	0.32	2.10
	1.06 V	0.28	1.90

(c) Characteristics of the cores

Figure 4.3: Characteristics of the MPSoC

with thermal parameters shown in Figure 4.3(b) which are within the ranges suggested by [37] and [53]. The parameters used in HotSpot are summarized in Figure 4.3(b). It should be noted that this is only one example, while our techniques are general and apply to a wide range of systems with various characteristics.

The workloads in our experiments consist of integer benchmarks provided in MiBench benchmark suite [29] which include automotive/industrial, network and telecommunications applications. Other than datasets provided in MiBench suite, we use datasets provided by [25]. To evaluate our technique under various conditions, we create moderate to intensive workloads consisting of varying number of tasks from MiBench suite. Instances of each task are generated regularly at every arrival period. We set deadline (d) and (τ) of the tasks to twice the execution time of that task at the slowest frequency on the slowest core (XScale). This way

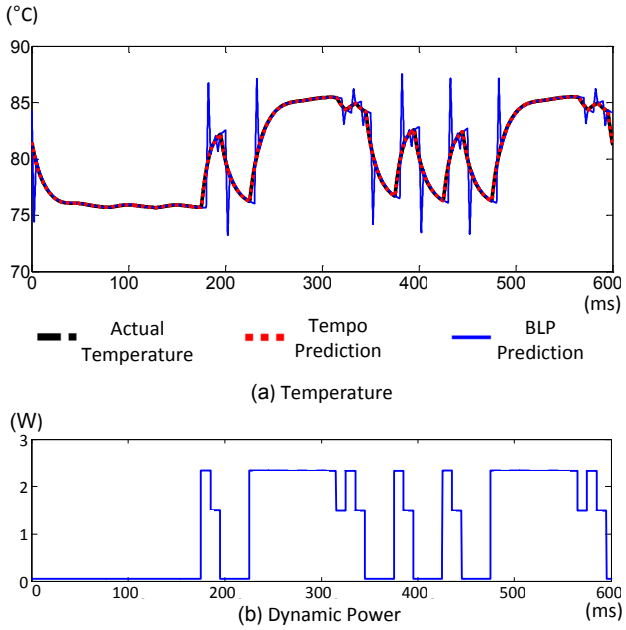


Figure 4.4: Comparison of Tempo and *BLP* predictor [10]

the tasks can meet their deadlines irrespective of the core type they are assigned to. First we compare *Tempo* with a state of the art temperature predictor called *Band-Limited Predictor (BLP)* [10]. It utilizes the band limited nature of temperature frequency spectrum to predict the temperature trend. Similar to *Tempo*, the coefficients used in calculations of *BLP* are also calculated at design time. No training phase is required. We use the same parameters used as in [10], namely $\alpha = 0.135$, $m = 3$ and $N = 3$.

Figure 4.4(a) shows the actual temperature of a core on a MPSoC along with the prediction results of *Tempo* and *BLP*, while Figure 4.4(b) shows the trace of dynamic power applied to the same core. The sharp changes in the dynamic power are caused by power state changes happening at scheduling ticks which are 10 ms apart. This is a slice of a longer trace of execution of the MiBench benchmarks so the cores have high initial temperatures.

As shown in the figure, as long as the temperature and power changes are smooth, both predictors do well. However, *BLP* fails when the power state of the core changes significantly. For example, in Figure 4.4(a), right after the first power state change at around $190ms$, *BLP* underestimates the temperature. This is because *BLP* relies exclusively on the temperature history and trend. Therefore,

even if the new power state is known, *BLP* cannot predict the temperature before the new power state is applied and has impacted the temperature trend. Moreover, as the figure shows, even after the first sample of temperature signal is observed after the change in temperature trend, *BLP* significantly overestimates the temperature. However, *Tempo* accurately predicts the future temperature given the next power state of the cores. As shown in this figure, the maximum prediction error of *BLP* can be over 5°C, while in our experiments, the maximum temperature prediction is always less than 0.5°C - an order of magnitude difference. *BLP* and any other predictors which depend only on temperature trend cannot accurately evaluate the thermal effects of scheduling decisions and power state changes. In contrast, *Tempo* can be efficiently used to evaluate alternative decisions regarding scheduling and power state changes. The results of our temperature aware scheduling techniques in the next section further illustrate the efficiency of *Tempo*.

4.5 Conclusion

In this chapter, *Tempo*, a novel temperature prediction method is introduced. Previous temperature prediction techniques mainly depend on general signal analysis and prediction techniques and predict the future temperature only based on the temperature history. Unlike those techniques, *Tempo* takes advantage of the knowledge of the physical characteristics of the thermal system of the chip. Using this knowledge, it accurately takes into account the potential thermal effects of any power state changes. This capability allows *Tempo* to reduce the maximum temperature prediction up to an order of magnitude compared to previous state of the art predictors. Moreover, *Tempo* makes it possible to evaluate the thermal safety of potential scheduling decisions before applying them to the system. This is a very useful capability for proactive temperature aware scheduling. The next chapter describes how we integrate *Tempo* at the heart of a dynamic thermal management framework, called *PROMETHEUS*.

Chapters 4 in part, is a reprint of the material accepted for publication at

Design, Automation and Test in Europe (*DATE*) 2012. Sharifi, S. Ayoub, S. and Rosing, T.S. and and the material under submission at IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems. Sharifi, S. Krishnaswamy, D. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

Chapter 5

PROMETHEUS Framework for Temperature-aware Scheduling on Heterogeneous MPSoCs

5.1 Introduction

Power density and temperature are continuously increasing due to scaling of device dimensions and higher power consumptions in modern processors. This results in system reliability degradation, leakage power increase, performance degradation and higher cooling and packaging costs. According to [61], more than 50% of all integrated circuit failures are related to thermal issues. Temperature has become one of the major factors in design, manufacturing and test of modern processor systems. One of the approaches taken to alleviate thermal issues in modern processors has been introduction of multi-core processors which achieves similar throughput with lower power density. Many of these Multi-Processor System-on-Chip (MPSoC) systems integrate cores of various types on the same die.

Heterogeneous MPSoCs provide trade-offs regarding performance, power and temperature by allowing customization of performance and power characteristics of the chip to match the requirements of the workload [41], [42]. They are used in a broad range of applications from cell phones (Qualcomms Snapdragon

platform) to wireless base stations (Mindspeeds Transcede 4000 processors for 4G wireless base stations). Temperature is of particular concern in heterogeneous MPSoCs due to the inherent imbalance in heat generation patterns across the die. They are used in many embedded systems which experience a wider range of environmental conditions than typically seen in data centers and offices. One example is the case of wireless base stations deployed outdoors in harsh environmental conditions with ambient temperatures even exceeding 80°C [77]. Electronic systems in automotive applications must operate at ambient temperatures which might exceed 125°C [60]. Cell phones also must be able to operate under a wide range of ambient temperatures e.g. Alaska in winter vs. Sahara in summer, without the benefit of more sophisticated packaging due to cost and space considerations. For such systems, on-chip thermal and power management techniques are key.

In this chapter we propose a framework called *PROMETHEUS* to address dynamic thermal management (DTM) of heterogeneous MPSoCs running embedded workloads. Our methodology allows systematic thermal management of MPSoCs considering individual performance, power and temperature characteristics of each core. Although we focus on the more complex case of heterogeneous cores, our technique is also applicable to homogeneous MPSoCs. Embedded systems often run known sets of workloads which can be pre-characterized in terms of power and performance. Given this information, the performance requirements of the workloads are estimated at runtime and used to find the optimal power states of the cores on an MPSoC which can provide the required performance under thermal limits.

PROMETHEUS framework provides two temperature aware scheduling techniques which proactively avoid power states leading to future thermal emergencies while matching the provided performance to the workload requirements. The first scheduling technique, *TempoMP*, integrates *Tempo* with an online multi-parametric optimization method. They are used in tandem to deliver locally optimal dynamic thermal management decisions to meet thermal constraints while minimizing power and maximizing performance. Our second scheduling technique, *TemPrompt*, uses *Tempo* as a part of a heuristic algorithm which provides compa-

rable efficiency at lower overhead.

The next section describes the related work, while the rest of the chapter explains details of the *PROMETHEUS* framework and the scheduling techniques.

5.2 Related work

Many temperature aware scheduling techniques have been proposed in recent years for single core processors and MPSoCs. Availability of multiple instances of similar processing resources on MPSoCs creates further opportunities for thermal management compared to single core processors by allowing distribution of activity and heat as necessary. But this also significantly complicates the thermal management process by increasing the size of the solution space. The solution space becomes huge due to the multitude of possibilities regarding assignment of frequencies and tasks to the cores and deciding about when to start the available tasks. The problem is even more complicated by the fact that temperature of any particular point on the die is a strong function of recent temperature and workload history. In general, task scheduling under thermal constraints is an NP-hard problem [87].

Several thermal management techniques for homogeneous MPSoCs have been proposed in previous research. In general purpose and high performance systems, where there is no a priori knowledge of the workload, the techniques are typically based on heuristics. Heat-and-Run [27] performs temperature-aware thread migration for multicore multi-threaded systems In [22], several combinations of thread migration, DVFS and clock gating are studied in thermal management of a homogeneous MPSoC. In [18], a probabilistic approach is taken where the scheduler changes the probability of assigning a task to a core based on the core's temperature history.

Some recent work leverages control theory and optimization to manage thermal issues in homogeneous MPSoCs. In [56], convex optimization is used to control the frequency of the cores on a homogeneous MPSoC to guaranty that thermal constraints are met. In [85], a linear quadratic regulator is used to solve

the frequency assignment problem for thermal balancing. To achieve a smooth control and to minimize performance loss and thermal fluctuations in an MPSoC, [84] proposes a technique based on model predictive control.

