# ABSTRACT

Title of dissertation: THERMAL TRACKING AND ESTIMATION FOR MICROPROCESSORS

Yufu Zhang, Doctor of Philosophy, 2016

Dissertation directed by: Professor Ankur Srivastava
Department of Electrical and Computer Engineering

Due to increasing integration density and operating frequency of today's high performance processors, the temperature of a typical chip can easily exceed 100 degrees Celsius. However, the runtime thermal state of a chip is very hard to predict and manage due to the random nature in computing workloads, as well as the process, voltage and ambient temperature variability (together called PVT variability). The uneven nature (both in time and space) of the heat dissipation of the chip could lead to severe reliability issues and error-prone chip behavior (e.g. timing errors). Many dynamic power/thermal management techniques have been proposed to address this issue such as dynamic voltage and frequency scaling (DVFS), clock gating and etc. However, most of such techniques require accurate knowledge of the runtime thermal state of the chip to make efficient and effective control decisions. In this work we address the problem of tracking and managing the temperature of microprocessors which include the following sub-problems: (1) how to design an efficient sensor-based thermal tracking system on a given design that could provide accurate real-time temperature feedback; (2) what statistical

techniques could be used to estimate the full-chip thermal profile based on very limited (and possibly noise-corrupted) sensor observations; (3) how do we adapt to changes in the underlying system's behavior, since such changes could impact the accuracy of our thermal estimation.

The thermal tracking methodology proposed in this work is enabled by on-chip sensors which are already implemented in many modern processors. We first investigate the underlying relationship between heat distribution and power consumption, then we introduce an accurate thermal model for the chip system. Based on this model, we characterize the temperature correlation that exists among different chip modules and explore statistical approaches (such as those based on Kalman filter) that could utilize such correlation to estimate the accurate chip-level thermal profiles in real time. Such estimation is performed based on limited sensor information because sensors are usually resource constrained and noise-corrupted. We also took a further step to extend the standard Kalman filter approach to account for (1) nonlinear effects such as leakage-temperature interdependency and (2) varying statistical characteristics in the underlying system model. The proposed thermal tracking infrastructure and estimation algorithms could consistently generate accurate thermal estimates even when the system is switching among workloads that have very distinct characteristics. Through experiments, our approaches have demonstrated promising results with much higher accuracy compared to existing approaches. Such results can be used to ensure thermal reliability and improve the effectiveness of dynamic thermal management techniques.

# THERMAL TRACKING AND ESTIMATION FOR MICROPROCESSORS

by

Yufu Zhang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Advisory Committee:
Professor Ankur Srivastava, Chair/Advisor
Professor Joseph JaJa
Professor Donald Yeung
Professor Charles Silio
Professor Jiuzhou Song

# Acknowledgments

I owe my gratitude to all the people who have guided me and supported me unconditionally. Without them this thesis will not be possible.

First and foremost I'd like to thank my advisor, Professor Ankur Srivastava, who has given me invaluable opportunity to work on challenging and extremely interesting projects. He has always made himself available when I needed guidance. I could never forget numerous days and nights spent together in the lab brainstorming ideas, discussing solutions and evaluating results. He has been a tremendous mentor. I still remember when he corrected me in an email saying that I do not work "under" him, I work together "with" him. He has also been very patient and understanding when I was going through a tough period of life and kept trusting me and encouraging me. I am very lucky to be able to work with such an extraordinary individual and I cherish this experience forever.

I would also like to thank my dissertation defense committee members, especially Professor Joseph JaJa and Professor Donald Yeung. Professor JaJa has personally taught me several courses which proved to be extremely useful in my research and also later in my career. Professor Yeung has given me valuable advice during my proposal stage, which has been very helpful for my research. To them I own deep gratitude.

I want to thank many other professors in our department or university who have taught me graduate level courses and/or given me valuable advices. I own my gratitude to Professor Gang Qu, Bruce Jacob, Rajeev Barua, Kazuo Nakajima,

Jiuzhou Song, and etc.

I would like to thank my colleagues and classmates, especially Bing Shi and Dominic Forte who are very good friends of mine. We spent a few most active years of our lives together pursuing similar academic goals. We constantly exchanged ideas. We helped each other evaluating experimental results, and we encouraged each other a lot in general. We had so many discussions on many shared topics of interest, both in research and courses. Those discussions have helped me greatly.

I would like to express my gratitude to some of the staff members in our department. Especially Melanie Prange, Tracy Chung, Vivian Lu and etc. Throughout the years, they have never hesitated to provide generous help whenever I seek it from them. They have made my study and research experience in a foreign country so much easier.

I own my deepest thanks to my wife, my parents, and my children. When I was lost they are the beacon so I know where to find home. They have made tremendous sacrifice to support me. To them I am forever grateful.

It is impossible to remember all, and I apologize to those I've inadvertently left out. I appreciate the help that anyone provided me the past a few years. Thank you all!

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Thermal Challenges

## 1.1 Historical Trends

The evolution of Very Large Scale Integrated Circuit (VLSI) is one of the most important technology developments of our times. It has far-reaching influence and has enabled many advancements in consumer electronics, such as high performance desktop computers, laptops, tablets, smart phones, music players, game consoles, and etc. As predicted by Moore's Law [34], the number of transistors integrated on a single silicon chip roughly doubles every two years (see figure 1.1, data from Intel Corp. [1]). This has enabled the performance of integrated circuits and their form factors to improve dramatically. The computing speed that we used to be able to achieve with room-sized machines can now be easily achieved by laptops or even smart phones. This dramatic improvement in computing power is usually attributed to technology scaling (the shrinking size of device dimensions printed on silicon). On one hand we can integrate more and more devices onto a single chip to improve its functionality and capability. On the other hand, the smaller devices can switch faster and the signal path becomes shorter, thus achieving ever higher clock frequency (the speed at which the circuit operates). These factors all contributed to the dramatically improved performance of silicon chips.

While we enjoy the improvements in VLSI technology, we are also facing new challenges. The traditional way of improving microprocessor performance by simply scaling down device dimension and increasing the operating frequency does not work that well any more. Modern processors have now reached a hard ceiling in terms of power and operating temperature due to the dramatic increase in circuit speed, integration density and leakage power. The power consumption of a typical modern microprocessor can easily reach as high as 150W [2] (see figure 1.2), which is now

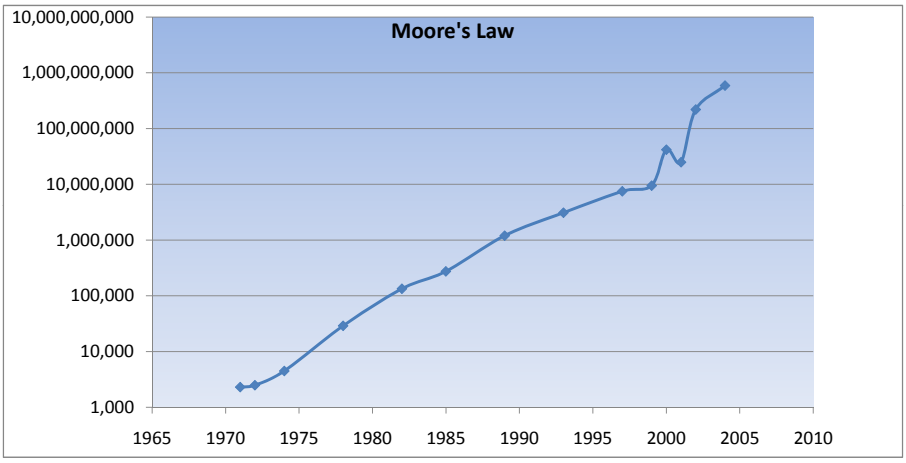



Figure 1.1: Moore's Law for integrated circuit - Intel [1].

approaching the limit of traditional air cooling systems. If the heat removal speed of a cooling system is outpaced by the heat generation rate of the underlying silicon, heat will accumulate within the chip package and cause the temperature of the system to rise quickly. This can lead to many undesirable effects related to the performance, reliability, and life span of a microprocessor. It is reported that about 50% of circuit failures are caused by overheating [40]. Thermal design is thus a critical problem that has to be taken into account from the very beginning of the design process [54].

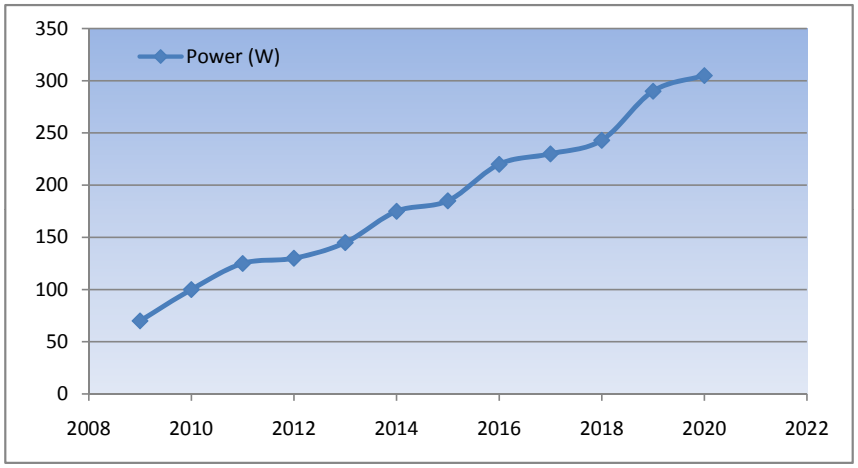



Figure 1.2: Trend of power consumption - ITRS [2].

Motivated by this critical constraint, there is an increasing amount of research

effort that focuses on solving thermal design and thermal management problems. Interestingly, such challenges have been pointed out as early as 1965 in the paper that first introduced Moore's Law: "Will it be possible to remove heat generated by tens of thousands of components in a single silicon chip?" [34]. Even though this problem has been predicted decades ago, it is not until recently that thermal constraint becomes a real obstacle. As we enter the nanometer era, the continuous improvement of the processor performance is now fundamentally limited by whether the cooling system's heat-removing capability can catch up with the heat generation rate of the processor. Although some more aggressive cooling schemes such as liquid cooling or solid-state refrigeration systems are being investigated, the cost, volume and implementation complexity of such schemes are major factors that hinder their adoption. Thus, the current challenge is to effectively manage the operating temperature of a processor to guarantee its thermal safety within the capability of cost effective cooling systems. In the next section, we will discuss some of the impacts that can be caused by elevated temperature.

## 1.2 Impact of High Temperature

### 1.2.1 Thermal Hot Spots

Traditionally, Thermal Design Power (TDP) is often used to guide the design of a system, such as the selection of the cooling solution, the selection of heat sinking material, and *etc.* Essentially, TDP indicates the maximum *total sustained* power dissipated by a microprocessor [40]. Recently, it is observed that TDP alone is no longer sufficient for guiding the thermal design of a system because there exists significant thermal variation across the chip. Such spatial variation can lead to on-chip hot spots (see figure 1.3) where power densities of $300+W/cm^2$ are possible [32]. Note that it is usually required that thermal constraints are satisfied everywhere on the chip to achieve desired thermal safety. In certain cases, even though TDP may be within the capability of the cooling system, some hot spots may have already exceeded a certain thermal threshold. In addition, temperature variation across the

die can cause problems such as mechanical stress and circuit timing errors. Thus, TDP as a lump-sum indicator is no longer adequate, new metrics and approaches which are finer in granularity are needed so that thermal variation across the silicon can be properly modeled, monitored and managed.

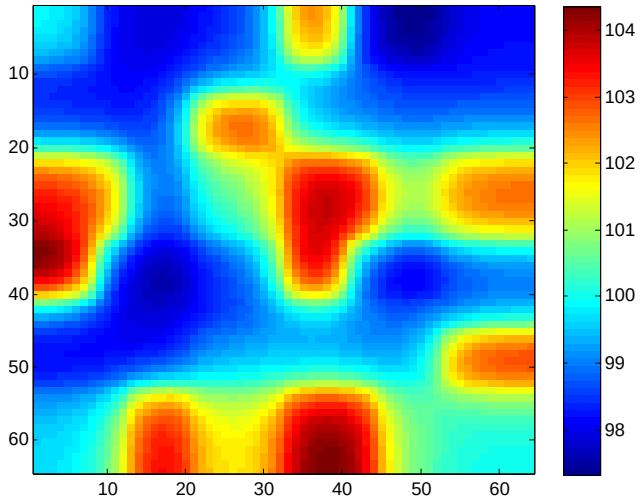
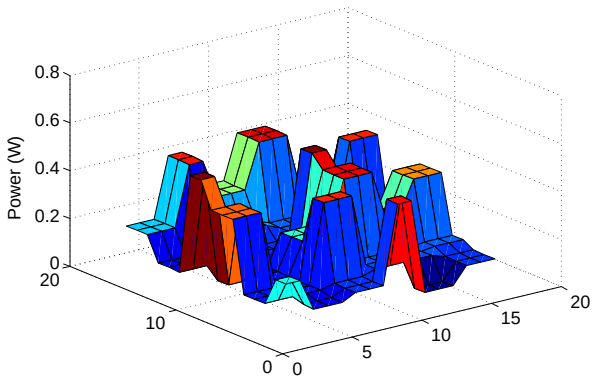
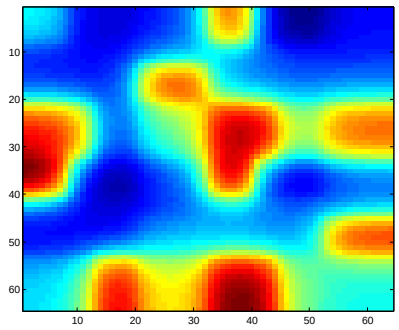



Figure 1.3: On-die hot spots for a typical microprocessor.

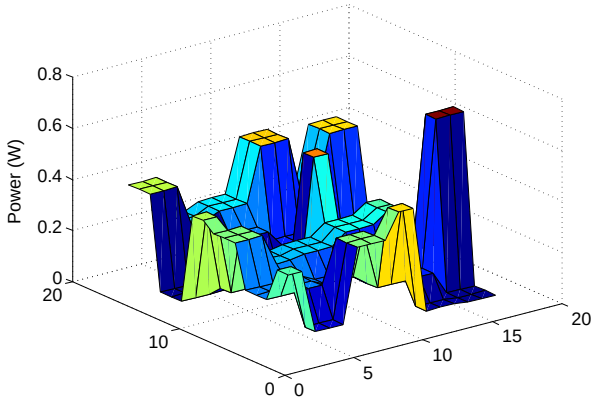
Temperature variation in space is not the only challenge that designers have to face. Today's high performance microprocessors are often general purpose and need to handle many different types of applications. Each application can be very different in its utilization of different circuit components. Some may require integer-heavy operations, others may demand floating-point heavy operations. Thus, the power and thermal profile of a typical processor can exhibit significant temporal variation as well. To demonstrate such variation, we simulated a typical processor using SPEC-2000 benchmarks. The varying power profile directly leads to fast-changing temperature distribution on the chip (see figure 1.4). Figure 1.5 shows how the temperature varies for the arithmetic logic unit (ALU) under different workloads. If many computationally intensive tasks are scheduled densely in time. The peak temperature of the chip can quickly rise to unsafe levels. This poses an opportunity as much as a challenge: if we can schedule the computation tasks well, with heavy workloads divided into smaller pieces that are interleaved with lighter workloads, then the temperature of such a processor can be significantly lowered [21, 44, 16].