Unlike general purpose processing, the type and characteristics of the workloads such as execution time are often known in advance in embedded domain. Although this extra information is helpful in devising better thermal management solutions, the complexity of the problem makes it practically infeasible to find the optimal solution. To manage this complexity and make the problem tractable, various techniques in this domain have used different simplifying assumptions to develop heuristics. In contrast to general purpose domain, in embedded domain the goal is not merely maximizing the throughput and each task might have individual performance requirements such as deadlines

In [21], the problem of scheduling a task graph on a homogeneous MPSoC to minimize the hotspots and balance the temperature distribution is formulated as Integer Linear Programming (ILP). The technique is based on heuristics such as minimizing the overlap between the tasks running on neighbor cores. In [14], an assignment and scheduling technique for hard real time applications on MPSoCs is proposed which uses a mixed-integer linear programming formulation to minimize peak temperature under hard real-time constraints and task dependencies. This technique is limited to tasks with large execution times and works based on a steady-state thermal model. It performs global optimization to minimize the temperature of a set of known tasks. The complexity of this approach increases exponentially with the number of tasks and number of cores and as authors have mentioned, this formulation is not practical for large problem instances. A heuristic approach is also presented for large problems which does not consider a thermal model and works based on the mobility of the tasks.

Relatively little work has focused on heterogeneous embedded MPSoCs. The work in [69] proposes an algorithm for energy and temperature aware scheduling of embedded workloads on heterogeneous MPSoCs where the scheduler operates in two modes based on the utilization of MPSoC. The workload is observed and its performance requirements are estimated. Then, based on the utilization of the

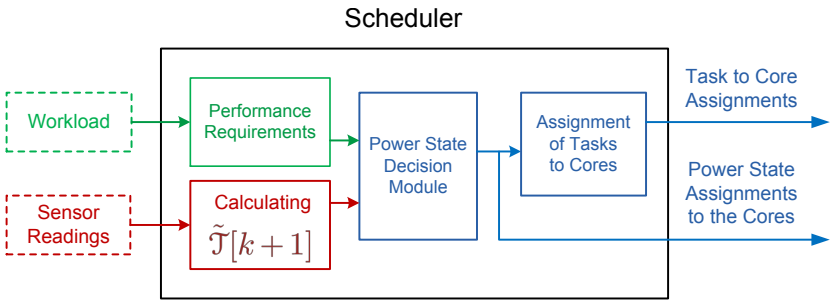


Figure 5.1: Scheduling system in *PROMETHEUS*

processor it is decided if the scheduler should work in energy saving or thermal management mode.

In this section, we propose *PROMETHEUS* framework for proactive management of temperature in heterogeneous MPSoCs by systematically selecting power state of the cores and task assignments. *PROMETHEUS* provides two scheduling techniques, *TempoMP* and *TemPrompt* which both are based on our temperature prediction technique, *Tempo*. In contrast to previous predictors, *Tempo* allows accurate evaluation of future thermal impacts of potential scheduling decisions without having to apply them to the system. Moreover, it is fully linear and does not need any runtime adaptation.

TempoMP incorporates *Tempo* temperature prediction with multi-parametric optimization to choose the optimal alternative among possible power states of the cores and task assignments which does not violate thermal requirements. The other technique, *TemPrompt* uses *Tempo* within a heuristic algorithm which provides comparable efficiency to *TempoMP*, but at a lower overhead.

5.3 PROMETHEUS Scheduling Framework

In this section, we describe *PROMETHEUS*, our proposed framework for proactive temperature aware scheduling. Our approach imposes no restrictions on the distributions of the size and the number of concurrent tasks in the workload which makes it applicable to various classes of workloads. It also systematically considers individual performance, power and thermal characteristics of the cores

so it can be applied to heterogeneous MPSoCs as well as homogeneous ones. Using *Tempo*, the thermal impact of alternative scheduling decisions are evaluated in advance and scheduling decisions which might result in thermal emergencies are avoided. Therefore, *PROMETHEUS* guaranties that the maximum temperature threshold will be met. Here, we present an overview of *PROMETHEUS*'s scheduling system followed by the details of two individual proactive scheduling techniques.

Figure 5.1 shows an overall view of our framework and its operation. Embedded workloads running on the system are pre-characterized in terms of execution time and power. Using pre-characterization information, the *performance requirement estimation* module can estimate the execution time of the tasks at different power states and for each core type to determine a set of power states on which tasks will be able to meet the performance requirements. At each scheduling tick, *temperature predictor module* calculates *Tempo*'s thermal state, $\mathcal{T}[k+1]$, which is the estimated temperature at next scheduling tick if the current power states of the cores do not change. Given the outputs of the *temperature prediction* and *performance requirement estimation* modules, *power state decision module* determines a set of thermally safe power states which are able to provide performance as close as possible to the workload's requirements. The output of this module is used to set the core power states and also in the *task assignment decision module*. Once the power states are determined by the *power state decision module*, *task assignment decision module* decides how the tasks should be assigned to the cores based on their performance requirements.

The two scheduling techniques provided in this framework, *TempoMP* and *TemPrompt* differ only in their *power state decision* modules. The first technique, *TempoMP*, determines the safe power states based on an optimization stage which is performed offline and its results are stored for runtime use. Although the optimization technique can provide locally optimal power state decisions, storing and fetching the optimization results incurs overhead. To avoid this overhead, we propose another scheduling technique called *TemPrompt* which uses a heuristic for its power state decision module. The next sub-sections describes the details of

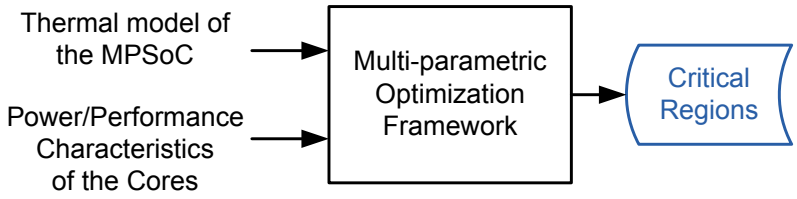


Figure 5.2: Offline stage of *TempoMP*

TempoMP and *TemPrompt* scheduling techniques.

5.3.1 Power state assignment in *TempoMP*

At each scheduling tick, temperature information received from the sensors and scheduling state provided by the scheduler are used in order to determine the new core power states and task assignments. In *TempoMP*, power state module determines the best power state for the cores by referring to the results of offline optimization stage.

Optimization is formulated based on the power and performance characteristics of the cores and the thermal model of the MPSoC as shown in Figure 5.2. The selection of locally optimal power states and task assignment are done based on a multi-parametric optimization framework which incorporates our temperature prediction methods.

Multi-parametric programming is a class of optimization problems in which given an objective function, a set of constraints, and a set of parameters, the optimal solution is obtained as an explicit function of the parameters. It is a very powerful approach for analyzing the effect of variations and uncertainty in optimization problems where objective function is to be minimized or maximized subject to a set of constraints, and a set of parameters which may vary within given bounds. In such optimization problems, multi-parametric programming obtains the objective function and optimization parameters as functions of the varying parameters. It also provides the regions in the solution space where these functions are valid. These regions are called critical regions. Multi-parametric optimization approach splits the optimization process into offline and online stages. Using multi-

parametric programming, the optimization problem is solved offline and the set of critical regions and parametric solutions are provided as output. No optimization needs to be done at runtime. Only a limited number of operations need to be performed at runtime in order to find the critical regions corresponding to the current solution.

We use this technique as a basis for our online optimization. Our goal is to choose what power states the cores should be set to in order to meet the performance requirements of the workload and thermal constraints. Therefore, decision variables are power states of the cores while performance requirements and the thermal state of the MPSoC are the varying parameters in this optimization problem. Without this optimization, various combinations of power states would have to be tried to find the thermally safe power settings which meet the performance requirements of the workload.

Figure 5.3 shows a simple illustrative example of using multi-parametric programming in power state assignment for a single core case. This core has one sleep state and three discrete voltage/frequency settings: f_{high}, f_{medium} and f_{low}. The goal is to set the core to the optimal frequency which is equal to or greater than the minimum frequency required to meet performance requirements of the workload (f_{min}), but does not cause the temperature to exceed the threshold. f_{min} is one of the varying parameters in the optimization and its value is estimated at runtime given the workload. Another parameter obtained at runtime in this decision is *Tempo*'s thermal state of the core, $\mathcal{T}[k+1]$, which is the estimated temperature of the core at the next scheduling tick assuming the power state does not change. $\mathcal{T}[k+1]$ reflects the current trend of the temperature of the cores.

This set of parameters define the solution space as shown in Figure 5.3. The Y axis represents f_{min}, while the X axis represents $\mathcal{T}[k+1]$. The space of the solutions consists of all possible combinations for $\mathcal{T}[k+1]$ and f_{min} which is divided into four regions. Based on the given input parameters, the corresponding region is identified and the corresponding solution is chosen. Each region corresponds to one frequency setting which minimizes the power while providing at least the minimum required frequency. In the example of Figure 5.3, each region is labeled

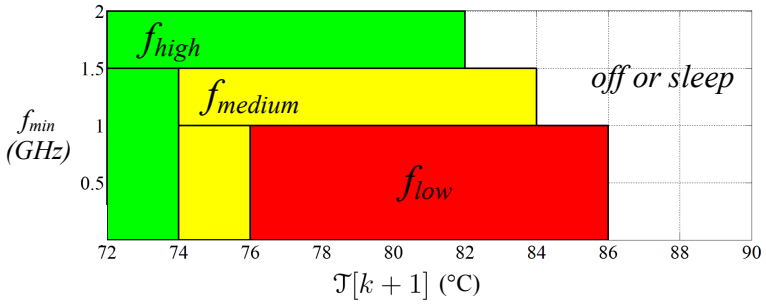


Figure 5.3: A very simple example describing use of multi-parametric programming in power state assignment

by the frequency corresponding to that region.

At very low $\mathcal{T}[k+1]$, even the highest frequency is thermally safe. Therefore, even if the frequency requirement (f_{min}) is high, it can be satisfied. This corresponds to the green area of Figure 5.3. When $\mathcal{T}[k+1]$ is very high, none of the active states are thermally safe, therefore the core should be put to sleep (e.g. when $\mathcal{T}[k+1] > 86°\text{C}$). This corresponds to the white region on the right side of the figure. At $\mathcal{T}[k+1]$ values between these two extremes, the core can be set to one of its active power states. As an example, parameters $\mathcal{T}[k+1] = 80°\text{C}$ and minimum frequency requirement of 1.1 GHz fall into the yellow region which corresponds to the medium voltage/frequency setting of the core. Using multi-parametric programming, we generate such functions which allow us to find optimum power states for each combination of parameters.

We formulate an optimization problem using *Tempo* prediction model with the goal of finding the locally optimal power states of the cores. Thermal states of the cores and performance requirements of the workload are given to the optimization framework as inputs, while the output of the optimization process is a set of power states for the cores that are thermally safe and are able to provide the performance requirements of the workload at the minimum power cost. The optimization formulation uses *Tempo* to evaluate thermal safety of the potential power states. The decision variables in this optimization are power states in the next scheduling tick which are represented by the vector $\alpha[k+1]$ (in bold in

equation (5.1)). The optimization problem is formulated as:

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & P_{total}(\boldsymbol{\alpha[k+1]}, \mathcal{T}[k+1]) \\ \text{subject to} \quad & \mathbb{T}(\boldsymbol{\alpha[k+1]}, \alpha[k], \mathcal{T}[k+1]) \prec T_{Th} \\ & \forall \Omega, v : \lambda_{\Omega,v} \geq \sigma_{\Omega,v}. \end{aligned} \tag{5.1}$$

where we use $\prec$ as element-wise *less than* operator and $\mathcal{T}[k+1]$ is the thermal state defined by *Tempo*. The objective of this optimization is to minimize the total power (P_{total}) of the cores under the thermal limits. P_{total} is the sum of dynamic and leakage power of the cores. Leakage component of P_{total} is calculated using equations (4.13) and (4.14). Details of the optimization objective and constraint formulation are provided below.

The optimization parameters are performance requirements and the temperature of the cores. If core n is set to power state v, then $\alpha_{n,v}$ is 1, and 0 otherwise:

$$\alpha_{n,v} = \begin{cases} 1 & \textit{if core } n \textit{ is set to power state } v \\ 0 & \textit{otherwise} \end{cases} \tag{5.2}$$

Optimization chooses $\alpha[k+1]$ values such that the total power is minimized under the thermal constraints while meeting performance requirements. We denote the number of cores of type Ω which are set to power state v as $\lambda_{\Omega,v}$, where

$$\lambda_{\Omega,v} = \sum_{\omega \in \Omega} \alpha_{\omega,v} \tag{5.3}$$

The minimum number of cores of type Ω that have to run at a given power state v to meet performance requirements of the workload is denoted as $\sigma_{\Omega,v}$. This is determined at runtime based on the performance requirements of the tasks using information such as deadlines or required throughput. For each core, if we assume core of type k has v active power states and one sleep state, the power consumption of core n which is of core type k can be written as

$$P[n] = \alpha_{n,1} \cdot P_{k,1} + ... + \alpha_{n,v} \cdot P_{k,v} + \alpha_{n,sleep} \cdot P_{k,sleep} \tag{5.4}$$

where $P_{k,v}$ is the power consumed at power state v at a core of type k.

The first constraint enforces the predicted temperature at the next scheduling tick to be lower than the maximum threshold, while the second set of constraints require the power states chosen by optimization to provide at least the performance required by the workload. Instead of solving the optimization problem at each scheduling tick, we use an approach based on multi-parametric programming [57]. Optimization parameters σ, $\alpha[k]$ and $\mathcal{T}[k+1]$ partition the parameter space into separate regions called *critical regions*. Each possible combination of σ, $\alpha[k]$ and $\mathcal{T}[k+1]$ corresponds to one and only one of these critical regions which represents the optimum power states of the cores for that specific combination. The region basically specifies the validity range of that set of power states such that temperatures of all the cores are below the threshold temperature and the total power is minimized. The actual values for the optimization parameters are found at runtime. Given the parameter values, the corresponding region is found which represents the appropriate set of power states for the cores. The set of optimal solutions ($\alpha[k+1]$) is obtained as an explicit function of the parameters (σ, $\alpha[k]$ and $\mathcal{T}[k+1]$). To get the optimum results at runtime, the *power state decision module* does not need to do any optimization online. It only needs a limited number of operations to be performed at runtime to find the regions representing the $\alpha[k+1]$ values corresponding to the current value of σ, $\alpha[k]$ and $\mathcal{T}[k+1]$.

Temperature of the cores at the next scheduling tick depends on decision variable $\alpha[k+1]$ and optimization variables $\alpha[k]$, $\mathcal{T}[k+1]$. Larger number of optimization parameters results in much larger solution space. In order to reduce the number of optimization variables, equation (4.10) is transformed to:

$$\mathbb{T}[k+1] = \underbrace{\Phi_{oo}P[k+1]}_{F_p} + \underbrace{(\mathcal{T}[k+1] - \Phi_{oo}P[k])}_{F_r} \tag{5.5}$$

It can be seen that at each scheduling tick, the only unknown component is F_p and the remaining component, F_r, is known. Therefore, at each scheduling tick, F_r can be calculated based on the observable temperature and current power states, and is used as the optimization parameter instead of $\alpha[k]$ and $\mathcal{T}[k+1]$. This significantly reduces the size of the parameter space.

F_r along with performance requirements $\sigma_{\Omega,v}$ are used by the power state

decision module to find the corresponding critical region which represents the set of optimal power states of the cores for the given parameters. After the power states are set, the last step of *TempoMP* is assigning tasks to the cores. The next subsection discusses how task assignment is done.

Offline optimization phase is performed in Matlab [51]. The multi-parametric programming framework is implemented in YALMIP [8] toolbox in Matlab which relies on Multi-Parametric Toolbox (MPT) [6]. Optimization results are saved in the scheduling decision module to be accessed during the online phase of our algorithm. The critical regions are stored in the form of the coefficients of the linear inequalities describing them. Memory required for the critical regions of the MPSoC of Figure 4.3 is less than 500KB.

5.3.2 Power state assignment in $TemPrompt$

Although *TempoMP* is able to choose locally optimal power states, the results of the offline optimization must be stored and retrieved at runtime. This overhead can be high for many core systems. In order to address this issue, we propose an alternative scheduling technique, *TemPrompt*, whose power state assignment is based on a heuristic. Algorithm 1 outlines the power state assignment algorithm of *TemPrompt*.

The algorithm assigns power state to the core types in the decreasing order of performance (line 1). At each stage, $\mathcal{P}[k+1]$ contains the power of potential power states of the cores which at the beginning is initialized to the power at the previous scheduling tick ($P[k]$). At each stage of the algorithm, $\mathcal{F}$ keeps the performance requirements which are not still met. If a core type has not been able to meet all of its performance requirements, the remaining elements are kept in $\mathcal{F}$ so that they are addressed by the next core type (line 8).

Tempo thermal state of the core, $\mathcal{T}_i[k+1]$, is the potential temperature of core i at the end of the next scheduling tick if the power states of the cores do not change. The effect of changes in the power states of the cores are reflected by component $\Phi_{oo}(P[k+1]-P[k])$ as shown in equation (4.12). Cores with lower *Tempo* thermal state are expected to have lower temperature in the future scheduling tick

Algorithm 1 Power state assignment in *TemPrompt*

$\mathcal{I} \Leftarrow$ types of cores in decreasing performance order
$P[k]$, $\mathcal{P}[k+1] \Leftarrow$ Current power of the cores
$\mathcal{F} \Leftarrow \emptyset$
while (Not all cores are assigned frequencies) **do**
  $\Omega \Leftarrow$ the next type in $\mathcal{I}$
  $\mathcal{C}_\Omega \Leftarrow$ set of cores of type Ω in increasing order of $\mathcal{T}_i[k+1]$
  $\sigma_\Omega \Leftarrow$ performance requirements of core type Ω
  $\mathcal{F} \Leftarrow \mathcal{F} \cup \sigma_\Omega$
  while ($\mathcal{C}_\Omega$ and $\mathcal{F}$ are not empty) **do**
    $i \Leftarrow$ the next core in $\mathcal{C}_\Omega$
    $f_{req} \Leftarrow$ the highest power state required from $\mathcal{F}$
    $v_{safe} \Leftarrow$ the highest power state for which $\mathbb{T}[k+1] \prec T_{Th}$
      $(\mathbb{T}[k+1] = \mathcal{T}[k+1] + \Phi_I(\mathcal{P}[k+1] - P[k]))$
    assign $min(f_{req}, v_{safe})$ to i
    update $\mathcal{P}[k+1]$ with the potential power of core i
    remove f from $\mathcal{F}$ and i from $\mathcal{C}_\Omega$
  end while
end while

and can be assigned to higher power states compared to the other cores. Therefore, the algorithm checks the instances of each type of cores in the increasing order of $\mathcal{T}_i[k+1]$.

The algorithm determines f_{req}, the highest power state that the current workload requires from the core type being checked (line 11). Then it finds v_{safe}, the highest thermally safe power state of the core given the current thermal state and power settings (line 12). Core i is set to the minimum of f_{req} and v_{safe} to provide just enough performance for the workload and not to exceed the temperature threshold. Then the power values are updated and then the next core is checked. If there is no core of this type left, then a new core type is examined (line 5). The algorithm finishes if the performance requirements are met or every core is assigned a power state.