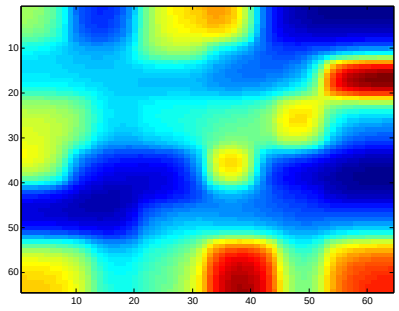

(1) Power profile (application A)




(B) Thermal profile (application A)




(C) Power profile (application B)




$(D)$Thermal profile (application B)

Figure 1.4: On-die power and temperature variation of a typical processor.

For reasons mentioned above, the spatial and temporal variations in temperature can lead to hot spots and can severely impact circuit performance. They have become important design considerations. If left unaddressed, such variations could cause serious timing, accuracy and reliability issues which we will briefly explain next.

### 1.2.2 Impact on Circuit Timing and Accuracy

Elevated temperature and thermal gradient across the die is particularly detrimental to the operation of analog circuits. High thermal gradient can cause mismatches between signal levels and bias currents, therefore degrading the accuracy and reducing the noise margin of such circuits. For example, many analog devices

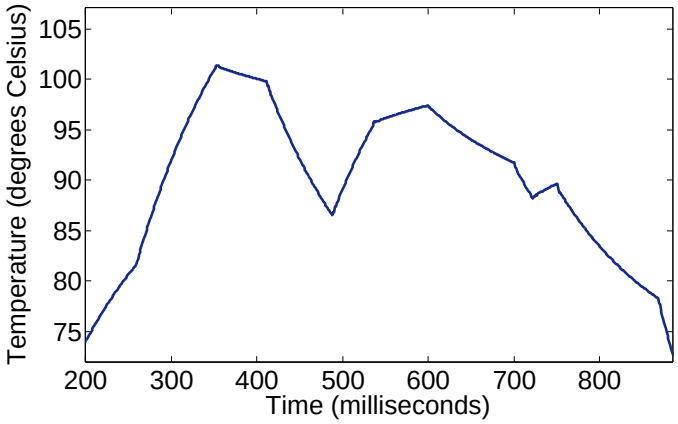



Figure 1.5: Thermal variation of the ALU unit of a typical processor running different applications

employ a constant current biasing substructure which is often implemented using current mirrors (see figure 1.6). However, such structures rely on the assumption that the bias current is very close to the designed value. Unfortunately, due to thermal variations on the chip, there can be significant drifts in the bias current, therefore impacting the accuracy and performance of such analog circuits.

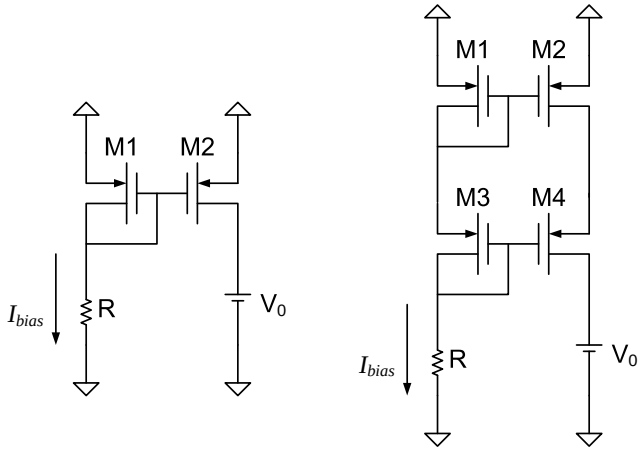



Figure 1.6: The structure on the left is a simple current mirror. The structure on the right is a cascode current mirror.

Thermal variation is undesirable for digital circuit as well since the delay of each logic gate is a strong function of temperature. Higher temperature can lead to slower device switching speed. In addition, thermal gradient can cause the delay

through similar circuit structures to vary significantly from one chip location to another. This makes the design more prone to timing errors when it is operating under heavy thermal stress.

To understand how temperature can impact the operation of digital circuit, let us take a closer look at the timing behavior of a simple inverter. Equation (1.1) shows the transition time for the inverter output to switch from supply voltage ($V_{DD}$) to ground (0) [48]. Here $\mu_n$ is the electron mobility in silicon for NMOS, $C_{ox}$ is the capacitance per unit gate area, $(W/L)_n$ is the width/length ratio for NMOS, $C$ is the effective load capacitance that the inverter is driving. $V_{DD}$ is the supply voltage and $V_t$ is the threshold voltage. The expression for transition time from ground to supply is similar.

$$t_{PHL} = \frac{2C}{\mu_n C_{ox}(W/L)_n(V_{DD} - V_t)} \left[ \frac{V_t}{V_{DD} - V_t} + \frac{1}{2} ln \left( \frac{3V_{DD} - 4V_t}{V_{DD}} \right) \right] \tag{1.1}$$

In this equation, we note that $V_t$ and $\mu_n$ are both temperature sensitive. $V_t$ decreases by about $2mV$ for every 1°C increase in temperature, while $\mu_n$ decreases with an increase of temperature in a more complex relationship [48]. Because the effect of the latter is a more dominant one, the overall effect of a temperature increase is a decrease in circuit switching speed [48]. To achieve the desired circuit operating frequency, we must ensure that there is no timing violation across the chip and across the entire range of potential operating temperature.

Temperature not only impacts the speed of combinational circuit, but also has a profound effect on sequential circuits such as latches, flip-flops and register files. Sequential circuits operate in a synchronous fashion under the control of clock signal. Large thermal variation across the chip can lead to severe clock skews (see figure 1.7), which means the clock signals can be propagated out-of-sync to different chip locations, leading to unexpected timing errors and erroneous circuit behaviors.

### 1.2.3 Impact on Reliability

Higher operating temperature can lead to a greater possibility of reliability issues of silicon chips. In fact, many electronic circuit failures are directly caused by

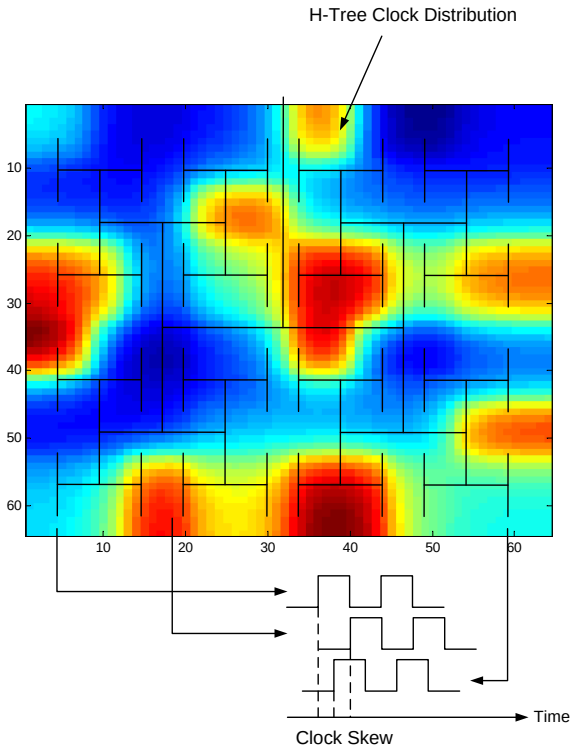



Figure 1.7: Spatial temperature variation causes clock skew.

or related to elevated temperature or high thermal gradient on die [32]. One obvious consideration is that significant thermal gradient can cause mechanical stress, which degrades the reliability of the overall chip system. In addition, temperature impacts reliability through the effects of electromigration (EM), which can cause long-term wear-out of interconnect metal wires.

EM is the process of current-induced transport of material at the atomic level (see figure 1.8). When the current in metal wire is high, current conducting electrons can form *electron wind* which leads to high collision rate with the constituent atoms of metal. This effect will lead to a net flux of metal atoms in the direction of electron flow, creating *voids* (depletion of material) upstream and *hillocks* (accumulation of material) downstream at locations of divergence [32] (see figure 1.9). Electromigration can cause uneven redistribution of resistance, dielectric cracking and eventually, a short between adjacent wires (circuit will fail at this point, marking the end of life for a microprocessor).

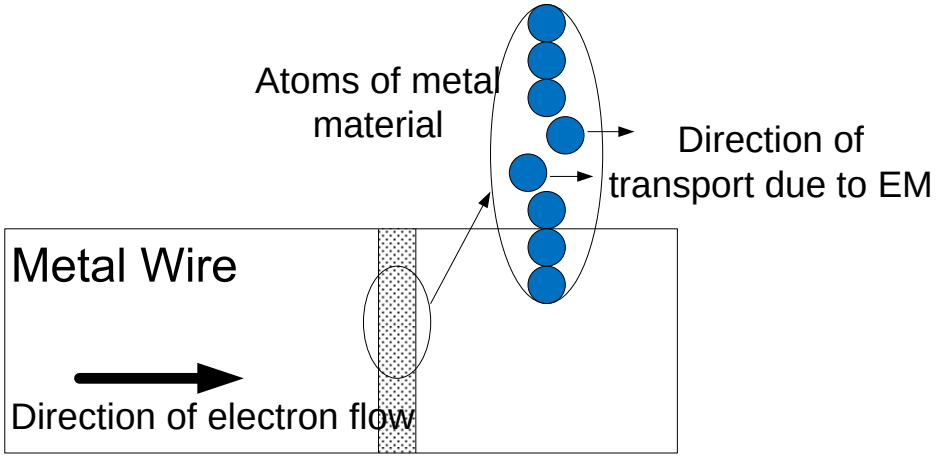



Figure 1.8: Illustration of electromigration effect.

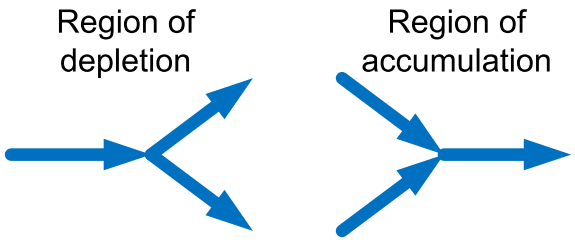



Figure 1.9: Locations where voids and hillocks are normally formed due to electromigration.

Elevated temperature can accelerate the electromigration effects by increasing the random thermal vibration of metal atoms, making them more prone to displacement once collided with an electron. This effect of temperature on circuit reliability is often described and measured using the metric - MTTF (mean time to failure). MTTF due to EM can be calculated using the well-known Black's equation [4]:

$$MTTF = AJ^{-n}e^{Q/kT} \tag{1.2}$$

where $A$ is a process and geometry dependent constant, $J$ is the current density, exponent $n$ is equal to 2 under normal use conditions, $Q$ is the activation energy for grain-boundary diffusion and is equal to $\sim 0.7eV$ for Al-Cu, $k$ is Boltzmann's constant, and $T$ denotes the metal temperature.

### 1.2.4 Impact on Power and Energy

Power and temperature are two closely and intricately related attributes. High power dissipation usually leads to high temperature and vice versa. However, this correlation does not always hold true since their relationship is also affected by spatial and temporal effects. For example, if a certain chip component exhibits low power density, its temperature may not necessarily be low if its surrounding area has high power density. Also, if a certain chip component dissipates a lot of power for only a brief period of time, it may not become hot immediately since it takes time for temperature to rise. On the other hand, temperature could also impact power dissipation through the leakage effect. In the following, we will discuss several different forms of power dissipation. This could help us build a solid foundation for understanding the intricate thermal-power interdependency in later sections.

Total power dissipation in a CMOS circuit can be calculated as:

$$P_{total} = P_{dynamic} + P_{static} + P_{short} \tag{1.3}$$

In equation (1.3), $P_{dynamic}$ represents the dynamic power or switching power that occurs when a logic gate makes a transition. $P_{static}$ is the static power or leakage power that is caused by static current drawn from power supply when the circuit is not switching. Such current is mostly due to gate leakage and threshold leakage, hence the name leakage current. $P_{short}$ represents the power dissipated during a very brief period when a NMOS and a PMOS transistor are potentially both conducting, creating a direct path between the supply and the ground. Usually, $P_{total}$ is dominated by $P_{dynamic}$ and $P_{static}$ which we explain in more detail below.

1. **Dynamic Power:** Dynamic or switching power usually dominates the total power consumption during the active state of a circuit. It can be expressed as follows [40].

$$P_{dynamic} = \frac{1}{2}\alpha C_{load} V_{DD}^2 f \tag{1.4}$$

Here $\alpha$ is the expected number of output transitions in one clock cycle, $C_{load}$ is the load capacitance (including gate input and interconnect capacitances), and $f$ is the clock frequency. When a logic gate makes a transition, its load capacitance is either charged to $V_{DD}$ or discharged to ground. When charging, half of the energy supplied by $V_{DD}$ is stored in the load capacitance and the other half is dissipated as heat. When discharging, that other half previously stored in the capacitance is dissipated as heat, hence the above equation.

Note that when a transistor operates in the active mode, its drain current $I_D$ is roughly proportional to the square of its gate-to-source voltage $V_{GS}$ ($I_D \propto (V_{GS} - V_t)^2$). Thus if $V_{GS} \approx V_{DD}$ and if the threshold voltage $V_t$ is small compared to the supply voltage $V_{DD}$, the switching current is approximately proportional to $V_{DD}^2$. This fact leads to the observation that the switching frequency $f$ is roughly proportional to supply voltage if everything else is kept constant ($f \propto \frac{1}{T_p} \propto \frac{I_D}{V_{DD} \cdot C_{load}} \propto \frac{V_{DD}}{C_{load}}$, where $T_p$ is the period of switching activity and is proportional to the time required to charge/discharge the load capacitance). Thus, we can see from equation (1.4) that $P_{dynamic}$ has a cubic dependency on supply voltage $V_{DD}$, and therefore on frequency $f$. Traditionally, the performance of a microprocessor can be improved by increasing its switching frequency. However, if we double the frequency, it could mean that $P_{dynamic}$ will increase roughly 8 times. A modern microprocessor is already dissipating power in the magnitude of hundred watts. It is apparent that we have hit a power ceiling where we can not afford the cubic increase in power to achieve further performance gains. Performance, or the speed of a circuit, is fundamentally limited by how quickly we can remove the heat generated by it. Designers have therefore switched to the multi-core paradigm in order to keep improving computational throughput without dramatic increase in power consumption and operating temperature [52].

2. **Static Power:** Leakage current is the main reason for static power dissipation. There are three dominant sources for leakage current (see figure 1.10):

(1) reverse-biased junction leakage current ($I_{rev}$); (2) gate direct tunneling leakage ($I_{gate}$); (3) subthreshold leakage ($I_{sub}$).

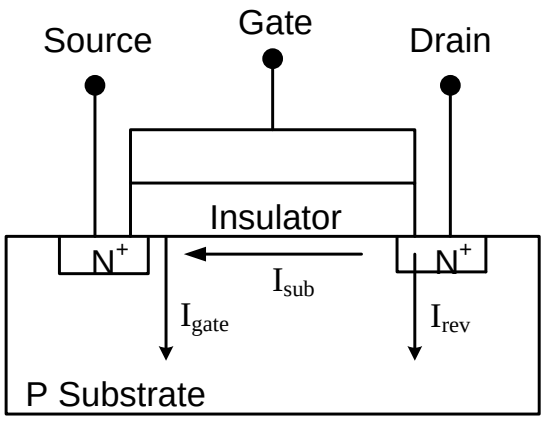



Figure 1.10: Three major leakage current components.