5.3.3 Runtime task assignment to the cores

Performance estimation module can determine power state at which each task needs to be run to be able to meet its performance requirements (deadline, throughput, etc.). Given this information from the performance estimation module and the output of power state decision module, the task assignment module tries to assign the tasks to the cores such that the tasks with higher performance requirements are assigned to the cores providing higher performance. Algorithm 2 explains how this is done at each scheduling tick in a deadline based system. This module works identically for both *TempoMP* and *TemPrompt*.

The algorithm checks the available tasks in the system starting with ones in the decreasing order of performance requirement (line 2). In a deadline based system, the task with the highest performance requirement is the one with the earliest deadline. At each step, the task j with the highest performance requirement is chosen (line 7). Then, the core i which has the highest performance (at the highest power state) is chosen (line 8). Then task j is assigned to core i (line 9) and this is repeated until no core or no task is left. The algorithm always finishes because at each iteration it assigns a task to an available core and finishes when no core or no task is left. The algorithm always finishes, because at every iteration,

Algorithm 2 Task to core assignment

```
1: C ⇐ currently available cores
2: J ⇐ current tasks in the system in the decreasing order of performance requirements

3: while (C and J are not empty) do
4:    Ω ⇐ the highest performance type of cores in C
5:    C_Ω ⇐ set of cores of type Ω in C
6:    while C_Ω is not empty do
7:       j ⇐ the first task in J
8:       i ⇐ the core at the highest power (and performance) state in C_Ω
9:       assign task j to core i
10:      remove j from J and i from C_Ω
11:   end while
12:   remove C_Ω from C
13: end while
```

it assigns a power state to one unassigned core and finishes when no unassigned core is left (the maximum number of iterations is the larger of the number of the cores and the tasks).

5.4 Experimental results

The experimental methodology used in this chapter is the same as that of the previous chapter. For details of the experimental methodology, please refer to section 4.4.

We compare the scheduling techniques of *PROMETHEUS*, *TempoMP* and *TemPrompt*, with three other state of the art temperature-aware scheduling techniques. The maximum safe temperature is assumed to be 90°C which must be respected by all techniques. ***PASTEMP*** and ***Thermal_PO*** [71] are both proactive techniques which use optimization to assign thermally safe power states to the cores based on the performance requirements of the workload and also the thermal state of the system. The optimization formulation in *PASTEMP* is based

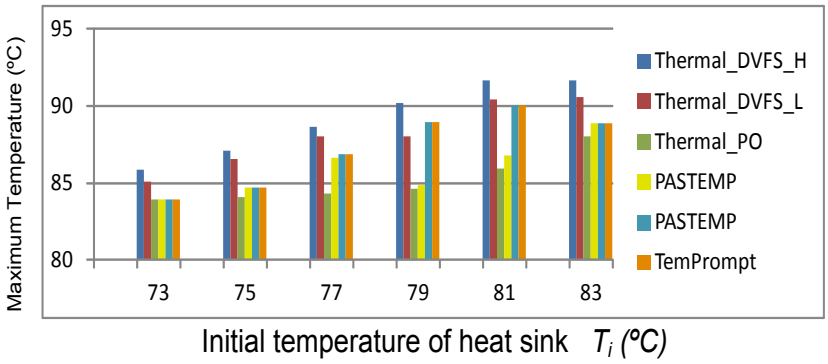


Figure 5.4: Comparison of Maximum Temperature

on a modified dynamic thermal model called *instantaneous thermal model* [71], while optimization in *Thermal_PO* is based on a steady state thermal model.

The third technique, ***Thermal_DVFS*** relies on the direct temperature readings from the thermal sensors. It switches the core to a lower power state when the core temperature reaches a critical threshold of T_{top}. When the temperature gets below a lower threshold, T_{bottom}, the power state of the core can be switched to a higher level again. This lower threshold helps prevent oscillations between power states. They are proactive and change the power state only if their estimated temperature falls in the safe range. We test two variations of *Thermal_DVFS* with different thresholds: *Thermal_DVFSL* with T_{top}=85°C and T_{bottom}=83°C and *Thermal_DVFSH* with T_{top}=87°C and T_{bottom}=85°C. *Thermal_DVFS* does default load balancing to create a balanced distribution of the workload across the cores.

We run a same mix of MiBench benchmarks for 100 seconds. Before each run, the heat sink is pre-heated to the initial temperature T_i ranging from 73 °C to 83 °C. Figure 5.4 reports the maximum core temperature observed under various conditions. At low T_i, all techniques are able to meet the 90°C temperature threshold. Only *Thermal_DVFS* violates the threshold at high T_is, because in this region the core temperature quickly reaches above the threshold before *Thermal_DVFS* can respond.

PASTEMP and *Thermal_PO* have consistently lower maximum temperature, because their thermal models overestimate temperature and as a result of these pessimistic temperature estimates, they tend to make more conservative

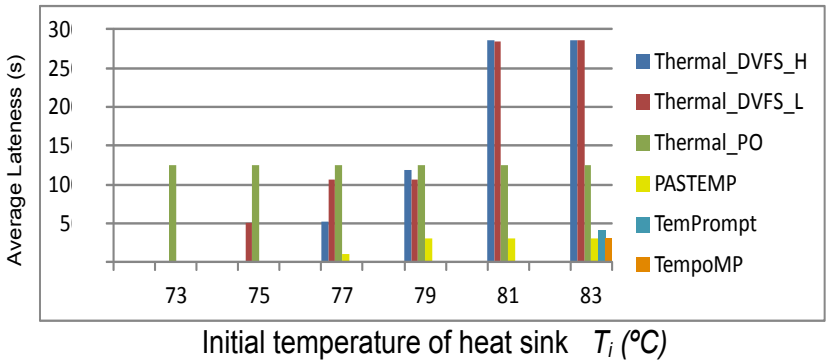


Figure 5.5: Average lateness (seconds)

scheduling decisions and use lower power states. Although this keeps the temperature lower, it results in more performance loss compared to the other techniques as shown next.

To compare how techniques address individual performance requirements of the tasks, we measure *lateness* of a task which is defined as the time it takes to finish the task after its deadline is missed. Since the workload contains tasks of various types and lengths, this metric is more relevant compared to the number of deadline misses. To compare how the potential processing capabilities of the MPSoC are utilized as a whole, we also look at the throughput of the MPSoC which is the number of instructions executed per second. Figure 5.5 and 5.6 report these two metrics. As expected, as T_i increases, so does the average lateness, while the throughput degrades.

TempoMP and *TemPrompt* miss deadlines only at the highest T_i where turning on the SPARC without violating T_{th} is not possible. In this case, no temperature aware scheduling technique can meet all the deadlines because while the performance of SPARC core is necessary to meet the deadlines, turning this core on will violate T_{th}. Another interesting point is that at the highest T_i, *TempoMP* and *PASTEMP* perform similarly and slightly better than *TemPrompt*. The reason is that in this case the larger core cannot be turned on and even *TempoMP* cannot use this core. Therefore its performance is similar to *PASTEMP*. They do slightly better than *TemPrompt* because they both use optimized power state assignments, while *TemPrompt* uses a heuristic approach to make power state decisions. Due

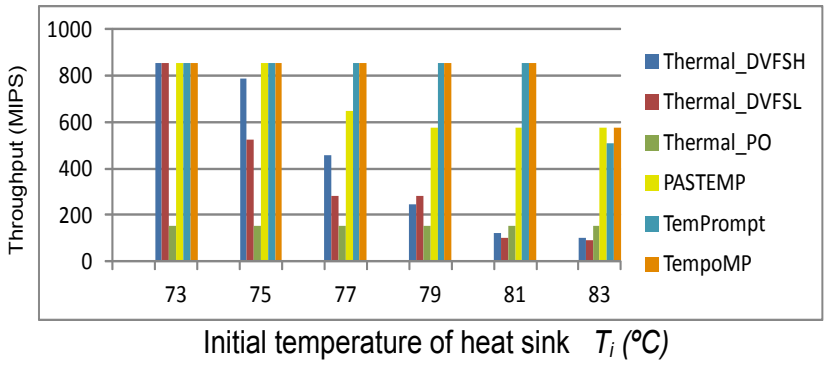


Figure 5.6: Throughput (million instructions executed per second)

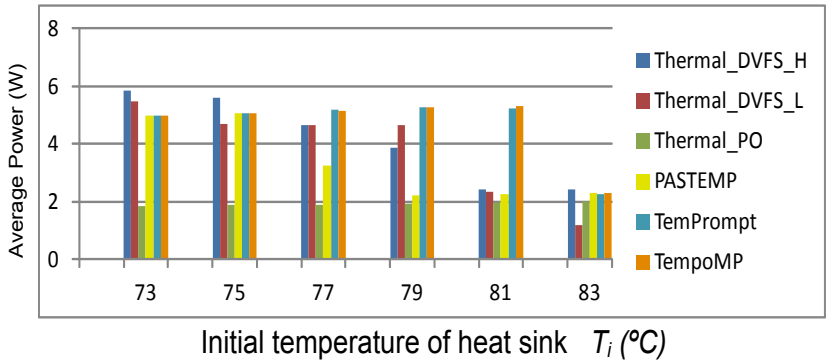


Figure 5.7: Average power consumption

to the pessimistic temperature estimates by the steady state thermal model of *Thermal_PO*, it prevents cores from operating at the highest possible power states. As a result, it consistently performs worse than the other techniques in terms of both lateness and throughput. *TempoMP* and *PASTEMP* improve the lateness by 2.5X and throughput by 1.9X compared to the average of the other techniques. To achieve this performance improvement, *TempoMP* and *TemPrompt* consume around 1.4X more power compared to the average of other techniques as shown in Figure 5.9.