There are many different leakage models proposed in recent literature. For example, in [22], the combined effect of all three leakage currents are approximated using an average current $I_{avg} = I_s(T_0, V_0) \cdot T^2 \cdot e^{K/T}$ and the total leakage power can be modeled as follows:

$$\begin{aligned} P_L &= N_{gate} \cdot V_{DD} \cdot I_{avg} \\ &= N_{gate} \cdot V_{DD} \cdot I_s(T_0, V_0) \cdot T^2 \cdot e^{K/T} \quad (1.5) \\ &= L \cdot T^2 \cdot e^{(K/T)} \quad (1.6) \end{aligned}$$

In the above equation, $P_L$ represents the total leakage power. $N_{gate}$ is the total number of logic gates in a circuit. $V_{DD}$ is the supply voltage. $T_0$ and $V_0$ are reference temperature and voltage, respectively. $I_s(T_0, V_0)$ is the saturation current at $T_0$ and $V_0$. $L = N_{gate} \cdot V_{DD} \cdot I_s(T_0, V_0)$ is a design/technology dependent constant. $K$ is also a technology-dependent constant for a fixed supply voltage. As can be seen, the leakage power has a rather complex relationship with temperature. The overall effect is that the leakage power increases dramatically with temperature.

Traditionally, leakage power is only a small component in total power consumption and dynamic power is usually the dominant one. However, as the

feature size of each technology generation continues to scale down, smaller devices tend to leak more and therefore the impact of leakage power is becoming more and more significant. It is reported that in 90 nanometer technology node, 40% or more of total power consumption is due to leakage [32]. This percentage is expected to increase even further as chips continue to shrink in size. Note that as indicated by equation (1.6), higher temperature leads to more leakage power, and more leakage power will in turn cause higher temperature. This positive feedback effect can significantly increase the total power consumption of a chip system. In some extreme cases, it may even cause thermal runaway where such a positive feedback eventually causes the chip system to fail. Thus, leakage and temperature interdependency poses a serious challenge for the thermal design and management of a chip system.

### 1.2.5 Impact on 3D Chip Technology

A recent advancement in VLSI technology is 3D integration. It exploits the possibility of stacking multiple layers of circuit components along the vertical dimension to form a denser chip structure (see figure 1.11). This new technology can help us integrate even more functionality, storage, and *etc.* into a single chip. However, the biggest obstacle on the way to 3D integration is the overheating problem. It can be expected that the power density and operating temperature of such a tightly integrated system will increase dramatically due to the stacked physical structure [42]. More heat will be generated in limited space while the area of the cooling surface stays approximately the same. Innovative thermal management solutions are thus critical to the adoption and success of this new technology.

## 1.3 Runtime Thermal Tracking and Management

Due to all the detrimental effects of elevated temperature on a chip system, many design-time and run-time techniques have been proposed to achieve thermal safety and maintain a balanced thermal profile in time and space. Thermal-aware

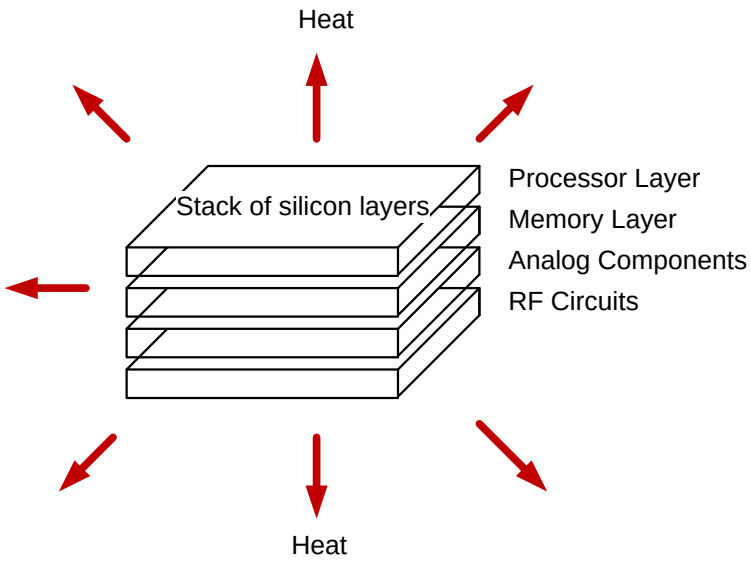



Figure 1.11: As we move into the era of 3D integration, the thermal design issue will become more challenging.

design techniques are special measures that are preventative in nature and should be taken into consideration when designing a chip. For example, designers can employ a multi-$V_t$ library so that circuit components with different threshold voltage can be used on different parts of the circuit. This way leakage power on non-critical paths can be minimized [58, 19, 61]. Thermal aware floorplanning and placement can be performed to arrange high power density units and low power density units in a spatially interleaved fashion to achieve a more balanced thermal profile [42, 23, 38, 47]. Recently, many researchers have investigated possibility of integrating Thermal Electric Coolers (TEC) into chip design and leverage their heat pumping capability to reduce chip temperature [12, 66, 55, 60, 59, 28]. In addition to such thermal-aware design techniques, there are also another set of techniques which are usually applied at runtime to manage the chip temperature in a reactive fashion. Such techniques are called dynamic thermal management techniques (DTM) [5, 18, 45, 8, 29, 51]. For example, dynamic voltage and frequency scaling (DVFS) is a popular technique often implemented in modern processors. Based on runtime temperature feedback (which can be sampled by sensors, for example), a central or several distributed control units will issue commands dynamically to scale down the voltage or frequency (and hence

toggle the speed) of those over-heated chip modules, let them cool down, and then bring them back up to full speed. Clock gating or power gating are also runtime techniques that are more aggressive in nature. They put the circuit components which are under thermal emergency into an OFF state to reduce power and bring down temperature [3, 16]. Runtime thermal management techniques are often very effective since they act based on the real temperature of the chip at runtime. Such information would be very hard to predict at design time since it is difficult to know what kind of workload will be executed and in what sequence. However, as described above, most DTM techniques make trade-offs between performance and thermal safety. Thus, it is important that accurate information regarding the true thermal state of the chip can be obtained so that optimal control decisions can be made to achieve the best balance between thermal safety and maximum throughput.

## 1.4 Organization of the Thesis

In this thesis, we focus on several important areas under the general subject of thermal tracking and estimation. In chapter 2, we propose a chip-level thermal profile estimation methodology using thermal sensors. This approach is based on statistical characteristics of the chip power/thermal behavior. It can take advantage of the correlation that exists among different chip modules and improve the estimation accuracy. It can also overcome the drawbacks of the traditional sensor range based estimation methods. In chapter 3, we propose a statistical framework for implementing a complete and efficient sensor based thermal tracking system on chip. We also discuss how to counter sensor noise as well as how to do sensor placement. In chapter 4, we discuss adaptive approaches that are developed from the theory of Kalman filter. Such adaptive techniques can autonomously detect any change in the chip workload characteristics and adjust the filter parameters to adapt to such changes, therefore providing continuously accurate thermal tracking results. We will also extend the standard linear Kalman filter formulation to account for the non-linear leakage-temperature interdependency. In chapter 5, we proposed an

methodology for extracting the chip power statistical characteristics automatically. Finally, in chapter 6 we conclude this thesis.

# Chapter 2

# Chip Level Thermal Profile Estimation

## 2.1 Motivation

To maximize thermal reliability and avoid detrimental effect of elevated temperature on silicon chips, many thermal management techniques have been proposed. These techniques are designed to help ensure thermal safety for modern high performance processors. The essence of such techniques is to trade off performance for less power dissipation. For example, the operating frequency and supply voltage could be scaled down to temporarily reduce power and hence temperature. Components with low utilization rate can be shutdown periodically. Scheduling algorithms could also be applied to distribute the workload more evenly in time and among different CPU cores, thus reducing the peak temperature of the chip [21, 44, 16]. Though these techniques can be powerful, one fundamental requirement is that they must know the accurate thermal state of the chip during runtime. If such knowledge is not available or inaccurate, improper thermal control decisions can be made which could impact the system performance and reliability in two ways: (1) If the thermal control decisions are too aggressive, the chip performance could be throttled unnecessarily. (2) If the thermal control decisions are too conservative, they may not be able to act in an effective and efficient fashion to quickly resolve a thermal emergency. Due to above reasons, estimating the accurate thermal state of the chip at runtime has become a crucial problem which has inspired several new research directions. For example, several researchers have proposed a simulation based strategy [65, 62, 29] in which the thermal behavior of the chip is captured using a set of representative thermal profiles, each of which is simulated based on a typical workload. These simulated thermal profiles, together with some runtime workload feedback, could then be used to predict the thermal state of the chip at runtime. These simulation-

based methods have several drawbacks: (1) the types of applications and the order in which they are scheduled to run are highly dynamic for a modern multi-purpose processor. Different application combinations and sequence of execution could lead to different thermal profiles which can be difficult to pre-characterize accurately. (2) Fabrication process and operating environment for a certain chip can introduce many types of variability such as supply voltage fluctuation, lithography induced device dimension variations, and *etc.* Such variability adds another level of uncertainty to the runtime thermal behavior. In order to estimate the thermal state of a chip accurately at runtime, thermal sensors are placed on chip to provide temperature sampling at runtime [13, 35, 36, 15, 39, 27, 41, 9]. The readings of these sensors are good indicators of the real operating temperature and therefore can be send to a thermal control unit to help it make effective thermal control decisions. However, such sensor observations are mostly used in a very ad-hoc way [49]. In this chapter we propose a statistical methodology for recreating the complete chip level thermal profile based on limited sensor observations.

For any sensor based thermal profile estimation problem, there are several challenges that need to be addressed: 1) the number of on-chip sensors is limited due to resource/power considerations; 2) sensor placement is constrained to locations where there is enough spatial slack (note that it is not practical to put sensors in pre-designed IP modules); 3) sensors can be noisy due to supply voltage fluctuations, fabrication variability, cross coupling and etc. In this chapter, we focus on the thermal profile estimation problem where we assume the sensor locations are predetermined and known. Now, given a few runtime sensor observations, our goal is to estimate the temperature across the entire chip, in spite of where sensors are placed. We will discuss a sensor placement methodology in later chapters.

Our proposed approach is inspired from signal detection and estimation theory. Essentially, we exploit the sensor observations and also the fact that there exists a high degree of correlation in power dissipation between different chip modules. By exploiting this correlation, even a few thermal sensors can be used to generate accurate chip level thermal profile estimates. Our technique is optimal (no other

methods can generate a better estimation) if the randomness associated with the power density of different chip modules is jointly Gaussian. We also develop an effective heuristic based on moment matching if the power density values exhibit non-Gaussian nature [75].

## 2.2 Preliminary - Modeling Thermal Profile of a Chip System

In this section, let us first examine the thermodynamics of a chip system and model its steady-state thermal state assuming the power density of the design is given. The typical structure of an integrated chip includes the silicon on which the circuit is built and the heat sink associated with the silicon within the package (see figure 2.1).

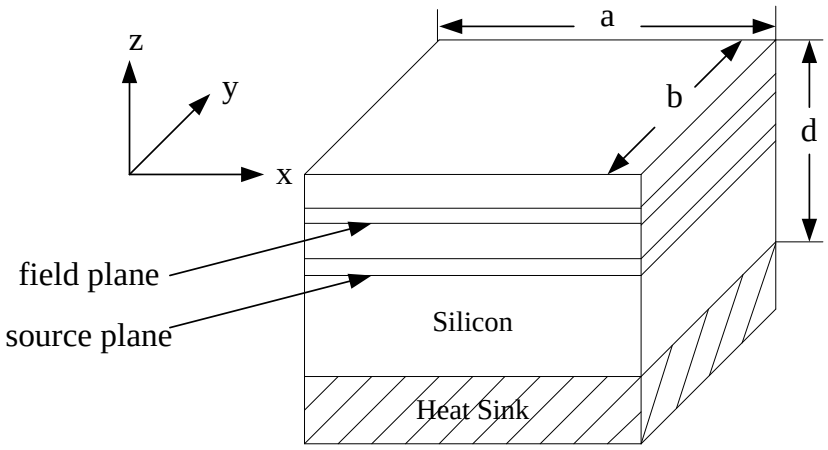



Figure 2.1: Silicon chip and the associated heat sink.

In figure 2.1, the source plane is defined as the thin layer of silicon where power is dissipated (source of heat generation) and the field plane is defined as the surface layer on which a thermal profile is of interest. For brevity in description, we assume that silicon has uniform thermal conductivity. The steady state temperature distribution inside the chip is governed by Poisson's equation [65]:

$$\nabla^2 T(r) = -\frac{P'_d(r)}{k_s} \tag{2.1}$$

subject to the following boundary conditions:

$$\begin{cases} \left.\frac{\partial T(r)}{\partial x}\right|_{x=0,a} = \left.\frac{\partial T(r)}{\partial y}\right|_{y=0,b} = \left.\frac{\partial T(r)}{\partial z}\right|_{z=0} \\ \left. k_s \frac{\partial T(r)}{\partial z}\right|_{z=-d} = h(T(r)|_{z=-d} - T_0) \end{cases} \tag{2.2}$$

In the above equations, $r = (x, y, z)$ and $T(r)$ is the temperature distribution in the silicon (in °C), $P'_d(r)$ is the volume power density (in $W/m^3$) and $k_s$ is the thermal conductivity of silicon (in $W/(m \cdot^\circ \mathrm{C})$). The vertical and top surfaces are assumed to be adiabatic whereas the interface between the silicon and the heat sink is assumed to be convective with an effective heat transfer coefficient $h$ (in $W/(m^2 \cdot^\circ \mathrm{C})$). The ambient temperature is denoted by $T_0$. Note that the boundary conditions highlighted above are specific to the package design. Although different packages with varying heat sink properties could change the boundary conditions, the general nature of the solution to the Poisson's equation will not change. The partial differential equation (2.1) can be solved using the method of Green's function. For brevity we do not go into the details of the derivation of this solution which can be found in [65, 37]. We only show that the solution can be expressed as follows:

$$T(r) = T_0 + \int_0^a \int_0^b \int_{-d}^0 G(r, r') P'_d(r') dx' dy' dz' \tag{2.3}$$

$$G(r, r') = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} Z_{mn}(z, z') cos(m\pi x/a) cos(n\pi y/b) cos(m\pi x'/a) cos(n\pi y'/b) \tag{2.4}$$

Here $G(r, r')$ is the Green's function. $r = (x, y, z)$ and $r' = (x', y', z')$ are the coordinates of an arbitrary point on the field plane and the source plane, respectively. $P'_d(r')$ is the volume power density at point $r'$. $Z_{mn}$ is a function of $z$ and $z'$ and can be calculated based on the boundary conditions (see details in [65, 37]). For any specific values of $z$ and $z'$, the above solution can be simplified to its 2-dimensional form:

$$T(x, y) = T_0 + \int_0^a \int_0^b G(x, y, x', y') P_d(x', y') dx' dy' \tag{2.5}$$

$$G(x, y, x', y') = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} C_{mn} cos(m\pi x/a) cos(n\pi y/b) cos(m\pi x'/a) cos(n\pi y'/b) \tag{2.6}$$

Here $T(x, y)$ is the temperature at an arbitrary point on the field plane and $P_d(x', y')$ is the 2D power density function (in $W/m^2$) for any point on the source plane. Constants $C_{mn}$ can be derived from $Z_{mn}(z, z')$. Note that the only unknown in these equations is the power density function $P_d$ which depends on the layout, device dimensions, switching activity, leakage, and *etc.*

For efficiency purpose and ease of computation, we would like to work in the discrete space. If we split the source and field planes into $J \times J$ and $I \times I$ grids respectively (figure 2.2), then each source grid $j$ can be assumed to have a constant power density $P_d(j)$ and each field grid $i$ can be represented by the average temperature $T(i)$ inside the grid (note that these grids can be arbitrarily small).

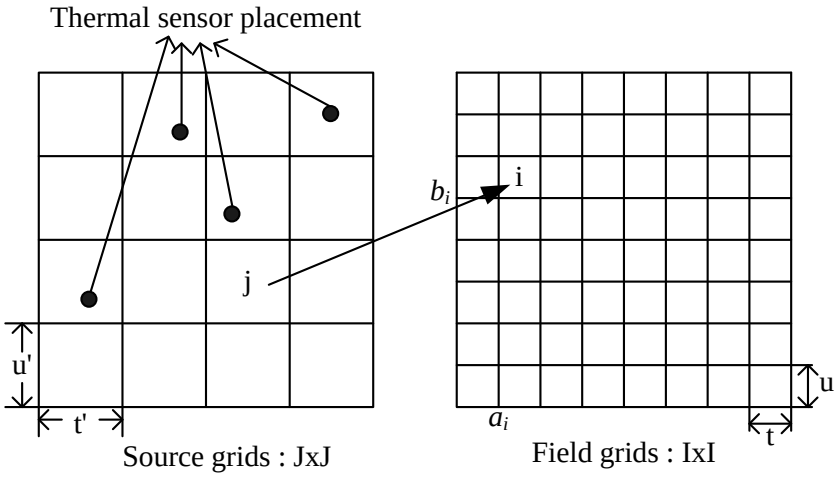



Figure 2.2: Source plane and field plane of a silicon chip.

Assuming we know $P_d(j)$, we are interested in calculating $T(i)$ which can be expressed as follows:

$$T(i) = \frac{1}{t \cdot u} \int_{a_i}^{a_i+t} \int_{b_i}^{b_i+u} T(x, y) dx dy \tag{2.7}$$

where $(a_i, b_i)$ are the 2D coordinates of the bottom left corner of the $i$-th grid cell (origin being the bottom left corner of the chip). $t$ and $u$ are the width and height of each cell on the field plane ($t = a/I$, $u = b/I$ assuming $a$, $b$ are the chip dimensions). Since the power density on the source plane is assumed to be a constant $P_d(j)$ in each grid cell $j$, we can substitute (2.5) into (2.7) and get

$$T(i) = T_0 + \sum_j \frac{P_d(j)}{t \cdot u} \int_{a_i}^{a_i+t} \int_{b_i}^{b_i+u} \int_{a_j}^{a_j+t'} \int_{b_j}^{b_j+u'} G(x, y, x', y') dx dy dx' dy' \quad (2.8)$$

Here $t'$ and $u'$ are the dimensions for each grid cell on the source plane (figure 2.2). $G(x, y, x', y')$ is again the Green's function as shown by equation (2.6). It can be seen that once we know the accurate power density profile, we can calculate the steady-state thermal profile analytically based on equation (2.8).

## 2.3 Problem Description

Recently much attention has been given to the subject of placing on-chip thermal sensors during design time and exploiting the sensor readings at runtime to perform dynamic thermal/power management [35, 36, 15, 39, 27, 10]. The central motivation behind such approaches is the growing need for sophisticated runtime management techniques to control the detrimental effects of unpredictable thermal hotspots. A key challenge, which is also the focus of this chapter, is how we can systematically reconstruct the chip-level thermal profile using the thermal sensor observations. Unfortunately, this problem is not trivial. The challenges are discussed below.

1. The total number of on-chip thermal sensors is limited and thus cannot cover all areas of a chip. Indeed, if we have the freedom to place infinite number of sensors at all locations, there would be no need for thermal profile estimation since the temperature at any location is accurately known. Unfortunately, since thermal sensors come with a cost in terms of area and power, we do not have the luxury to place infinite number of sensors under todays ever pushing design constraints. In addition, the sensors can not go into the pre-designed on-chip IP cores should that be the critical area of interest. The thermal profile estimation problem is thus highly constrained by the number and the locations of the thermal sensors. The key challenge here is to take the readings

of only a few temperature sensors and re-create the chip level thermal profile as accurately as possible.

2. To solve the thermal profile estimation problem one has to account for the underlying randomness in power density. Let us consider equation (2.8) again which is used to compute thermal profile of a chip. Assuming that the silicon thermal conductivity and heat transfer coefficient are constants, the only unknown in equation (2.8) is $P_d(j)$. If we knew $P_d(j)$ accurately at design time, then we would not need any thermal sensors at all since the entire thermal profile could be computed analytically. However, the power dissipation of a chip is a strong function of application workload (which is unpredictable until runtime), device parameters (which are random due to fabrication variability), the environment (such as the ambient temperature change and supply voltage fluctuation). All these unpredictable factors make power density a random quantity in reality. Therefore thermal profile becomes a random quantity as well. The key challenge here is to develop a probabilistic methodology to account for the random nature of power density and use it to effectively estimate the chip-level thermal profile, when given a few temperature sensor observations at runtime.

3. As discussed above, the desired methodology must account for the fact that we only have a few sensors and the underlying power density is random. An important observation about this randomness is that different parts of the chip can exhibit highly correlated power behavior due to similarity in their activity. The random fabrication parameters which indirectly affect power dissipation (such as channel length, oxide thickness, and *etc.*) also exhibit strong spatial correlation. Therefore it can be concluded that the random power density in different parts of the chip exhibits strong correlation. We can take advantage of this property and use the power density in certain parts of the chip to predict the power density in other parts. For this reason, even if the number of thermal sensors is limited, reasonably accurate thermal profile could

be estimated by exploiting this correlation. It is reasonable to assume that the probabilistic properties of chip power density, such as mean, correlation, variance, covariance, and *etc.*, can be characterized *a priori* through extensive simulations and experiments (this will be demonstrated in the next section).

All in all, the key challenge is to exploit the temperature readings of a few thermal sensors, along with knowledge of the random characteristics of chip power density to generate accurate thermal profile estimates. As would be highlighted in the experimental results, ignoring the correlation information leads to highly inaccurate estimates of thermal profile even with relatively more sensors whereas exploiting the correlation information enables high fidelity thermal estimates with only a few thermal sensor readings.

## 2.4 Modeling Randomness in Power Density

In order to quantify the correlated power behavior mentioned earlier, we used Wattch [6] with alpha binary to generate the power consumption results for different parts of a processor. We simulated a high performance aggressive out-of-order processor with pipeline width of 8 instructions and an instruction window of 128 instructions. Level 1 caches (both instruction cache and data cache) are 32KB 4-way set associative. All the caches in the hierarchy are using LRU replacement policy and a block size of 64 bytes. For benchmarks, we simulated all the SPEC 2000 CPU benchmark suite [20] compiled with the default parameters provided with the suite. We bypassed the startup part, based on simpoint [20], and simulated a representative 250M instructions for each of the benchmarks.

Figure 2.3 highlights the variation of power dissipation across different benchmarks for the instruction window module. This demonstrates that the power consumption of a chip module is indeed a random quantity whose characteristics can be captured by a probabilistic distribution. The correlations in power dissipation of different modules are computed from the benchmark simulation and are presented in table 2.1. It can be seen that different components have a high degree of correlation

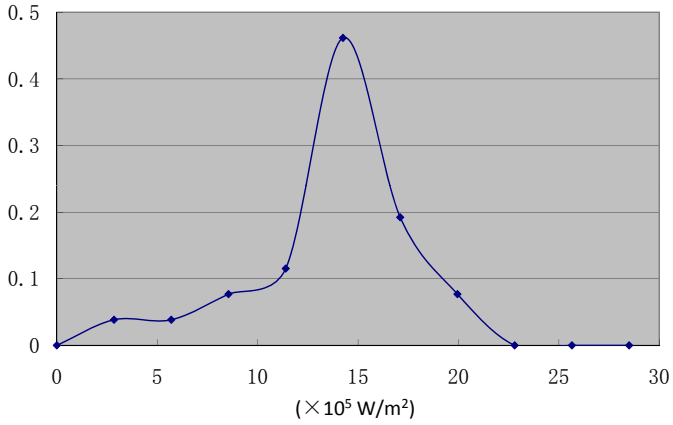



Figure 2.3: Power density distribution for instruction window module.

in power dissipation. For example, the correlation between instruction window and register rename modules is as high as 0.91. This means that for a given application running on the processor, if there is high power density in instruction window module, then the power density for register rename module is likely to be high as well.

Table 2.1: Statistical power density characteristics of different chip modules

| correlation | branch predictor | rename | instruction window | load/store queue | register file | ALU | Icache | Dcache |
|---|---|---|---|---|---|---|---|---|
| Dcache | 0.42 | 0.54 | 0.69 | 0.98 | 0.42 | 0.47 | 0.56 | |
| Icache | 0.66 | 0.97 | 0.89 | 0.61 | 0.58 | 0.68 | | |
| ALU | 0.36 | 0.74 | 0.90 | 0.51 | 0.93 | | | |
| register file | 0.13 | 0.63 | 0.84 | 0.47 | | | | |
| load/store queue | 0.45 | 0.58 | 0.74 | | | | | |
| instruction window | 0.54 | 0.91 | | | | | | |
| rename | 0.61 | | | | | | | |
| mean ($10^5 W/m^2$) | 0.34 | 4.33 | 12.48 | 1.09 | 3.88 | 3.30 | 4.71 | 1.03 |
| std-dev ($10^5 W/m^2$) | 0.18 | 1.68 | 4.21 | 0.42 | 1.49 | 1.33 | 1.87 | 0.41 |

To model this random yet correlated power behavior, we discretize the chip into $J \times J$ grids and use $P_d(j)$ to represent the power density in grid cell $j$ (as we explained in section 2.2). This $P_d(j)$ is a random variable which can be expressed

as follows:

$$P_d(j) = F\Big(SW(j), V_t(j), L_{eff}(j), V_{DD}(j), \ldots\Big) \tag{2.9}$$

Equation (2.9) highlights the general dependence of the power density $P_d(j)$ on its local switching activity $SW(j)$, device parameters like threshold voltage $V_t(j)$ and effective channel length $L_{eff}(j)$, the environment factor like supply voltage $V_{DD}(j)$ and *etc.* Because of these random parameters, $P_d(j)$ is random as well. Various simulation and analytical techniques can be carried out to extract the probabilistic characteristics of $P_d(j)$ (for $\forall j$) including the mean, variance and covariance [11, 14, 64].

For example, instruction issue register will exhibit correlated behavior in switching activity with other computational modules. Similarly, the fabrication variability of devices in close vicinity would be correlated as well thereby leading to correlation in their power density. Knowing the correlations in various parameters in equation (2.9) would allow us to estimate the correlation between different $P_d(j)$ variables. There are already many existing works on modeling the correlated fabrication variability [7]. Extensive simulation could also be used to obtain the correlations in the switching activity, leakage and *etc.* of different grid cells (similar to the simulation that we did earlier to extract the correlation data shown in table 2.1). All these could be used to extract the correlations among various power density variables. We do not go any further into the details of how this modeling can be done since the focus of this work is different. We only assume that the $J \times J$ random power density variables could be represented as a random vector $\vec{P}$ with mean $\vec{\mu}_p$ and covariance matrix $\Sigma_{pp}$ (*note that we do not assume the power densities follow any specific distribution, we just assume that the mean and covariance are known*).

Though the values obtained here are not completely accurate, the key idea is that these approximate values, when combined with the runtime sensor observations, could significantly increase the accuracy of thermal profile estimation.

## 2.5 Estimation Methodology

### 2.5.1 Formal Problem Statement

Let us use vector $\vec{P}$ to represent the power density values at all $J \times J$ grid locations. Similarly the thermal profile of the chip can be represented by vector $\vec{T}$.

$$\vec{P} = [P_1, P_2, \ldots, P_{J^2}]'$$

$$\vec{T} = [T_1, T_2, \ldots, T_{I^2}]'$$

Now suppose we have $n$ on-chip temperature sensors at the following locations:

$$s_1 = (x_1, y_1), s_2 = (x_2, y_2), \ldots, s_n = (x_n, y_n)$$

These sensors provide a vector of $n$ temperature readings $\vec{T_s}$.

$$\vec{T_s} = [T_{s_1}, T_{s_2}, \ldots, T_{s_n}]'$$

We also assume that we have characterized *a priori* the mean $\vec{\mu}_p$ and covariance matrix $\Sigma_{pp}$ for $\vec{P}$. The problem is to find the most probable thermal profile of the chip ($\vec{T}$) based on such information. According to signal estimation theory [43], the optimal estimator (in terms of the mean square error) for our problem is the conditional expectation of $\vec{T}$ given $\vec{T_s}$, $\vec{\mu}_p$ and $\Sigma_{pp}$:

$$E(\vec{T}\big|\vec{T_s}, \vec{\mu}_p, \Sigma_{pp}) \tag{2.10}$$

Note that we are exploiting both the sensor readings and the probabilistic characteristics of power density to provide accurate estimation. Exploiting the latter would enable us to use fewer sensors while providing high fidelity chip level thermal estimates.

### 2.5.2 Optimal Solution for Jointly Gaussian Distribution

Now we present an approach that would allow us to solve the problem posed by (2.10). In this section we consider the case where the power density variables

are jointly Gaussian in nature. Let us consider (2.8) once again which could be rewritten as follows:

$$T(i) = T_0 + \sum_j \alpha_{i,j} P_d(j) \tag{2.11}$$

where

$$\begin{aligned}\alpha_{i,j} =& \frac{1}{t \cdot u} \int_{a_i}^{a_i+t} \int_{b_i}^{b_i+u} \int_{a_j}^{a_j+t'} \int_{b_j}^{b_j+u'} G(x, y, x', y') dx dy dx' dy' \\ =& \frac{1}{t \cdot u} \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} C_{mn} \int_{a_i}^{a_i+t} cos\left(\frac{m\pi x}{a}\right) dx \int_{b_i}^{b_i+u} cos\left(\frac{n\pi y}{b}\right) dy \\ & \int_{a_i}^{a_i+t'} cos\left(\frac{m\pi x'}{a}\right) dx' \int_{b_i}^{b_i+u'} cos\left(\frac{n\pi y'}{b}\right) dy'\end{aligned} \tag{2.12}$$

The infinite summation in variables $m$ and $n$ can be approximated by a finite summation where $m$ and $n$ are truncated at $M$ and $N$. In [65], it was demonstrated that setting $M$ and $N$ to around 64 leads to satisfactory accuracy in thermal estimation. Therefore the constants $\{\alpha_{i,j}, \text{for } \forall i, j\}$ can be computed using equation (2.12) and this approximation. Now equation (2.11) can be rewritten in its vector form where $\alpha_{i,j}$ (and hence matrix $A$) is a constant and known:

$$\vec{T} = \vec{T_0} + A\vec{P} \tag{2.13}$$

$$A = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,J^2} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,J^2} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{I^2,1} & \alpha_{I^2,2} & \cdots & \alpha_{I^2,J^2} \end{pmatrix} \tag{2.14}$$

Again, $\vec{T_0}$ is the ambient temperature. Equation (2.13) highlights that the dependency between temperature and power density can be approximated with a linear transformation [65]. Since $\vec{P}$ is a random vector with mean $\vec{\mu}_p$ and covariance matrix $\Sigma_{pp}$, $\vec{T}$ is a random vector too with mean and covariance as follows [43]:

$$\vec{\mu}_T = T_0 + A\vec{\mu}_p \tag{2.15}$$

$$\Sigma_{TT} = A\Sigma_{pp}A^T \tag{2.16}$$

Now that we know the relationship between $\vec{T}$ and $\vec{P}$ is linear, estimating the thermal profile can be simplified to estimating the power density due to the

following:

$$E(\vec{T}\big|\vec{T_s}) = T_0 + AE(\vec{P}\big|\vec{T_s}) \tag{2.17}$$

Equation (2.17) is obtained by taking the conditional expectation of equation (2.13) on both sides. Note that for brevity we have omitted $\vec{\mu}_p$ and $\Sigma_{pp}$ from the notation. It can be seen that the problem posed by (2.10) can be solved optimally by finding the optimal estimator for the following:

$$E(\vec{P}\big|\vec{T_s}) \tag{2.18}$$

The advantage of first solving (2.18) and then (2.10) is that in general the statistical properties such as correlation are easier to extract for power density (as we demonstrated in section 2.4). After generating an estimate for power density, we can then easily obtain the desired thermal estimate through (2.17).