Figure 5.9 reports the average power consumption of MPSoC with various DTM techniques studied. In some cases *TempoMP* and *TempoMP* consume more power than the other techniques. Due to the accurate estimation of thermal slack, these two techniques are able to turn on the higher power large cores to meet the

performance requirements. Other techniques do not use these cores due to their pessimistic temperature estimates and as a result have much lower performance as shown earlier. For example, in all cases, *Thermal_PO* consumes the least power and has the lowest performance.

We also compare these techniques using two different measures of combined energy efficiency and performance: energy consumed per billion instructions executed and energy-lateness product (ELP). ELP is similar to energy-delay product (EDP) metric, but applied to the systems with deadlines. In terms of both of these measures, *TempoMP* and *TempoMP* are on average an order of magnitude better compared to the other techniques. This is mainly because these two techniques provide just enough performance for the workload requirements and use the larger cores -which are less power and energy efficient - only when their high performance is necessary. Therefore, they can better match the performance provided to the performance required. *Thermal_PO* is the least energy efficient because while it uses lower power states of the lower power cores, due to longer execution time and leakage power, it is not energy efficient.

As the experimental results show, *TempoMP* performs the best across all the techniques. It provides the best trade-off between temperature, power and performance. The *TemPrompt* is generally close to *TempoMP*. The main reason that these techniques performed better compared to the other techniques is the accurate evaluation of the thermal impacts of future scheduling decisions. Overestimated temperature leads to conservative decisions thus not taking full advantage of thermal slack which results in performance loss. Running the cores slower might also result in higher energy consumption due to leakage. On the other hand, underestimating future temperature can cause violating temperature requirements and reliability issues. Taking advantage of accurate predictions of *Tempo*, *PROMETHEUS* performs better across various metrics compared to other reactive and proactive techniques. On average, *PROMETHEUS* scheduling techniques reduced the lateness of the tasks by 2.5X and the energy-lateness product by $\sim$5X.

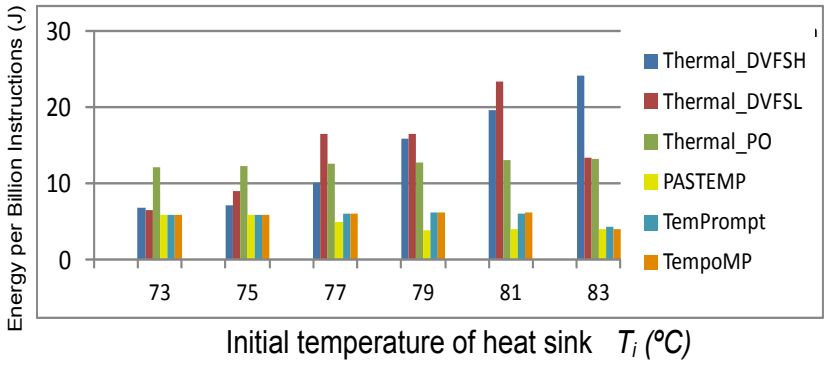


Figure 5.8: Average energy per billion instructions executed

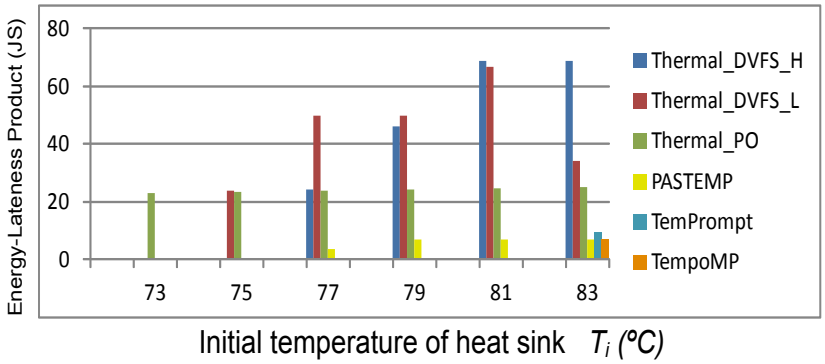


Figure 5.9: Average Energy Lateness Product (ELP)

5.5 Conclusion

In this chapter we introduced *PROMETHEUS*, a framework for proactive temperature aware scheduling of embedded workloads on heterogeneous Multi-Processor Systems-on-Chip (MPSoC). It systematically combines various thermal management mechanisms such as temperature aware task assignment, task migration and DVFS. *PROMETHEUS* is based on our novel low overhead temperature prediction technique, *Tempo*, which was introduced in chapter 4. Using *Tempo*, *PROMETHEUS* framework provides two temperature aware scheduling techniques which proactively avoid power states leading to future thermal emergencies while matching the provided performance to the workload requirements.

The first technique, *TempoMP* incorporates *Tempo* temperature prediction

with multi-parametric optimization to choose the locally optimal alternative among possible power states of the cores and task assignments which does not violate the thermal requirements. As a lower overhead solution, our second scheduling technique, *TemPrompt* uses *Tempo* in a heuristic algorithm which provides comparable efficiency at lower overhead. Taking advantage of accurate predictions of *Tempo*, scheduling techniques of *PROMETHEUS* performs better across various metrics compared to other reactive and proactive techniques. On average, *PROMETHEUS* scheduling techniques reduced the lateness of the tasks by ∼2.5X and the energy-lateness product by ∼5X.

Chapter 5 in part, is a reprint of the material accepted for publication at Design, Automation and Test in Europe (*DATE*) 2012. Sharifi, S. Ayoub, R. and Rosing, T.S. and and the material under submission at IEEE Transactions in Computer Aided Design of Integrated Circuits and Systems. Sharifi, S. Krishnaswamy, D. and Rosing, T.S. The dissertation author was the primary investigator and author of these papers.

Chapter 6

Conclusion and Future Work

MPSoCs are increasingly used in many applications including communication, signal processing, multimedia, etc. One of the major reasons for prevalence of MPSoCs is that they provide higher performance within a specific power budget and thermal envelope. But as technology scales, decreasing device dimensions and increasing power densities result in higher temperatures even in MPSoCs. Elevated temperatures negatively affect reliability, timing characteristics, power, cost and lifetime of these systems. These issues have made temperature one of the major factors which must be addressed in design, manufacturing and test of MPSoCs and other modern computing systems. High cost of off-chip thermal management and high cost of design time techniques in addressing run time workload variations have made DTM essential in these systems. Using DTM, the packaging and cooling can be designed for typical cases which can reduce the costs significantly. When the system temperature approaches critical levels, the chip temperature is controlled by DTM.

This dissertation proposes several techniques to improve efficiency of DTM in MPSoCs. The general approach is applying analytical and formal methods toward thermal management as opposed to the ad-hoc and heuristic methods. This approach allows systematic analyses and development of algorithms which are able to guarantee meeting desired requirements.

This dissertation first introduces an analytical upper bound on the maximum spatial thermal gradients which might happen at run time. Then novel tech-

niques are introduced for accurate temperature sensing which is the starting step for efficient DTM. The first one, called *accurate direct temperature sensing* is a design time technique for optimal sensor allocation and placement. The second one, *accurate indirect temperature sensing* targets inaccuracies which cannot be completely addressed at design time, such as sensor degradation, dynamic change of hotspots, noisy sensors, etc. For efficient proactive DTM, this dissertation proposes a new temperature prediction technique called *Tempo*. Finally, *PROMETHEUS* is introduced, which is an efficient proactive temperature-aware scheduling framework for single ISA heterogeneous MPSoCs.

The rest of this chapter provides a summary of the contributions of this dissertation, followed by some potential directions for future research.

6.1 Thesis Summary

6.1.1 Analytical Model for Upper Bound on On-chip Spatial Thermal Gradients

Spatial and temporal temperature gradients determine the device reliability at moderate temperatures [44]. Spatial temperature differences can create dynamic performance mismatch due to the temperature effect on carrier mobility and interconnect resistance. This phenomenon also affects clock tree design and optimization by introducing thermal-induced clock skew. Therefore, analysis of the temperature variations and maximum spatial thermal gradients (maximum temperature difference between two locations on the die) is crucial. The maximum spatial thermal gradients under different workloads can be found by extensive simulations, which is obviously computationally expensive and workload dependent. For systems interacting with other systems, even the same workload may result in completely different behavior due to interactions with other systems. This makes simulations of a set of benchmarks even less reliable in finding the actual maximum temperature difference.

The first part of this dissertation tries to bring determinism to this area by

proposing an analytical model which guaranties finding the upper bound on spatial thermal gradient between any two locations on the die. A close upper bound is calculated which does not depend on the workloads or interactions between systems. Moreover, the model provides this upper bound with practically no overhead. Our results show that the traditional simulation-based techniques can be up to 9°C off while our technique guaranties finding accurate maximum spatial thermal gradient. In addition, it is able to identify the conditions under which the maximum temperature difference happens. This information can be used in generating test data for thermal stress tests and augmenting the existing benchmarks.

6.1.2 Accurate Direct Temperature Sensing

The first step for efficient dynamic thermal management is obtaining accurate temperature values of various points on the die. If the temperature estimations are lower or higher than the actual temperature, the DTM technique may be activated late or early which are both undesirable. If DTM is activated late, it might fail in keeping the temperature under the desired threshold. On the other hand, if DTM is activated early, it might result in unnecessary performance loss.

One of the major sources of errors in temperature sensing is sensor placement error because typically sensors cannot be placed directly on the locations of interest. Previous sensor placement methods were typically heuristic based and workload dependent, therefore unable to guaranty the accuracies desired by the designer. To address these issues, this dissertation proposes an efficient design time thermal sensor allocation and placement technique which we refer to as *accurate direct temperature sensing*. Relying on our analytical model for thermal gradient analysis, this technique is able to guaranty that the sensor placement error is bounded and less than the desired accuracy. It also shows an average of 16% reduction in the number of thermal sensors compared to the previous techniques.