Now we present an optimal solution to (2.18) when the power density vector is jointly Gaussian. This is a reasonable assumption in reality due to the principle of Central Limit Theory [43] (also see the simulation result in figure 2.3). A heuristic solution for the non-Gaussian distribution case will be presented in the next section. As mentioned earlier, $\vec{T_s}$ is the vector of sensor readings. Since we know the location of these sensors, we can relate $\vec{T_s}$ to $\vec{P}$ as follows.

$$\vec{T_s} = T_0 + A_s\vec{P} \tag{2.19}$$

where $A_s$ is a submatrix of $A$ formed by selecting the rows corresponding to sensor grids. For example, if we have 3 sensors placed at the $\{5, 11, 23\}^{th}$ grid cells, then $A_s$ is simply formed as follows:

$$A_s = \begin{pmatrix} \alpha_{5,1} & \alpha_{5,2} & \dots & \alpha_{5,J^2} \\ \alpha_{11,1} & \alpha_{11,2} & \dots & \alpha_{11,J^2} \\ \alpha_{23,1} & \alpha_{23,2} & \dots & \alpha_{23,J^2} \end{pmatrix} \tag{2.20}$$

Now given the sensor observations $\vec{T_s}$, the statistical power density characteristics $\{\vec{\mu}_p, \Sigma_{pp}\}$, and the linear transformation between $\vec{T_s}$ and $\vec{P}$ shown by (2.19), we would like to estimate the value of the power density vector $\vec{P}$. This can be

achieved by calculating the conditional expectation which is the optimal estimator in terms of the mean square error. Under the assumption that the randomness in power density $\vec{P}$ is jointly Gaussian, we have the following closed form solution:

$$E(\vec{P}\big|\vec{T}_s) = \vec{\mu}_p + \Sigma_{ps}\Sigma_{ss}^{-1}(\vec{T}_s - \vec{\mu}_s) \tag{2.21}$$

$$= \vec{\mu}_p + \Sigma_{pp}A_s^T(A_s\Sigma_{pp}A_s^T)^{-1}(\vec{T}_s - A_s\vec{\mu}_p - T_0) \tag{2.22}$$

In equation (2.21), $\vec{\mu}_s$ is the average of $\vec{T}_s$ across all observations and $\vec{\mu}_s = T_0 + A_s\vec{\mu}_p$ based on (2.19). $\Sigma_{ps}$ is the covariance matrix between $\vec{P}$ and $\vec{T}_s$, $\Sigma_{ss}$ is the covariance matrix for $\vec{T}_s$ itself. These two covariance matrices can be calculated based on (2.19) as well ($\Sigma_{ps} = \Sigma_{pp}A_s^T$, $\Sigma_{ss} = A_s\Sigma_{pp}A_s^T$), therefore we have the final solution (2.22). Note that this is the analytical solution to (2.18) and hence is the optimal estimator for the power density vector $\vec{P}$ given sensor observations $\vec{T}_s$ (for detailed proof, please see [43]). Also note that $\Sigma_{ps}\Sigma_{ss}^{-1}$ in equation (2.21) is a constant matrix that can be computed in advance. Therefore this solution is a simple linear estimator and can be computed very efficiently for each observation $\vec{T}_s$. Once $E(\vec{P}\big|\vec{T}_s)$ is computed, estimating the thermal profile (vector $\vec{T}$) can be simply done using equation (2.17) which we restate here:

$$E(\vec{T}\big|\vec{T}_s) = T_0 + AE(\vec{P}\big|\vec{T}_s)$$

Equation (2.22) is the crux of our work. Its optimality indicates that no other linear estimator can perform better in terms of the mean square error. In this sense our methodology gives the optimal solution for the thermal profile estimation problem for the jointly Gaussian case. The general flow of our method is shown in Fig. 2.4.

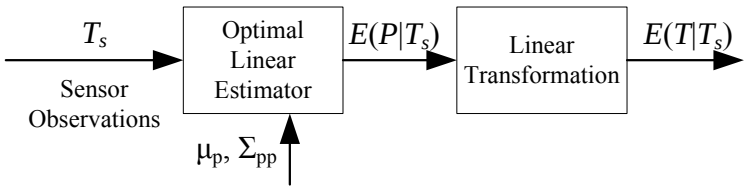



Figure 2.4: Estimation flow.

### 2.5.3 Heuristic Solution for Non-Gaussian Distribution

When the underlying randomness of power density does not exhibit Gaussian nature, the estimator presented in equation (2.22) is not optimal anymore. However, for a non-Gaussian distribution, a closed form solution is often hard to find, sometimes even impossible. In this section, we present a heuristic algorithm to approximate the solution to (2.18) when the power density variables have non-Gaussian distribution.

In general, real data is not far from a Gaussian distribution due to Central Limit Theory [43]. Under these scenarios, we can approximate the actual joint probability of the power density vector with a Gaussian distribution. The approximation is based on the moment-matching approach. This enables us to use the estimator of equation (2.22) to approximate the result of (2.18).

Suppose the power density vector $\vec{P}$ has non-Gaussian joint probability density function (*abbr.* JPDF) $f_N(\vec{P})$. We would like to approximate $f_N(\vec{P})$ using a simplified Gaussian JPDF $f_G(\vec{P})$. The approximation is done by matching the characteristic functions of $f_N(\vec{P})$ and $f_G(\vec{P})$. The characteristic function is defined as the Fourier transform of any JPDF function. For brevity, here we only show the two variable case (where the vector $\vec{P}$ has 2 elements), the multi-variable case can be generalized easily. The characteristic function of any JPDF $f_{X_1,X_2}(x_1, x_2)$ is

$$\phi(y_1, y_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{i(y_1x_1+y_2x_2)} f_{X_1,X_2}(x_1, x_2)dx_1dx_2 \tag{2.23}$$

Expanding the exponential term in the above equation gives a series representation of $\phi(y_1, y_2)$:

$$\begin{aligned}\phi(y_1, y_2) =& 1 + iy_1 \iint x_1 f_{X_1,X_2}(x_1, x_2)dx_1dx_2 + iy_2 \iint x_2 f_{X_1,X_2}(x_1, x_2)dx_1dx_2 \\ &- \frac{y_1^2}{2} \iint x_1^2 f_{X_1,X_2}(x_1, x_2)dx_1dx_2 - \frac{y_2^2}{2} \iint x_2^2 f_{X_1,X_2}(x_1, x_2)dx_1dx_2 \\ &- y_1y_2 \iint x_1x_2 f_{X_1,X_2}(x_1, x_2)dx_1dx_2 + \dots\end{aligned} \tag{2.24}$$

In the above equation, the coefficients of $y_1$ and $y_2$ are defined as the moments of

$f_{X_1,X_2}$ and can be formalized below:

$$m_{ij} = E(x_1^i x_2^j) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1^i x_2^j f_{X_1,X_2}(x_1,x_2)dx_1dx_2 \tag{2.25}$$

Therefore the characteristic function of $f_{X_1,X_2}(x_1,x_2)$ can be represented as an infinite series in terms of the moments:

$$\phi(y_1,y_2) = 1 + iy_1m_{10} + iy_2m_{01} - \frac{y_1^2}{2}m_{20} - \frac{y_2^2}{2}m_{02} - y_1y_2m_{11} + \ldots \tag{2.26}$$

Note that $m_{10} = E(x_1) = \mu_{x_1}$ is simply the mean of $x_1$. Similarly $m_{01} = \mu_{x_2}$, $m_{11} = E(x_1,x_2) = Cov(x_1,x_2) + \mu_{x_1}\mu_{x_2}$, $m_{20} = E[x_1^2] = Var(x_1) + \mu_{x_1}^2$ and so on.

Now let us suppose that we want to approximate $f_{X_1,X_2}$ with $f^G_{X_1,X_2}$ where $f^G_{X_1,X_2}$ is a bivariate Gaussian distribution. For this we match the moments of the two distributions so as to evaluate the unknowns of $f^G_{X_1,X_2}$. Since $f^G_{X_1,X_2}$ is Gaussian, it has 5 unknowns $\{\mu_{x_1},\mu_{x_2},\sigma_{x_1},\sigma_{x_2},\rho_{x_1,x_2}\}$ which can be obtained by matching the first 5 moments of $f_{X_1,X_2}$, *i.e.*

$$m_{10} = m^G_{10}, \quad m_{01} = m^G_{01}, \quad m_{20} = m^G_{20}, \quad m_{02} = m^G_{02}, \quad m_{11} = m^G_{11} \tag{2.27}$$

Thus in order to approximate $f_N(\vec{P})$ with $f_G(\vec{P})$, the moments of the two distributions are matched, or equivalently the characteristic functions are matched. Further, because the Gaussian JPDF $f_G(\vec{P})$ is completely determined by its mean vector ($n$ unknowns) and covariance matrix ($n(n+1)/2$ unknowns). Here we assume $n$ is the number of elements of $\vec{P}$. Hence there are a total of $n(n+3)/2$ unknowns. By matching the first $n(n+3)/2$ moments, we can determine the Gaussian JPDF approximation.

In this way, we can take the prior knowledge of the non-Gaussian distribution of $\vec{P}$ and fit a Gaussian model to it. This enables us to use the estimator of equation (2.22) as an approximation. The accuracy of this heuristic algorithm will be demonstrated by our experimental results.

## 2.6 Experimental results

In this section we present the results obtained using our estimation methodology. We simulated a high performance aggressive out-of-order processor with

pipeline width of 8 instructions and an instruction window of 128 instructions. Level 1 caches (both instruction and data) are 32KB 4-way set associative. The shared level 2 cache is 1MB and 8-way set associative. All the caches in the hierarchy are using LRU replacement policy and a block size of 64 bytes. Firstly we simulated this architecture using Wattch [6] and all the SPEC 2000 CPU benchmark suite [20]. The power distribution data was generated for each functional module of the processor across all different benchmarks. Using this data, we extracted the mean, variance, and covariance (data illustrated in table 2.1). Note this variance in power dissipation is due to the fact that we do not know which mix of benchmarks will be executed by the processor. We used the techniques described in section 2.5.3 to approximate the data with a Gaussian distribution. We created an ad-hoc floorplan of the processor with dimensions $2mm \times 2mm \times 0.5mm$. In the experiments, we set the thermal conductivity $k_s$ and the effective heat transfer coefficient $h$ to 148 $W/(m \cdot^\circ \mathrm{C})$ and 8700 $W/(m^2 \cdot^\circ \mathrm{C})$ respectively (consistent with the values used in [65]).

We then tried to address the problem of estimating the thermal profile of the chip, given the power density probabilistic characteristics and a few sensor observations. For a specific benchmark, we calculated the real power density map (through Wattch simulations) and therefore the real thermal profile which was used as a basis for comparison. Then we placed 5 sensors in modules like I-Cache, Instruction Window, ALU and *etc.* arbitrarily and noted the temperature at these points. Finally we used the method proposed in section 2.5.2 to estimate the thermal profile. We also used regression to fit a 2 dimensional second order polynomial onto the sensor observations. This regression-based thermal profile was used as a reference to compare the quality of our estimate.

Figures 2.5, 2.6 and 2.7 show the real thermal map, our estimated thermal map and the one generated using polynomial regression for the EON benchmark. We used only 5 sensors in our technique whereas the regression based technique needed 16 sensors. It is obvious that even with fewer sensors, the accuracy of our estimation methodology far exceeds that of the regression based approach.

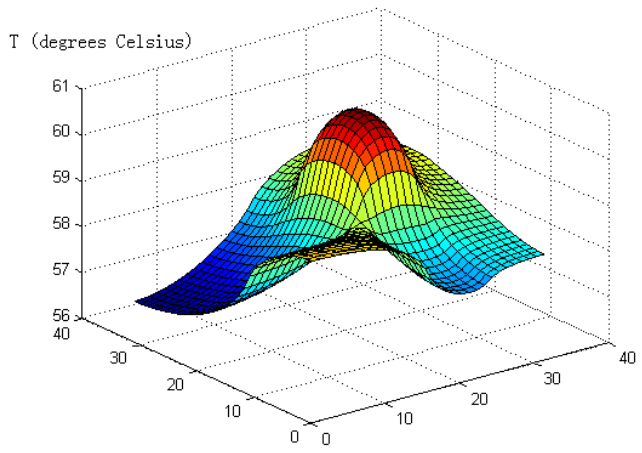



Figure 2.5: Real thermal profile

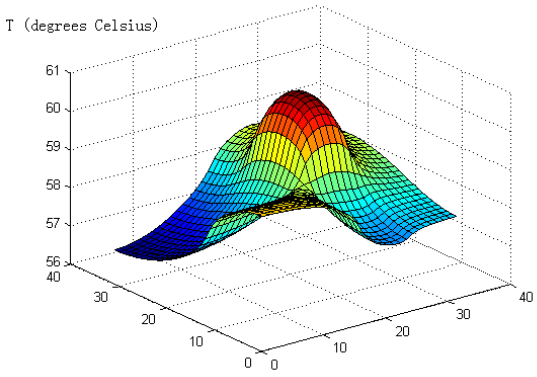



Figure 2.6: Estimated thermal profile

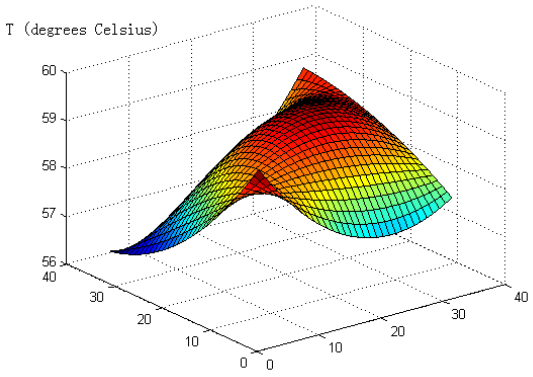



Figure 2.7: 2 dimensional polynomial fit

In figure 2.8, we highlight the relationship between RMS error of our approach and number of sensors utilized. As the number of sensors increases, the error decreases as expected. Figure 2.9 compares the RMS error obtained using our approach and using polynomial fit for various SPEC benchmarks. As can be seen, our approach has significantly smaller error with even fewer sensors. These results clearly highlight the effectiveness of our method.