6.1.3 Accurate Indirect Temperature Sensing

Due to limitations in sensor placement such as routing or constraints in silicon real estate, sometimes it is not possible to place enough number of sensors and close to the locations of interest. In such cases, the readings from the sensors are not accurate. Moreover, there are other sources of inaccuracy in temperature sensing which cannot be addressed at design time. Failure or gradual degradation of sensors are examples of these issues. This dissertation proposes a run time *indirect accurate temperature sensing* method to address these issues. This technique is based on Kalman filtering technique and allows accurate temperature sensing using noisy sensors and even at the locations far from the available sensors. The experiments show that this technique can reduce the mean absolute error and variance of these errors by an order of magnitude.

Indirect accurate temperature sensing is complemented by another run time method for early detection of sensor failures and degradations. Using statistical hypothesis testing techniques, we can detect sensor failure and degradations are detected early and before they affect the efficiency of the thermal management technique. Upon detection of any of these problems, the system adapts and calibrates the *indirect accurate temperature sensing* to avoid sensing inaccuracies due to the new failure or degradation.

6.1.4 *Tempo* Temperature Prediction

Proactive techniques for DTM are more efficient because unlike reactive techniques, they try to avoid thermal emergencies rather than waiting for them to happen and then respond. Therefore, by intelligent and planned actions they can keep the response time and performance overhead of DTM low. Typically, proactive DTM techniques rely on temperature predictors. The efficiency of a proactive DTM directly depends on how well it can predict the temperature. Previous temperature prediction methods typically do not consider the underlying physical characteristics of the temperature. They use general signal analysis and prediction methods and depend only on the history of the temperature. They are not able to evaluate the thermal effects of potential scheduling decisions. In addition, some

previous temperature prediction methods need costly run time adaptations too.

This dissertation proposes a novel temperature prediction method called *Tempo* to address the deficiencies in the existing temperature prediction methods. *Tempo* is a completely deterministic techniques. Based on the physical characteristics of the chip and packaging, its parameters are calculated at design time. Therefore, it does not need any kind of run time adaptations. Moreover, unlike some previous methods, it doesn't depend only on the temperature history and can take into account the effect of future power state changes before they are applied to the system. This capability allows it to accurately evaluate future thermal impact of potential scheduling decisions.

6.1.5 *PROMETHEUS* Framework for Temperature-aware Scheduling in Heterogeneous MPSoCs

Although heterogeneous MPSoCs are used increasingly in various applications, there has been few work addressing temperature aware scheduling on them. This dissertation proposes *PROMETHEUS* framework which allows proactive temperature-aware scheduling in heterogeneous MPSoCs. It is able to consider individual performance, power and thermal characteristics of various cores, which makes it applicable to heterogeneous MPSoCs as well as homogeneous ones. *PROMETHEUS* uses *Tempo* to systematically decide about thermally safe power states of the cores and task to core assignments. It provides two different scheduling techniques based on *Tempo*, *TempoMP* and *TemPrompt*. *TempoMP* incorporates *Tempo* with multi-parametric optimization to choose the optimal and thermally safe alternative among possible power states of the cores and task assignments. In the interest of scalability, *TemPrompt* uses a heuristic algorithm which provides efficiency comparable to *TempoMP* at a lower overhead. Both scheduling techniques in *PROMETHEUS* can guarantee meeting thermal requirements by evaluating future impacts of possible scheduling decisions and avoiding decisions leading to thermal emergencies.

6.2 Future Research Directions

In this dissertation, thermal management of heterogeneous MPSoCs consisting of cores with the same instruction set architecture (ISA) was addressed. There are also more strongly heterogeneous MPSoCs where various types of general purpose (GP) and special purpose (SP) cores are integrated on the same chip are not addressed here. These MPSoCs are getting more prevalent and there is a growing need for techniques addressing thermal management challenges specific in these systems.

The other problem not addressed in this dissertation the increasing complexity of the techniques proposed here which superlinearly depends on the number of cores in an MPSoC. Therefore, these techniques cannot be directly applied to many-core systems because of their computational overhead. Increasing demand for performance and functionality has caused a constant increase in the number of cores on a MPSoC which has led to emergence of hundred- and thousand-core systems. Techniques are needed which are able to overcome the complexity of thermal management in many-core systems with reasonable overhead.

Here we discuss the future research directions addressing these challenges and propose potential approaches to these problems.

6.2.1 Thermal Management in Heterogeneous MPSoCs with Special Purpose Cores

Integrating cores with the same ISA which operate at various power and performance points is one form of heterogeneity. The heterogeneity can also be provided by addition of diverse special purpose (SP) cores (or accelerator cores) specifically designed for certain applications. Existing examples of such embedded heterogeneous MPSoCs are Qualcomm's Snapdragon platform [63], or Texas Instruments' OMAP platform [78]. Qualcomm's latest Snapdragon platform includes two general purpose cores also called application cores (ARM Cortex A9), a DSP, a multimedia co-processor and a graphics processing unit. Texas Instrument's OMAP5 platform [7] includes four general purpose cores (two ARM Cortex

A15 and two ARM Cortex M4 cores), a DSP, 3D and 2D graphics accelerators, video accelerator, an image signal processor and an audio processor, plus some other accelerators and coprocessors.

Special purpose cores are specifically designed and optimized for the special tasks they are supposed to do. They may be third party IP (intellectual property). Many of them are hard IP cores which are optimized for certain technologies and described in low-level physical descriptions and could not be modified at the time of integration. Such components typically do not run general purpose operating systems (OS). Also they are not under full control of the central operating system. In some cases, these cores might have autonomous control of their power states for thermal management purposes, but usually these controls are not coordinated with the other cores in the system. These issues create challenges for thermal management of the overall MPSoCs. Increasing use of heterogeneous MPSoCs with special purpose cores calls for techniques addressing these challenges.

One possible approach is controlling the temperature on the SP cores indirectly through the knobs available for controlling general purpose cores such as DVFS and task migration. The OS running on the system (on GP cores) should be aware of the heterogeneity of MPSoC, characteristics of the cores and performance requirements of the tasks in the system. This allows the OS to indirectly control the temperature of the autonomous SP cores. For example, when an autonomous SP core is getting hot, the OS running on the general purpose (GP) cores can migrate tasks running on its neighbor GP to a GP farther away from the SP core.

6.2.2 Thermal Management in Many-core MPSoCs

Demand for high performance is continuously increasing, but typically the power and thermal limits have not increased. This, plus the diminishing return of extracting performance from traditional instruction-level parallelism (ILP) lead to use of multi-core processors to keep up with Moore's law. To continue this trend, going from multi-core to many-core and thousand-core processors is inevitable [12]. Intel's 80-core chip [80], TILERA's TILE-Gx100 [79] with 100 cores and AMD's Radeon family of graphics chips with up to 1600 stream processing units [5] are

signs of this emerging trend.

Traditionally, a centralized controller monitors the temperature of the entire MPSoC and based on the temperature and state of the cores makes decisions regarding assignment of the tasks to the cores and setting power states of the cores. Therefore, the amount of information from thermal sensors and the computational overhead for decisions in many-core systems will be prohibitive. As a matter of fact, as the number of cores on an MPSoC increases, the complexity of this kind of thermal management increases exponentially. Therefore, centralized approaches are not scalable and practical for many-core systems.

New approaches are needed to manage the complexity of dynamic thermal management in these systems. Decentralizing the thermal control can significantly reduce this complexity and make the solution more scalable. Therefore, distributed thermal management will definitely be an interesting direction to follow.

One possible approach would be using a collaborative approach where each core has its own simple controller, but these controllers cooperate and collectively decide about their future power state. To decide which core to set to a lower power state, information is required regarding the tasks running on the cores, their characteristics and performance impact. This information can be exchanged among the neighbor cores through a low overhead message passing mechanism. The simple controller can be implemented in hardware where there is no OS running.

Such a distributed approach is also applicable to cases where power states of some of the cores cannot be controlled by the OS. In this case, when there is no control on the power states of a core, its controller can request the neighbor cores to switch to lower power states in order to cool down that core.

Appendix A

Compact Thermal Modeling

The work presented in this dissertation uses a compact thermal modeling methodology which translates the three dimensional partial differential heat diffusion equation into a compact network of thermal resistances and thermal capacitances. This model allows studying steady state temperature as well as the transient evolution of temperature. Here we describe this compact thermal RC modeling methodology briefly. For more information and detailed discussion, interested readers can refer to [74, 33, 32].

This methodology allows developing a parameterized compact thermal model for multi-layered package structures. Typical components that are modeled include silicon die, heat spreader, thermal interface material, heat sink, etc. These layers can be modeled with different levels of granularity and details, e.g. at the level of functional units or regular grid cells. The methodology can be applied at early stages of design where detailed layout is not available.

A.1 Electrical Representation of Heat Transfer

The differential equations describing the heat flow have a form dual to that of electrical current. This duality is the basis for the micro-architectural thermal model proposed in [74] and is further explained in [33] and [32].

Using the law of heat conduction, also known as Fourier's Law, the temperature at a given point in a homogeneous material can be related to the coordinates

of that point and the time elapsed by the following equation:

$$k\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right)+g=\rho\cdot C_p\cdot\frac{\partial T}{\partial t} \tag{A.1}$$

where k is the thermal conductivity in W/m^oC, T is the temperature in °C, g is the power density of the heat source in W/m^3, ρ is the density of the material in Kg/m^3, and C_p is the specific heat in J/Kg^oC. This partial differential equation describing the heat flow has a form similar to the equation describing electrical current. This is the basis for the well-known duality between thermal and electrical phenomena which is summarized in Table A.1. The heat flow (W) through a thermal resistor (in $^oC/W$) corresponds to the electrical current (A) through an electrical resistance (Ohm). Similarly, the temperature difference ^oC corresponds to voltage difference (Volt). Heat absorbing capability of a material is described by the thermal capacitance (in $J/^oC$), which corresponds to electrical capacitance (Farad) which describes the the ability of a material in accumulating electrical charges . g is the heat flow generated by power consumption of the functional units which corresponds to electrical current.