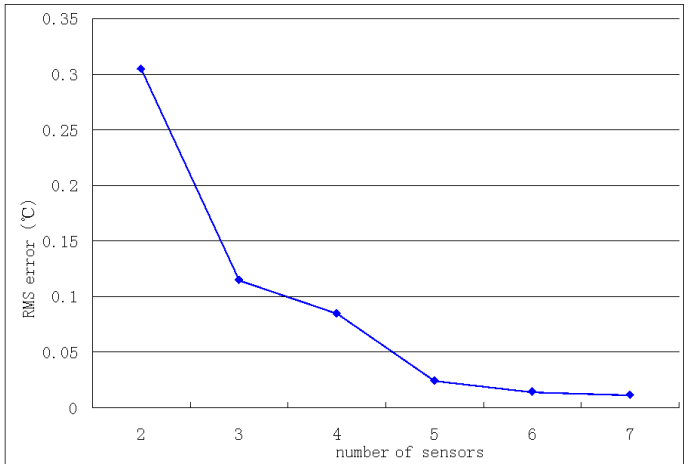



Figure 2.8: RMS error decreases as number of sensor increases

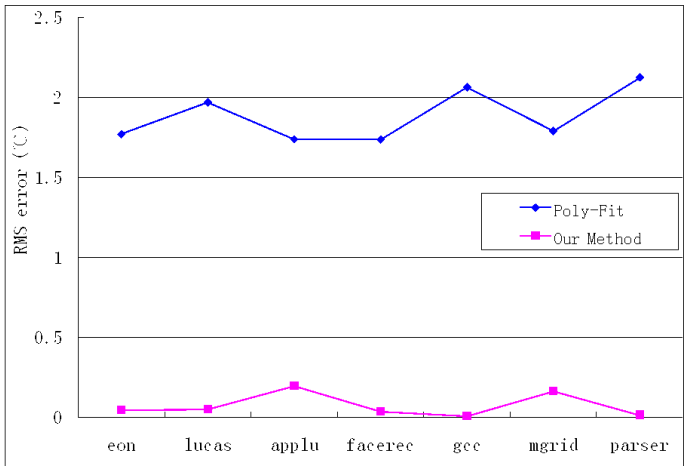



Figure 2.9: RMS error comparison for different benchmarks

### 2.6.1 Summary

In this chapter, we addressed the problem of estimating the chip-level thermal profile using only a few on-chip sensor observations. The underlying random nature of the thermal/power characteristics are well accounted for. We proposed two methodologies for solving this problem: 1) When the probability density function governing the power density variables exhibits jointly Gaussian distribution, we presented an optimal estimator for recreating the chip-level thermal profile. 2) When such a probabilistic property does not exist we presented a heuristic algorithm to approximate the optimal solution. The experimental results demonstrated significant

advantage (as much as 100x more accurate) of our method over a simple 2 dimensional $2^{nd}$ order polynomial regression strategy. Our approach for accurately estimating the chip-level thermal profile can be used to guide dynamic thermal/power management.

# Chapter 3

# Designing a Sensor-based Thermal Tracking Infrastructure

## 3.1 Motivation

In this chapter, we propose a statistical framework for designing a complete and accurate sensor-based on-chip thermal tracking infrastructure. Such a temperature tracking or monitoring system is very important because most of today's dynamic thermal management (DTM) techniques rely on it to make judicious control decisions. DTM technique usually works by throttling voltage or operating frequency in exchange for less power dissipation. Essentially they are trading off performance for lower operating temperature. If they are too aggressive or if they receive false alarms, chip performance could suffer. On the other hand, if such techniques are too conservative, thermal reliability is at jeopardy. Therefore, they must receive accurate and efficient feedback regarding a chip's thermal state at runtime to come up with the optimal thermal control decisions. For this purpose, several on-chip thermal sensor placement algorithms have been proposed in [35, 27, 33] to systematically deploy sensors across the chip. Although such work is promising, very few researchers have investigated the development of a complete framework that enables accurate and efficient thermal sensing and estimation. In this chapter, we will try to address this problem.

On chip thermal sensing infrastructure consists of several important design components which have strong interplays among each other [30, 71, 67]. These components include sensor placement, individual sensor design/compression and data fusion (as shown in figure 3.1). To see why these components are important and how they interact with each other, let us first take a look at the normal information flow in any sensor network infrastructure. First the sensors will collect information which reflects its local environment (temperature in our case). The data collected

by each sensor will undergo preliminary processing/compression (digitalization for example). When we have local resource constraints such as area and transmission overhead, data compression at each sensor is highly desirable. In addition, sensors (especially in on-chip environment) can be particularly susceptible to noise and fabrication variability. Therefore we must take these effects into consideration when we compress and use the sensor data. The compressed sensor readings will then be collected at a data fusion center to generate a complete thermal profile of the entire system. Within the fusion center, all sensor readings will be stored in a central register. A high-level coordinator (which could be implemented either in hardware or software) will take these sensor readings for information extraction purposes (eg. noise removal, signal filtering and estimation, etc.). The final outcome would be accurate knowledge of the thermal state of the entire chip.

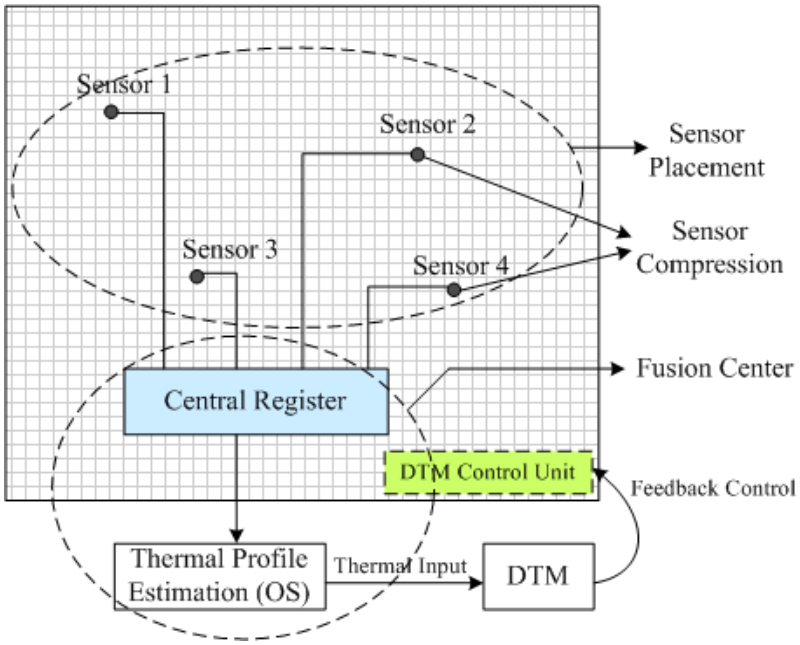



Figure 3.1: Thermal Sensing Infrastructure

Partly due to the nature of the thermal sensing problem and partly due to global area/power constraints, each of these individual design components are strongly inter-dependent. For example, the size of the central register, which depends upon the amount of area available, will impact the number of sensors we can

place and the degree of compression necessary at each sensor. The local area slack to fit individual sensors is another constraint. It may be easier to fit more sensors with high compression ratios than fewer sensors with no compression. The overall complexity of this sensor infrastructure can be determined as a tradeoff between the degree of accuracy desired and the implementation overhead.

In this thesis, we develop a unified statistical methodology for instantiating such a thermal sensing infrastructure: we decide sensor locations, degree of sensor compression and the design of fusion center in a unified way. Our methodology is capable of enforcing constraints such as the area available for each design component and also accounts for the intricate interplay between them. The specific contributions of this thesis are as follows:

1. **Sensor placement:** We develop a statistical methodology for deciding the sensor locations. Most existing works on this topic assign a *range* for each sensor [27, 35, 33, 31] and allocates sensors so as to *cover* all the potential hotspots. The metrics of *range* and *cover* are inaccurate for thermal sensors since sensor measures only the temperature of the location where it is placed (unlike cameras which have a field of view). Instead, our method uses the statistical correlation between sensor and hotspot temperature to predict the probability of capturing all the hotspots. This probability, which is a more sound metric than *range*, is used to drive sensor placement.

2. **Sensor design and compression:** Due to the potential lack of space to fit thermal sensors and the associated wiring, local compression of data is necessary. Compression of sensor data is also needed due to limited space for storing the sensor readings in the central register at the fusion center. Sensors are also prone to noise caused by fab-variability, supply voltage fluctuation etc. Our sensor design and compression methodology accounts for all the above considerations. To achieve our goals we use concepts from compression, signal estimation theory, optimization and VLSI design.

3. **Fusion center design:** Based on the thermal correlations, we develop statis-

tical techniques to accurately estimate the temperatures at all chip locations by exploiting the possibly compressed and noise-corrupted sensor observations.

4. **Exploiting the interdependency:** As discussed earlier, in order to obtain the best thermal sensing infrastructure solution which also has the minimal overhead (area, power etc), we will exploit the interplay between the above-mentioned aspects. For example, while deciding the sensor locations, our method will avoid areas where fitting sensors is difficult. Also, the finite number of bits available at the central register to store all sensor readings should be a limiting factor when deciding the total number of sensors allocated on chip and their compression rates. When designing the sensor placement schemes we introduce a feedback mechanism for incorporating the effects of compression. Essentially the compressed sensors will not provide thermal readings as accurately as it is assumed in the initial placement attempt and such effects must be accounted for. The sensor placement is thus an iterative designing process, refining the sensor locations in each iteration by understanding the compression effects from the later design stage.

To the best of our knowledge, such a complete and unified methodology for designing an on-chip thermal sensing infrastructure has not been investigated in the past. To demonstrate the effectiveness of our methods, we did experiments assuming the sensors are either noiseless (in ideal scenarios) or noisy (in realistic situations where fabrication variability can affect the sensor operation). Our results showed that having more sensors with compressed observations outperformed having fewer sensors with no compression when given the same space constraint at the global fusion center. On average our sensor placement and compression schemes can achieve about 35% reduction in the overall RMS error as compared to the range-based placement scheme and uniform compression (with about equivalent overhead). Our algorithms only took around 9 seconds in the worst case to develop the overall solution for placement, compression and data fusion. It is also noteworthy that our framework is general enough to incorporate different statistical models.

## 3.2 Sensor & Fusion Center Co-design

### 3.2.1 Fusion Center

A global fusion center collects and combines the sensor readings in order to estimate the chip's thermal state at any given time. It has two distinct components: central register and fusion algorithm. The central register is basically a register that holds all the thermal sensor readings (it could be a single or a combination of several actual registers). The fusion algorithm utilizes the combined sensor observations to estimate the complete thermal profile of the chip. The design of the entire thermal sensing infrastructure depends critically on how a few thermal sensor readings are used to predict the thermal profile. In this section we use a variant of the statistical approach presented in chapter 2. This approach is more straightforward and it combines the information provided by a few on-chip thermal sensors with the thermal statistical information (such as the thermal correlations among different chip locations) to generate accurate temperature estimates at all chip locations. The idea is that by exploiting such thermal correlations between different chip modules, the temperature sampled at the sensor locations can be used to predict the thermal state at other chip locations as well. We will describe this fusion algorithm in more detail below.

An integrated system on chip consists of multiple functional modules that all dissipate power (ALU, branch predictor, instruction window, cache and etc.). When a certain application is running on the chip, heat is generated in different rates at various modules and each of these modules will have an associated power density. In the ideal case if we can find out the power density profile of the chip accurately at runtime then we can use analytical methods to calculate the thermal profile without the need for sensors [65, 53]. However in reality the unpredictable workloads and the variability in transistor and interconnect parameters cause randomness in the chip thermal behavior. Thus the real thermal profile at runtime is highly random. If we divide the entire chip into $N \times N$ grids (see figure 3.2) and approximate the temperature within each grid cell as uniform (note the grids can be arbitrarily

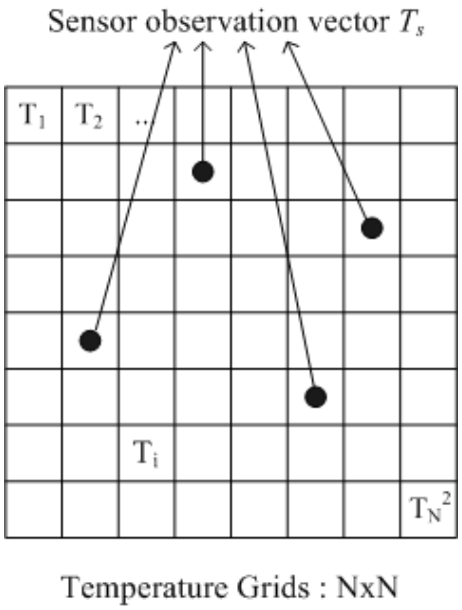



Figure 3.2: A simplified chip from a thermal perspective

small), the entire thermal profile of the chip can be modeled as a random vector $\vec{T}$ with dimension $N^2 \times 1$. Each element $T_i$ of this vector represent the temperature in the $i$-th grid cell and is a random variable with an associated probability density function (PDF). Note that since the thermal behavior of a chip is affected by many independent sources of randomness (such as unpredictable workloads, supply voltage fluctuation and various fabrication induced circuit parameters variability), their collective effects are reasonably close to a Gaussian distribution (see Central Limit Theory). Based on this observation we can model $\vec{T}$ as a Gaussian vector with mean $\vec{\mu}_T$ and covariance matrix $\Sigma_{TT}$. The advantage of a Gaussian model is that it is reasonably close to reality in most cases and we can obtain the analytical form of the optimal estimator for $\vec{T}$ (complete thermal profile) given $\vec{T}_s$ (the sensor observations). This optimal estimator is simply the conditional expectation $E(\vec{T}|\vec{T}_s)$ (see equation (3.1)). The interested readers can refer to chapter IV.B of [43] for proof. In practice, even if the actual temperature distribution is not strictly Gaussian, it will be reasonably close to Gaussian as long as there are many independent tasks running on the chip, which is the case for most of today's general-purpose processors. Thus equation (3.1) can be expected to generate accurate thermal estimates

in practice.

$$E(\vec{T}|\vec{T_s}) = \vec{\mu}_T + \Sigma_{TS}\Sigma_{SS}^{-1}(\vec{T_s} - \vec{\mu}_S) \tag{3.1}$$

$$\hat{\Sigma} = \Sigma_{TT} - \Sigma_{TS}\Sigma_{SS}^{-1}\Sigma_{ST} \tag{3.2}$$

Here the thermal sensors are assumed to be placed in some of the grid cells (say, subset $S$). Therefore, at runtime, we can observe the temperatures at these grids ($\vec{T_s}$) by simply sampling the sensors. $\Sigma_{SS}$ represents the covariance matrix for the sensor grids and is a submatrix of $\Sigma_{TT}$ where each row/column in $\Sigma_{SS}$ corresponds to a grid where a sensor has been placed. By the same logic, $\Sigma_{TS}$ is the covariance between all grids (represented by set $T$) and sensor grids (set $S$). Also, $\vec{\mu}_S$ is the mean temperatures of sensor grids $S$. Note that thermal correlations are exploited in this approach to generate thermal estimates. Such correlations are reflected in the covariance matrix $\Sigma_{TT}$ and $\Sigma_{TS}$ and they exist due to physical proximity and also similar power behaviors of different functional units. Equation (3.1) takes the deviation of our sensor observations from their average values ($\vec{T_s} - \vec{\mu}_S$) and maps this difference to other chip locations based on the thermal correlation. Equation (3.2) shows the conditional covariance ($\hat{\Sigma}$) of $\vec{T}$ given sensor readings $\vec{T_s}$. This essentially captures the new reduced uncertainty associated with our thermal estimates, now that we know the readings $\vec{T_s}$ from sensors. The diagonal elements of $\hat{\Sigma}$ give us the reduced variance at the corresponding grid cells. The trace (sum of the diagonal elements of a matrix) of $\hat{\Sigma}$ is thus the total variance for the entire chip which is an indicator of how good our thermal estimation is. From equation (3.2) we can see this trace depends on sensor locations, i.e. the selection of set $S$ and hence $\Sigma_{TS}/\Sigma_{SS}$ (we discuss more on sensor placement in subsequent sections). It is noteworthy that this statistical approach is more sound than the existing range-based approaches [35, 27, 33]: instead of assuming a range for each sensor and ignoring the thermal gradient within such range (as well as discarding the information outside the range), it calculates the conditional expectation of the temperatures at all grid locations. It also gives the variance associated with such estimates which reflects our confidence in our estimated temperatures. From the expectation and variance we can easily deduce the possibility of capturing the potential hotspots. The above presented

fusion algorithm works by first combining all sensor readings into a central register at the fusion center and then computing the thermal estimates based on the value in this register.

**Overall error:** Note that $trace(\hat{\Sigma})$ is not the only error that needs to be considered in our framework. Sensor compression also contributes to the overall error. In the formulation below we will try to summarize the overall optimization problem posed by our framework in simple mathematical terms. The details and further explanations will be given in the subsequent sections.

$$Error_{overall} = \sum_i E\Big((T_i^e - T_i^r)^2\Big) \tag{3.3}$$

$$= Error_{est} + Error_{com} \tag{3.4}$$

$$s.t. \quad \begin{cases} |S| \leq n \\ \sum_{i=1}^{n} s_i \leq M \end{cases}$$

In equation (3.4), $T_i^e$ and $T_i^r$ are the estimated temperature and the real temperature at grid $i$ respectively. $n$ is the total number of sensors allowed to be placed on chip. $M$ is the size constraint at the central register (the total number of storage bits available). $s_i$ is the number of bits allocated to sensor $i$.

The first component of the overall error $Error_{est}$ is the error associated with the estimation. It will be affected by the estimator we choose, the constraint $n$ on the total number of sensors and the actual sensor placement scheme. For the estimator we just introduced, this error is simply $trace(\hat{\Sigma})$ where $\hat{\Sigma}$ is given in equation (3.2) (assuming we have the correct value of $\Sigma_{TT}$). Note that the sensor placement scheme will affect our selection of subset "$S$", which in turn affects $\Sigma_{SS}$ and $\Sigma_{TS}$ and therefore the estimation error. In the ideal case if sensors are placed at all locations of the chip, we have $\Sigma_{TS} = \Sigma_{SS} = \Sigma_{TT}$ which leads to equation (3.2) evaluating to zero (no estimation error if we have sensors everywhere).

The second error component is the compression induced error $Error_{com}$ which captures how much impact the compression has on the final estimated thermal map. This error is given in equation (3.13) and will be discussed in more detail in section

3.2.3. It is a strong function of $\{s_i\}$ (the bit allocation scheme). The overall error term shown in equation (3.4) guides the design of our entire sensing infrastructure.

### 3.2.2 Noisy Thermal Sensor

In this section, we focus on thermal sensor design and compression. To make things more concrete, we use ring oscillator based thermal sensor as an example to describe our method. Note that our methodology is general and can be applied to any types of sensor equally well. A ring oscillator (RO) simply consists of an odd number of inverters. The change in temperature will affect the delay of each inverter and hence change the frequency of the RO. We can have a counter at the output of the RO to count the number of state flips within a fixed period of time $t_p$ (see figure 3.3). The output of this counter at the end of the counting period reflects the frequency of the RO. Due to the fact that the frequency of an RO has a close-to-linear relationship with its local temperature, ROs are often used to implement thermal sensors. The number of bits needed to represent the counter output captures the precision of the sensor.

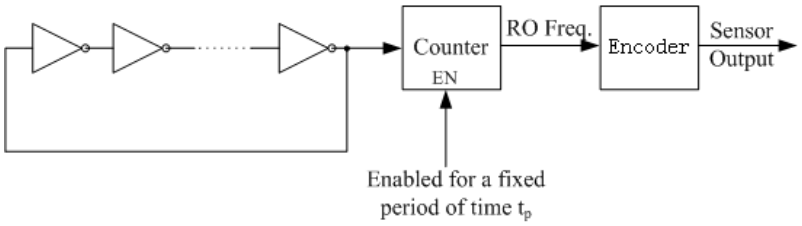



Figure 3.3: Ring oscillator as a thermal sensor

In the ideal case where a sensor is noiseless, it gives the same sensor reading for the same grid temperature irrespective of its location and time of sampling. In such a case the sensor readings present no ambiguity whatsoever and could be relied upon completely. Since the number of bits $b$ of the counter output is limited, the sensor cannot provide infinite precision. If we uniformly divide the potential temperature range $H_{total}$ of a sensor into $n = 2^b$ sub-ranges $\{H_1, H_2, \ldots, H_n\}$, then the sensor can report the specific sub-range $H_i$ that the sensor grid is experiencing. The finite

size of the counter output imposes a small quantization error.

In reality, thermal sensors are highly susceptible to fabrication variability, supply voltage fluctuations and *etc.* [53, 68, 69]. To understand the effect of such sensor noise, we experimented with Monte Carlo simulation and graphed the randomness we observed in RO frequency (figure 3.4) caused due to various noise factors. We assumed 5% variation in threshold voltage, channel length/width, oxide thickness and supply voltage. As shown, for each underlying actual temperature, the sensor frequency is a random quantity and can take a range of values. This example illustrates the worst-case spread of noisy sensor readings because some of the variation can be eliminated by calibration. For example, the process parameter variations do not change after manufacture and can be compensated with post-manufacture calibration. However, noise due to voltage fluctuations and cross-coupling will persist and can be treated with our technique. Additionally, post-manufacture calibration may be expensive. Our noise reduction technique can reduce the required sensor accuracy and thus the cost of calibration.

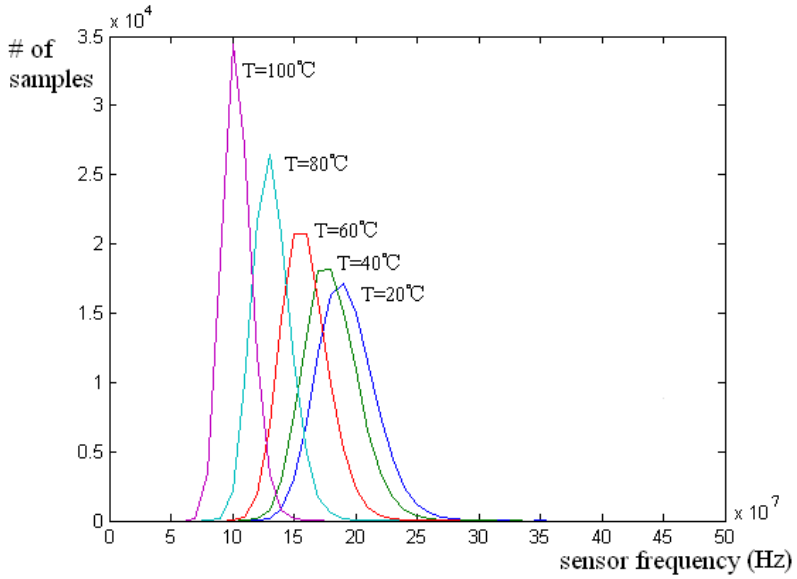



Figure 3.4: Simulated RO frequency distributions for different underlying temperatures ranging from 20 °C to 100 °C with 20 °C increments ($10^5$ samples for each curve).

Now, for a given sensor reading, the actual temperature that caused the reading cannot be decided deterministically because of sensor noise. In order to estimate the temperature for a given sensor reading, we formulate the problem in a hypothesis testing framework [43]. Hypothesis testing has the advantage that it can generate accurate thermal estimates for any type of noise distribution. It can also be easily extended to handle the sensor compression and bit allocation problems as well (which we will discuss in later sections).

As mentioned earlier, we can divide the thermal range $H_{total}$ into $n = 2^b$ sub-ranges $\{H_1, H_2, \ldots, H_n\}$. We are interested in estimating which sub-range the sensor temperature falls within. We can assume each $H_i$ is a hypothesis for the temperature of this sensor with an associated prior probability ($P_i$ for hypothesis $H_i$). Prior probability captures the likelihood of having sub-range $H_i$ as the sensor temperature before any observation is made. Such prior probability can be obtained by simulating benchmarks or typical chip workloads (for details please see the result section). As explained earlier, for a given thermal hypothesis $H_i$ at the sensor grid, the reading from this sensor is random due to sensor noise. The randomness in sensor observation for each underlying hypothesis can be modeled as a PDF as illustrated in figure 3.4. This modeling could be obtained using various statistical schemes that characterize the behaviors of the thermal sensor under different thermal conditions. Given a sensor observation $T_o$, our goal is to choose one of the hypotheses as our best prediction for the actual sensor temperature such that the expected prediction error is minimized.

**Formal problem formulation:** Let us define $P_i$ as the prior probability of hypothesis $H_i$. Note that $\sum_i P_i = 1$ since one of these $n$ hypothesis must be true. Now, given the set of all observations $\Omega$ (which for a thermal sensor with $b$ output bits would contain $2^b$ values), we would like to partition $\Omega$ into $n$ (number of hypotheses) subsets $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}$ such that each subset corresponds to a specific hypothesis prediction (i.e. if the observation falls within a certain subset it means the corresponding hypothesis must be true). The challenge is to come up with this

partition/decision rule $\delta$ such that the expected prediction error is minimized.

$$H_p = \delta(T_o) = \begin{cases} H_1 & T_o \in \Gamma_1 \\ H_2 & T_o \in \Gamma_2 \\ \cdots & \\ H_n & T_o \in \Gamma_n \end{cases} \tag{3.5}$$

Here $T_o$ is the sensor observation. $H_p$ is the predicted hypothesis. The prediction error is simply the absolute difference between the real and the predicted hypothesis $|H_p - H_{real}|$. The expected penalty is defined as follows.

$$E(|H_p - H_{real}|\big|T_o) \tag{3.6}$$

$$= \sum_{i=1}^{n} |H_p - H_i| \times prob(H_{real} = H_i|T_o) \tag{3.7}$$

where $H_{real}$, $T_o$ and $H_p$ represent the real sensor hypothesis (basically real temperature), the noisy sensor observation and our prediction respectively. Since sensors are noisy, several different hypotheses could have caused the same observation $T_o$. This fact is captured by the probability $prob(H_{real} = H_i|T_o)$ which could be computed as follows:

$$\begin{aligned} & prob(H_{real} = H_i|T_o) \\ =& \frac{prob(T_o|H_{real} = H_i) \times P_i}{prob(T_o)} && \text{(Bayes-Rule)} \\ =& \frac{prob(T_o|H_{real} = H_i) \times P_i}{\sum_{j=1}^{n} prob(T_o|H_{real} = H_j) \times P_j} && (3.8) \end{aligned}$$

Here $prob(T_o|H_{real} = H_i)$ is simply the probability of obtaining a specific sensor reading $T_o$ for a given hypothesis $H_i$. It could be computed a-priori using statistical schemes (for example, as we illustrated in figure 3.4). Based on equation (3.8) we can easily compute (3.7). Now, the overall expected cost associated with the decision rule can be defined as follows:

$$E(|H_p - H_{real}|) = \sum_{\forall T_o \in \Omega} E(|H_p - H_{real}|\big|T_o) \cdot prob(T_o) \tag{3.9}$$

The optimal overall expected cost can be achieved by minimizing $E(|H_p - H_{real}|\big|T_o)$ for each $T_o$ separately since $prob(T_o)$ is simply a constant for each fixed

$T_o$. Thus, we can generate our optimal decision rule for minimizing the overall cost in equation (3.9) as follows:

1. For each $T_o \in \Omega$, select the hypothesis that minimizes equation (3.6). This could be done using a simple $O(n)$ complexity search through all possible hypotheses ($\mathbb{H}$ represents the set of all hypotheses).

$$\delta(T_o) = \underset{H_p \in \mathbb{H}}{\operatorname{argmin}} \; E(|H_p - H_{real}| \big| T_o) \tag{3.10}$$

2. Store the optimal prediction for each observation ($T_o \rightarrow H_p$) in a look up table.

The above decision rule can be pre-determined quite easily for all possible sensor observations given the required statistical information. Thus our decision rule can be easily implemented in hardware as an encoder stage of the sensor as shown in figure 3.3. Essentially this encoder stage is responsible for translating the noisy sensor readings into our predictions.

When we have multiple sensors, the observations from them can be treated as a vector and the real temperature predictions for this set of sensors can be made in a vector form as well. The same hypothesis testing methodology applies with the only difference being that we have a larger solution space since we are handling vectors instead of individual sensor observations/predictions. By putting multiple sensors observations together, any correlation information among these sensors can be exploited by our hypothesis testing method to make thermal predictions.

### 3.2.3 Fusion Center & Sensor Co-Design

Now, we extend our hypothesis testing based approach for co-design of the fusion center and the sensors.

#### 3.2.3.1 Problem Formulation

For clarity of exposition let us first assume that sensors are noiseless (the noisy sensor case is similar and will be discussed later). The fusion center design problem

essentially boils down to how to allocate the total number of bits in the central register to all sensors. A larger central register implies more bits for sensors which in turn implies more precise sensor readings and more accurate overall estimates (assuming a fixed sensor placement). However more bits in the central register would also imply more area overhead for storing these bits and also communicating more data from sensors thereby complicating the wiring. Note that the exact implementation of the central register (one big or many small actual registers) is not of concern here. We are just trying to address the finite total space for storing sensor data. In this section, we assume that the number of sensors and their placement is fixed so we focus primarily on the central register size.

For a given size of $M$ bits and a given placement scheme (as determined by our placement algorithm discussed in section 3.3), we can have several policies for distributing them among the sensors. A uniform policy would imply the same level of precision for each sensor. Interestingly, some sensors are more informative than others (ones that have a higher correlation with hotspot locations) and therefore should be given more precision (number of bits). Also, some sensors, if given more bits, may not have the space available for routing the extra wires to the central register. These considerations need to be accounted for while distributing the total $M$ bits. The choice of $M$ itself is more complex since it depends critically on how much area/power penalty we can tolerate, how much accuracy is desirable and sensor locations as well. In general $M$ can be specified by designers based on various design goals.

Let us suppose each sensor $i$ can have at most $b_i$ bits. Here $b_i$ depends on the space available for the sensor and its wiring. Based on the informativeness of the sensors and their space constraints (parameter $b_i$), we would like to distribute the total $M$ bits to sensors such that the thermal estimates produced by our framework (see equation (3.1)) are as accurate as possible. Now letting $\vec{T}_s^a$ and $\vec{T}_s^c$ represent the accurate (without $M$ constraint) and the compressed (due to a finite $M$) sensor observations respectively, the thermal estimates obtained for these two situations are $E(\vec{T}|\vec{T}_s^a)$ and $E(\vec{T}|\vec{T}_s^c)$ respectively (see equation 3.1). Note that $\vec{T}$ is a vector

that consists of the thermal estimates at all grid locations. The error in the $i$-th grid caused by compression is simply $|E(T_i|\vec{T}_s^c) - E(T_i|\vec{T}_s^a)|$. Our goal is to allocate the total $M$ bits to sensors such that the total error $\sum_i |E(T_i|\vec{T}_s^c) - E(T_i|\vec{T}_s^a)|$ is minimized. If we use $s_i$ to represent the allocated number of bits for sensor $i$, then the error is a function of $(s_1, s_2, \ldots, s_n)$ which is our bits allocation scheme. Thus our co-design problem can be formulated as follows:

$$\text{minimize} \qquad E\Big(error(s_1, s_2, \ldots, s_n)\Big) \tag{3.11}$$

$$s.t. \qquad \begin{cases} m_i \leq s_i \leq b_i \\ \sum_{i=1}^{n} s_i \leq M \end{cases} \tag{3.12}$$

Here $m_i$ is the minimum number of bits any sensor must be given (generally set to 0), and once again $s_i$ is the number of bits assigned to sensor $i$ and $b_i$ is the maximum number of bits for a sensor. Note that *error* itself is a random variable since the underlying observations are random. Therefore the expected error $E(error)$ should be used as the cost function for our optimization problem. If $M$ was not a limiting constraint and all sensors worked at their perfect precision (i.e. $s_i = b_i$ for $\forall i$), then the expected error should be zero. Otherwise the error creeps in due to compression of the sensor observations. For a fixed sensor placement, this error depends explicitly on how the total $M$ bits is allocated among sensors and implicitly on how each sensor is compressed locally. Hence allocating the total $M$ bits and designing individual sensors (so that the allocated bits are used most effectively) must be done hand in hand. Now let us focus on the error term below:

$$\begin{aligned} & error(s_1, s_2, \ldots, s_n) \\ &= \sum_{\forall grids:i} (|E(T_i|\vec{T}_s^c) - E(T_i|\vec{T}_s^a)|) && (3.13) \\ &= \sum_{\forall rows} (|\Sigma_{TS}\Sigma_{SS}^{-1}(\vec{T}_s^c - \vec{T}_s^a)|) && (3.14) \\ &= \sum_{\forall rows} (|\Sigma_{TS}\Sigma_{SS}^{-1}\Delta\vec{T}_s|) && (3.15) \end{aligned}$$