Based on this duality, the thermal dynamics of the chip and package are modeled by a network of thermal resistances and capacitances. In this electrical representation of heat transfer, the node voltages will represent the temperature of the corresponding nodes in the material. Current sources will model the power consumption of the functional units and an independent voltage source will model the ambient temperature. To model the lateral heat conduction path between the neighboring nodes, a thermal resistance is added. Heat transfer between the connected layers of material is modeled by vertical thermal resistances which connect the corresponding nodes in the thermal RC network. Figure A.1 is an example of this thermal model created for the chip with 4 functional units and its package shown on the left.

Based on this model, the dynamics of the temperature and the relation between the temperature, power consumption of the cores and thermal characteristics of the system is described as:

$$C_{th}\frac{d}{dt}T(t)=-G_{th}T(t)+P(t) \tag{A.2}$$

Table A.1: Duality between Thermal and Electrical Quantities

Thermal Quantities	***Electrical Quantities***
Power	Current
Temperature Difference	Voltage Difference
(R_{th}) Thermal Resistance	(R) Electrical Resistance
(C_{th}) Thermal Capacitance	(C) *Electrical Capacitance*
($R_{th} \times C_{th}$) Thermal RC Time Constant	($R \times C$) Electrical RC Time Constant

where the vectors and matrices are defined as:

T	Temperature at all the nodes of the thermal network
P	Power consumptions of the nodes of thermal network
G_{th}	Thermal conductance matrix
C_{th}	Thermal capacitance matrix

A.2 Extracting the Parameters of the Thermal Network

In this section we describe how the values for thermal resistances and capacitances are computed. Thermal resistance in a material can be calculated as:

$$R = \frac{t}{K.A} \tag{A.3}$$

where t is the thickness of the material, A is the cross-sectional area across which the heat is being transferred and K is the thermal conductivity of the material per unit volume which is $100W/(m.K)$ for silicon and $400W/(m.K)$ for copper at 85°C. Thermal capacitance, which is proportional to both thickness and area of the material can be calculated as:

$$C = c.t.A \tag{A.4}$$

where c is the thermal capacitance per unit volume which is $1.75x10^6 J/(m^3.K)$ for silicon and $3.55x10^6 J/(m^3.K)$ for copper. It should be noted that this model

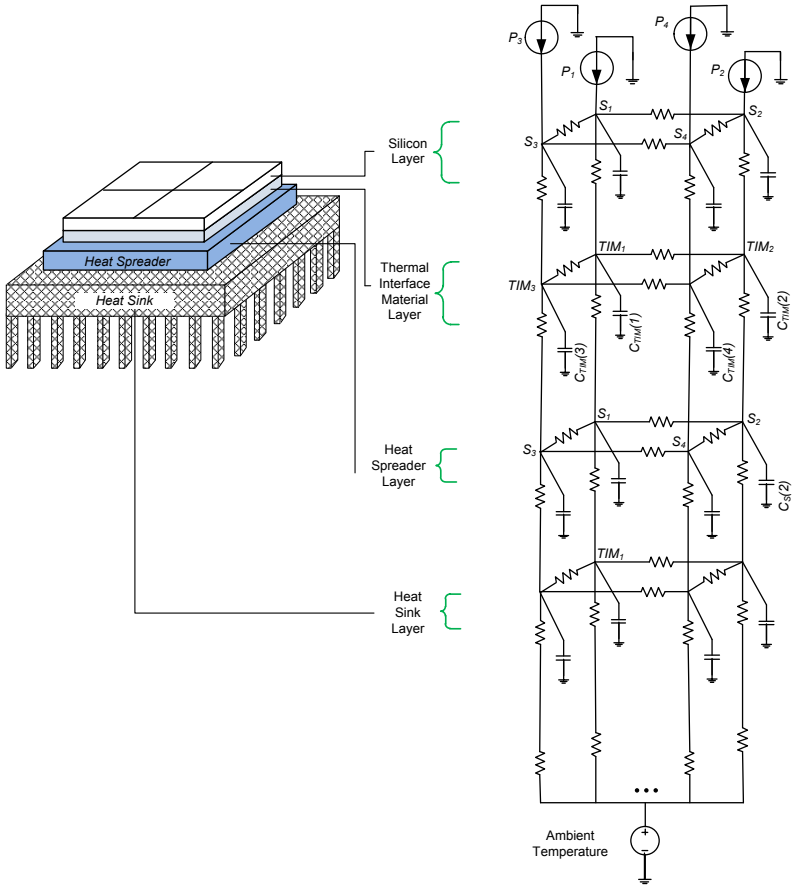


Figure A.1: An example of a chip and package, together with their corresponding thermal RC network

requires a scaling factor to be applied to the capacitors to account for some simplifications in this lumped model relative to a full, distributed RC model. These factors can be analytically derived from physical properties [74]. The thermal resistance of heatsink to air convection can also be modeled as:

$$Rconvection = \frac{1}{h.A} \tag{A.5}$$

where h is the heat transfer coefficient and A is the surface area of convection.

For more detailed discussion on the extraction of the thermal model and parameters, the interested reader can refer to [74, 33, 32].

Bibliography

[1] Intel® PXA270 processor, electrical, mechanical and thermal specification data sheet. http://www.intel.com.

[2] International technology roadmap for semiconductors (ITRS). http://public.itrs.net/.

[3] lp_solve. http://lpsolve.sourceforge.net/.

[4] Hotspot temperature modeling tool, 2010. http://lava.cs.virginia.edu/HotSpot/.

[5] AMD Radeon™ graphics chips., 2011. http://www.amd.com /us/products/desktop/graphics.

[6] Multi-parametric toolbox (mpt), 2011. http://control.ee.ethz.ch/mpt/.

[7] Ti omap5 platform, 2011. http://www.ti.com.

[8] Yalmip, 2011. http://users.isy.liu.se/johanl/yalmip/.

[9] A. Ajami, K. Banerjee, and M. Pedram. Modeling and analysis of nonuniform substrate temperature effects on global ulsi interconnects. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(6):849 – 861, june 2005.

[10] R. Z. Ayoub and T. S. Rosing. Predict and act: dynamic thermal management for multi-core processors. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, ISLPED '09, pages 99–104, 2009.

[11] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The m5 simulator: Modeling networked systems. *IEEE Micro*, 26, July 2006.

[12] S. Borkar. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual Design Automation Conference*, DAC '07, pages 746–749, New York, NY, USA, 2007. ACM.

[13] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, pages 83 –94, june 2000.

[14] T. Chantem, R. Dick, and X. Hu. Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 288 –293, 2008.

[15] M. Cho, S. Ahmedtt, and D. Z. Pan. TACO: temperature aware clock-tree optimization. In *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, ICCAD '05, pages 582–587, Washington, DC, USA, 2005. IEEE Computer Society.

[16] R. Cochran and S. Reda. Spectral techniques for high-resolution thermal characterization with limited sensor data. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 478–483, New York, NY, USA, 2009. ACM.

[17] J. Cong, J. Wei, and Y. Zhang. A thermal-driven floorplanning algorithm for 3d ics. In *Proceedings of the 2004 IEEE/ACM International Conference on Computer-Aided Design*, pages 306–313, 2004.

[18] A. Coskun, T. S. Rosing, and K. Whisnant. Temperature aware task scheduling in mpsocs. In *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, pages 1 –6, 2007.

[19] A. K. Coskun, T. S. Rosing, and K. C. Gross. Utilizing predictors for efficient thermal management in multiprocessor socs. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 28:1503–1516, October 2009.

[20] A. K. Coskun, T. S. Rosing, K. Mihic, G. De Micheli, and Y. Leblebici. Analysis and optimization of MPSoC reliability. *Journal of Low Power Electronics*, 2(1):56–69, 2006.

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

[22] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture*, pages 78–88, 2006.

[23] K. Fu. *Sequential methods in pattern recognition and machine learning*. Mathematics in science and engineering. Academic Press, 1968.

[24] G. Fursin, J. Cavazos, M. O'Boyle, and O. Temam. Midatasets: creating the conditions for a more realistic evaluation of iterative optimization. In *Proceedings of the 2nd international conference on High performance embedded architectures and compilers*, HiPEAC'07, pages 245–260, Berlin, Heidelberg, 2007. Springer-Verlag.

[25] G. Fursin, J. Cavazos, M. O'Boyle, and O. Temam. Midatasets: creating the conditions for a more realistic evaluation of iterative optimization. In *Proceedings of the 2nd international conference on High performance embedded architectures and compilers*, HiPEAC'07, pages 245–260, Berlin, Heidelberg, 2007. Springer-Verlag.

[26] G. Golub and C. Van Loan. *Matrix Computations*. Baltimore, MD, second edition, 1989.

[27] M. Gomaa, M. D. Powell, and T. N. Vijaykumar. Heat-and-run: leveraging smt and cmp to manage power density through the operating system. In *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, ASPLOS-XI, pages 260–270, New York, NY, USA, 2004. ACM.

[28] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop*, pages 3–14, Washington, DC, USA, 2001. IEEE Computer Society.

[29] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop*, pages 3–14, Washington, DC, USA, 2001. IEEE Computer Society.

[30] S. Heo, K. Barr, and K. Asanović. Reducing power density through activity migration. In *Proceedings of the 2003 international symposium on Low power electronics and design*, ISLPED, pages 217–222, 2003.

[31] HP. Cacti, 2011. http://www.hpl.hp.com/research/cacti.

[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. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(5):501 – 513, May 2006.

[33] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam. Compact thermal modeling for temperature-aware design. In

Proceedings of the 41st annual Design Automation Conference, DAC '04, pages 878–883, New York, NY, USA, 2004. ACM.

[34] Intel. Intel pxa270 processor, electrical, mechanical and thermal specification data sheet., 2011. http://www.intel.com.

[35] C. Isci, G. Contreras, and M. Martonosi. Hardware performance counters for detailed runtime power and thermal estimations: Experiences and proposals. In *Hardware Performance Monitor Design and Functionality Workshop in conjunction with 11th International Symposium on High-Performance Computer Architecture (HPCA-11)*, 2005.

[36] C. Isci, G. Contreras, and M. Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, pages 359–370, Washington, DC, USA, 2006. IEEE Computer Society.

[37] JEDEC Solid State Technology Association. JEDEC Standard 51-2A, Integrated Circuit Thermal Test Method Environmental Conditions-Natural Convection (Still Air), 2008. http://www.jedec.org/standards-documents/docs/jesd-51-2a.

[38] P. Kabisatpathy, A. Barua, and S. Sinha. *Fault diagnosis of analog integrated circuits*. Frontiers in electronic testing. Springer, 2005.

[39] K. W. Kenny C. Gross and A. Urmanov. Electronic prognostics through continuous system telemetry. In *Meeting of the Society for Machine Failure Prevention Technology (MFPT)*, pages 53–62, 2006.

[40] O. Khan and S. Kundu. Hardware/software co-design architecture for thermal management of chip multiprocessors. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 952 –957, 2009.

[41] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Processor power reduction via single-isa heterogeneous multi-core architectures. *Computer Architecture Letters*, 2, 2003.

[42] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In *ISCA*, pages 64–75, 2004.

[43] R. Kuppuswamy, S. Sawant, S. Balasubramanian, P. Kaushik, N. Natarajan, and J. Gilbert. Over one million TPCC with a 45nm 6-core Xeon® CPU. In *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 70 –71,71a, feb. 2009.

[44] C. J. M. Lasance. Thermally driven reliability issues in microelectronic systems: status-quo and challenges. *Microelectronics Reliability*, 43(12):1969–1974, 2003.

[45] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems. In *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*, MICRO 30, pages 330–335, Washington, DC, USA, 1997. IEEE Computer Society.

[46] K.-J. Lee, K. Skadron, and W. Huang. Analytical model for sensor placement on microprocessors. In *Proceedings of the 2005 International Conference on Computer Design*, ICCD '05, pages 24–30, Washington, DC, USA, 2005. IEEE Computer Society.

[47] A. S. Leon, K. W. Tam, J. L. Shin, D. Weisner, and F. Schumacher. A power-efficient high-throughput 32-thread sparc processor. *Solid-State Circuits, IEEE Journal of*, 42(1):7 –16, 2007.

[48] P. Liu, H. Li, L. Jin, W. Wu, S. X.-D. Tan, and J. Yang. Fast thermal simulation for runtime temperature tracking and management. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(12):2882–2893, 2006.

[49] Y. Liu, R. Dick, L. Shang, and H. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, pages 1 –6, 2007.

[50] J. Long, S. O. Memik, G. Memik, and R. Mukherjee. Thermal monitoring mechanisms for chip multiprocessors. *ACM Trans. Archit. Code Optim.*, 5:9:1–9:33, September 2008.

[51] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[52] M. Meterelliyoz, H. Mahmoodi, and K. Roy. A leakage control system for thermal stability during burn-in test. In *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*, pages 10 pp. –991, nov. 2005.

[53] F. Mohammadi and M. Marami. Dynamic compact thermal model of a package. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 2869 –2872, may 2008.

[54] S. Mondal, R. Mukherjee, and S. O. Memik. Fine-grain thermal profiling and sensor insertion for fpgas. In *ISCAS*, 2006.

[55] R. Mukherjee and S. O. Memik. Systematic temperature sensor allocation and placement for microprocessors. In *Proceedings of the 43rd annual Design Automation Conference*, DAC '06, pages 542–547, New York, NY, USA, 2006. ACM.

[56] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. De Micheli. Temperature control of high-performance multi-core platforms using convex optimization. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '08, pages 110–115, New York, NY, USA, 2008. ACM.

[57] D. Narciso, N. Faisca, and E. Pistikopoulos. A framework for multi-parametric programming and control; an overview. In *Engineering Management Conference, 2008. IEMC Europe 2008. IEEE International*, pages 1 –5, june 2008.

[58] A. Naveh, E. Rotem, A. Mendelson, S. Gochman, R. Chabukswar, K. Krishnan, and A. Kumar. Power and thermal management in the intel core duo processor. *Intel Technology Journal*, 10(2):109–122, 2006.

[59] A. Odabasioglu, M. Celik, and L. Pileggi. Prima: passive reduced-order interconnect macromodeling algorithm. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 17(8):645 –654, aug 1998.

[60] S. Park, J.-J. Chen, D. Shin, Y. Kim, C.-L. Yang, and N. Chang. Dynamic thermal management for networked embedded systems under harsh ambient temperature variation. In *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, pages 289 –294, 2010.

[61] M. Pedram and S. Nazarian. Thermal modeling, analysis, and management in vlsi circuits: principles and methods. In *Proceedings of the IEEE*, 2006.

[62] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, and et al. The design and implementation of a first-generation CELL processor. *ISSCC 2005 IEEE International Digest of Technical Papers SolidState Circuits Conference 2005*, 10(2):184–186, 2005.

[63] Qualcomm®. Snapdragon ™ Processor Family., 2011. http://www.qualcomm.com/snapdragon.

[64] S. Remarsu and S. Kundu. On process variation tolerant low cost thermal sensor design in 32nm cmos technology. In *Proceedings of the 19th ACM Great Lakes symposium on VLSI*, GLSVLSI '09, pages 487–492, New York, NY, USA, 2009. ACM.

[65] E. Rotem, J. Hermerding, A. Cohen, and H. Cain. Temperature measurement in the Intel® Core™ duo processor. In *Thermal Investigations of ICs and Systems (THERMINIC), International Workshop on*, 2007.

[66] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron. A case for thermal-aware floorplanning at the microarchitectural level. *Journal of Instruction-Level Parallelism*, 7(2), October 2005.

[67] M. Santarini. Thermal integrity: a must for low-power-ic digital design. *EDN*, pages 37–42, September 2005.

[68] T. Sato, J. Ichimiya, N. Ono, K. Hachiya, and M. Hashimoto. On-chip thermal gradient analysis and temperature flattening for soc design. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, ASP-DAC '05, pages 1074–1077, New York, NY, USA, 2005. ACM.

[69] S. Sharifi, A. K. Coskun, and T. S. Rosing. Hybrid dynamic energy and thermal management in heterogeneous embedded multiprocessor SoCs. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, ASPDAC, pages 873–878, 2010.

[70] S. Sharifi and T. S. Rosing. Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(10):1586 –1599, 2010.

[71] S. Sharifi and T. S. Rosing. Package-Aware Scheduling of Embedded Workloads for Temperature and Energy Management on Heterogeneous MPSoCs. In *International conference on computer design*, 2010.

[72] D. Simon. *Optimal state estimation: Kalman, H∞ and nonlinear approaches*. Wiley-Interscience, 2006.

[73] T. Simunic, L. Benini, P. W. Glynn, and G. D. Micheli. Event-driven power management. *IEEE Trans. on CAD of Integrated Circuits and Systems*, pages 840–857, 2001.

[74] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proceedings of the International Symposium on Computer Architecture*, pages 78–88, 2003.

[75] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Archit. Code Optim.*, 1, March 2004.

[76] D. Sylvester and H. Kaul. Future performance challenges in nanometer design. In *Proceedings of the 38th annual Design Automation Conference*, DAC '01, pages 3–8, New York, NY, USA, 2001. ACM.

[77] A. Telikepalli. Designing for power budgets and effective thermal management. *Xcell Journal*, (56), 2006.

[78] Texas Instruments®. OMAP ™ Mobile Processors., 2011. http://focus.ti.com/omap/docs/omaphomepage.tsp .

[79] TILERA®. Tile-Gx ™ Processor Family., 2011. http://www.tilera.com/products/processors.

[80] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-tile sub-100-w teraflops processor in 65-nm cmos. *Solid-State Circuits, IEEE Journal of*, 43(1):29 –41, jan. 2008.

[81] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebonheur. Thermal performance challenges from silicon to systems. *Intel Technology Journal*, vol(3):1–16, 2000.

[82] J. M. Wang and T. V. Nguyen. Extended krylov subspace method for reduced order analysis of linear circuits with multiple sources. In *Proceedings of the 37th Annual Design Automation Conference*, DAC '00, pages 247–252, New York, NY, USA, 2000. ACM.

[83] K. Whisnant, K. Gross, and N. Lingurovska. Proactive fault monitoring in enterprise servers. In L. T. Yang, H. R. Arabnia, Y. Li, S. N. Salloum, and J. G. Delgado-Frias, editors, *CDES*, pages 3–10. CSREA Press, 2005.

[84] F. Zanini, D. Atienza, L. Benini, and G. De Micheli. Multicore thermal management with model predictive control. In *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, pages 711 –714, aug. 2009.

[85] F. Zanini, D. Atienza, and G. De Micheli. A control theory approach for thermal balancing of mpsoc. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, ASP-DAC '09, pages 37–42, Piscataway, NJ, USA, 2009. IEEE Press.

[86] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *Proceedings of the IEEE/ACM international conference on Computer-aided design*, ICCAD, pages 281–288, 2007.

[87] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, ICCAD '07, pages 281–288, 2007.

[88] Y. Zhang and A. Srivastava. Accurate temperature estimation using noisy thermal sensors. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 472–477, New York, NY, USA, 2009. ACM.