Here (3.14) is obtained by substituting equation (3.1) into (3.13). $\Delta\vec{T}_s = \vec{T}_s^c - \vec{T}_s^a$ is a vector representing the difference between the compressed sensor observations

and the uncompressed ones. The product of $\Sigma_{TS}\Sigma_{SS}^{-1}$ is a coefficient matrix with dimension $N^2 \times n$ which represents the sensitivity of the error on sensor compression (here $N$ is the grid dimension of the chip and $n$ is the number of sensors). If we assume the element at the $d$-th row and the $k$-th column of this coefficient matrix is represented by $a_{d,k}$, then the above error function can be written in its scalar form:

$$error = \sum_{d=1}^{N^2} |\sum_{k=1}^{n} a_{d,k}\Delta T_k| \tag{3.16}$$

where $\Delta T_k$ is the $k$-th element of the $\Delta\vec{T_s}$ vector and represent the compression error at sensor $k$. Note that some grids may be more important than others. Therefore, we can assign different weights to different chip grids to reflect this design consideration. Thus the error can be redefined as:

$$error = \sum_{d=1}^{N^2} w_d|\sum_{k=1}^{n} a_{d,k}\Delta T_k| \tag{3.17}$$

By introducing this weight coefficient $w_d$ we can account for the fact that different chip regions may have different thermal constraints and hence should be given different importance in the error function. Now, the overall expected error mentioned in our problem formulation (equation (3.11)) can be defined as follows:

$$\begin{aligned}
cost &= E(error) && (3.18)\\
&= E(\sum_{d=1}^{N^2} w_d|\sum_{k=1}^{n} a_{d,k}\Delta T_k|) && (3.19)\\
&\leq E(\sum_{d=1}^{N^2} w_d \sum_{k=1}^{n} |a_{d,k}\Delta T_k|) && (3.20)\\
&= E(\sum_{d=1}^{N^2} w_d \sum_{k=1}^{n} |a_{d,k}||\Delta T_k|) && (3.21)\\
&= \sum_{d=1}^{N^2} w_d \sum_{k=1}^{n} |a_{d,k}|E(|T_k^c - T_k^a|) && (3.22)\\
&= \sum_{k=1}^{n} g_k E(|T_k^c - T_k^a|) && (3.23)
\end{aligned}$$

Here in equation (3.23), $g_k = \sum_{d=1}^{N^2} w_d|a_{d,k}|$ can be viewed as the sensitivity of the overall expected error on the local compression error at sensor $k$.

Now let us focus on the term $E(|T_k^c - T_k^a|)$. This is the local expected error at the sensor $k$ caused by compression. Note that this error depends explicitly on how the total $M$ bits are allocated among sensors and implicitly on how each sensor is actually compressed. Let us address the second problem first since the first one depends on the second one.

**Sensor Compression:** In principle and form, the expected error $E(|T_k^c - T_k^a|)$ can be defined in a hypothesis testing framework similar to the one we described in section 3.2.2 (see equation (3.9)).

$$E(|T_k^c - T_k^a|) = \sum_{\forall T_o \in \Omega} E(|T_k^c - T_k^a| \big| T_o) \cdot prob(T_o) \tag{3.24}$$

Here subscript $k$ represents the $k$-th sensor. Initially this sensor has $b_k$ bits (its local maximum). Let us define a $(2^{b_k} \times n)$ matrix $Q$ where $n$ is the number of hypotheses (grid temperatures) that the sensor can experience. Each row of this matrix corresponds to a specific observation. The element $Q[i, j]$ stores the joint probability of getting observation $i$ when hypothesis $j$ is true. When the sensor is noiseless, there can only be exactly one hypothesis with non-zero probability for each observation. All other hypotheses have zero probability (note that this can only happen when $2^{b_k} = n$). Now we compress this sensor by, say, 1 bit. This means that the sensor will lose some fidelity and will have to be compressed from a larger set of fine-grained observations to a smaller set of coarse-grained observations. This implies some rows (observations) in $Q$ should be combined. Let us suppose that we combine row $i$ and row $l$ into one row. Now the probability of getting this new combined observation under hypothesis $j$ is simply $Q[i, j] + Q[l, j]$. Thus the compression of sensor observations amounts to combining and adding the corresponding rows. The objective of the compression scheme is to decide which rows to combine and which hypotheses to predict for the new set of compressed observations such that the cost function (3.24) is minimized. Now, let $Q^{new}$ be the new matrix after compression which has $2^{s_k}$ rows. Note that in our case the cost of predicting hypothesis $i$ while hypothesis $j$ is true is simply $|H_i - H_j|$. Given the compressed matrix $Q^{new}$, we would like to generate a decision rule that

translates each observation into a predicted hypothesis such that the expected cost is minimized. Just like in the noisy sensor case (section 3.2.2), the same sensor reading can be observed under multiple hypotheses. The best decision rule that achieves the minimum cost can be generated using the techniques discussed earlier (basically equation (3.10)). The cost of this optimal decision rule is given by equation (3.24) and depends on how we compress the sensor observations. This cost can be calculated in the same way as we calculate equation (3.9) based on (3.6) – (3.8). Note that the underlying probabilities needed by these equations can be obtained by calculating the matrix $Q^{new}$. The compression at each sensor will generate a $Q^{new}$ from $Q$ by combining $2^{b_k}$ rows into $2^{s_k}$ rows such that (3.24) is minimized. The compression policies generated could be implemented in the encoder stage of the sensor (see figure 3.3). Given an observation that has $b_k$ bits, it outputs a new code of length $s_k$ bits where $s_k$ is the number of bits allocated to the sensor. Also, several observations that have been compressed into one are given the same output code of length $s_k$ bits. The details of the algorithm for generating this $Q^{new}$ are described in Algorithm 1.

**Bit Allocation:** As explained above, when given a fixed number of bits for a certain sensor $k$, we can come up with a compression scheme as well as a prediction policy for any observation so that the expected local misprediction error at this sensor ($E(|T_k^c - T_k^a|)$) is minimized. Now we present our scheme for allocating the total $M$ bits among sensors such that the overall expected error for the entire chip (equation (3.18)) is minimized. This part is relatively easy since in our problem formulation we have already established the relationship between a sensor's local compression error and the overall expected error as highlighted in equation (3.23). We can start by assuming the maximum bits $s_i = b_i$ for each sensor $i$ and gradually reduce them until the constraint in (3.12) is met. This can be done in a greedy fashion by reducing one bit at a time at the sensor which results in the least cost increase in each iteration (compression causes increase in cost). Based on the way in which we construct the overall cost function (3.23), our greedy bit allocation scheme will allocate fewer bits to the less informative sensors while giving more precision to

the more important ones.

#### 3.2.3.2 Co-Design Algorithm

Now we present an algorithm for simultaneous fusion center and sensor co-design. Our technique estimates the overall error for any bit allocation scheme by generating the best compression scheme at each sensor and then evaluating equation (3.23). The results are then used to drive the bit allocation process. The final outcome includes both a bit allocation scheme and a compression policy at each sensor. The algorithm is shown below:

**Algorithm 1** Sensor & Fusion Center Co-Design:

**Require:** Initially $s_i \leftarrow b_i$ for each sensor $i$

**Ensure:** Bit allocation and sensor compression schemes

1: **while** $\sum_{i=1}^{n} s_i \geq M$ **do**
2: Choose sensor $k$ with the least sensitivity $g_k$ to the overall cost (see equation (3.23))
3: If $s_k == m_k$ then choose the next sensor
{// Compress this sensor by one bit in the following}
4: Let $Q$ be the matrix for sensor $k$ such that $Q[i, j]$ is the probability of getting observation $i$ with hypothesis $j$
5: Generate all possible $Q^{new}$ (by combining the rows of $Q$) such that the number of rows in $Q^{new}$ is 1/2 of $Q$ (one bit reduction)
6: For each $Q^{new}$ select the optimal decision rule using equation (3.10) and use the associated cost to determine the best $Q^{new} = Q^{new}_{best}$
7: $Q \leftarrow Q^{new}_{best}$; $s_k \leftarrow s_k - 1$; update the overall error (equation (3.23))
8: **end while**

We start with all sensors allocated the maximum bits. Then in each iteration, we select a sensor which has the least sensitivity to the overall expected error and compress it by 1 bit. We repeat this process until we reach a feasible solution that satisfy the global "$M$" constraint.

#### 3.2.3.3 Noisy Case

When the sensors are noisy, for a given observation there could be several hypothesis possible. Hence, in the uncompressed case, the $Q$ matrix is such that for each observation $i$, the row $Q[i]$ can have several hypothesis with non-zero probabilities. Fundamentally, algorithm 1 can simply take this new $Q$ matrix and perform the compression in a similar fashion as the noiseless case. Hence the same algorithm can be applied to the noisy case as long as the $Q$ matrix is updated appropriately.

## 3.3 Sensor Placement

In this section we describe a thermal sensor placement approach that is very different from most existing placement algorithms [27, 33, 35]. As explained earlier most existing placement algorithms assume that each sensor has a "*coverage region*" around it and the temperature within this region can be accurately monitored; yet a sensor have no or very little knowledge of the temperature outside of this region. Thus the goal is to place as few on-chip sensors as possible so that all pre-identifiepd hotspots fall within the coverage region of a certain sensor. In reality thermal sensors only measure the temperatures of the grids that they are located in. Defining a range on them, therefore, becomes problematic: on one hand the thermal gradient within the sensor range is ignored. On the other hand the information about locations outside the sensor *coverage region* is discarded. Thus the accuracy of the range-based methods highly depends on how large or small this sensor *range* is. Granted, if the range is chosen to be small it can achieve reasonable accuracy. However such accuracy comes with a cost: more sensors need to be placed on chip due to the smaller range each sensor can cover. Our method does not have such drawbacks, it exploits the thermal correlation to place sensors so that the sensors not only provide thermal information for its local area but also provides information for remote locations as long as there exists certain amount of correlation. Thus our methods can provide better thermal sensing accuracy with even fewer sensors. In this thesis we use statistical techniques (equation (3.1)) to estimate the entire

thermal profile from limited sensor observations. The fundamental error associated with a placement scheme is given by the variance shown in equation (3.2). The overall error also depends on the degree of compression imposed on the sensors. As discussed earlier, the more spatially constrained a sensor is, the more we will have to compress its readings since we do not have ample space available for routing data wires. Generally sensors that have high compression factors could be undesirable since either the sensors are less informative (leading to compression in favor of others) or do not have sufficient space. Hence sensor placement is a complex design problem which needs to consider not only how much information a potential sensor location can provide but also the available space slack at that location. It also has to account for the finite size $M$ of the central register.

### 3.3.1 Problem Formulation

The purpose of the sensor placement algorithm is to choose optimal locations for a limited number of sensors such that the entire thermal profile (or certain critical regions of interests) can be estimated as accurate as possible. Let $S$ be a subset of the grids with size $m$ representing the sensor locations. As mentioned earlier, there are two kinds of errors associated with a sensor placement: the fundamental error given by equation (3.2) and the compression error. A solution with low fundamental error might have high compression error. For example, the sensor locations chosen might not have sufficient space to fit the routing wires. Given a sensor count, $m$, the sensor placement problem is to find the optimal locations for these sensors such that the overall error is minimized. Note that since the compression error accounts for the area constraint at sensor locations and the size constraint of the central register, minimizing the compression error implies accounting for both such space limitations.

For simplicity in exposition, let us ignore the compression error for the moment and focus primarily on the fundamental error. Finding a sensor placement that

minimizes this error can be formulated as follows:

$$\text{choose } S \subset T \text{ with } |S| = m \tag{3.25}$$

$$\text{such that trace}(\hat{\Sigma}) \text{ is minimized} \tag{3.26}$$

Here $\hat{\Sigma}$ is given by equation (3.2) and represents the conditional covariance associated with our thermal estimates $E(\vec{T}|\vec{T}_s)$. Thus the $i$-th diagonal element of $\hat{\Sigma}$ is the potential variance of the estimated temperature at the $i$-th grid cell. The trace of $\hat{\Sigma}$ is the sum of all diagonal elements and hence is a good indicator for the fundamental error since it reflects the total variance in the estimated thermal profile. Our goal is to choose a subset $S$ of size $m$ (from all grid locations $T$) as our sensor locations such that trace($\hat{\Sigma}$) is minimized. As can be seen from equation (3.2), the value of this selected cost function only depends on the sensor locations $S$ through terms $\Sigma_{TS}$ and $\Sigma_{SS}$. Optimizing this cost function is a very complex task and the problem is in general NP-hard. In the following we will present two heuristic algorithms for generating sensor placement schemes.

### 3.3.2 Sensor Placement Algorithms

Our first placement algorithm works by directly minimizing the cost function (see Algorithm 2). It chooses one sensor at a time by trying all potential sensor locations and then selecting the one with the minimum cost calculated using equation (3.2). This algorithm has relatively higher computational cost (compared to our second algorithm to be explained next) since it has to calculate equation (3.2) in each iteration and for every candidate location. On the other hand it has very good solution quality since it directly minimize the expected variance of the estimated thermal profile.

Our second algorithm tries to minimize trace($\hat{\Sigma}$) indirectly (see Algorithm 3). It uses the thermal correlation as the guidance when choosing sensor locations. Intuitively, we would like to select those sensor locations that provide the maximum information about places where sensors do not exist. Also, in order to minimize redundancy, the sensors themselves should have little information about each other.

**Algorithm 2** Placement 1

**Require:** The desired number of sensors $m$, all grid location set $T$ and the covariance matrix $\Sigma_{TT}$

**Ensure:** Sensor location set $S$ with $|S| = m$

1: $S \leftarrow \emptyset$
2: **while** $|S| < m$ **do**
3:  **for all** $i \in T \setminus S$ **do**
4:  $S_{new} \leftarrow S \cup i$
5:  Calculate the new $\hat{\Sigma}$ using $S_{new}$ (equation (3.2))
6:  **end for**
7:  Select $i_{min}$ so that it results in the minimum trace($\hat{\Sigma}$) among all $i$
8:  $S \leftarrow S \cup i_{min}$
9: **end while**

The degree of mutual information between two arbitrary grid locations can be captured by their thermal correlation: if a sensor grid is highly correlated with other non-sensor grids then in general it is a good candidate. If a sensor location has high correlation with other sensors yet lower correlation with non-sensor locations, then it should not be included. Following this intuition we can simplify the cost function as follows:

$$\sum_{\forall grids:i} \max\left(0, \quad 1 - \sum_{\forall sensors:j} c_{ij}^2\right) \tag{3.27}$$

Here $c_{ij}$ is the thermal correlation between grid $i$ and $j$. For each grid location $i$, the expression $\max(0, 1 - \sum_{\forall sensors:j} c_{ij}^2)$ captures the uncertainty of grid $i$ given the set of sensors. If a lot of sensors have high correlation with $i$, then this expression should evaluate to almost 0. Note that if $i$ is a sensor location itself, then this expression would always be equal to 0 since $c_{ii} = 1$. Choosing a set of sensors that maximizes the correlation between sensors and non-sensors will be encouraged by this cost function. On the other hand, due to the inherent max function, it will avoid picking too many sensors that have high correlation with the same grid (this could indicate redundancy). This cost function will also discourage choice of sensors which have high correlation among each other (which is wasteful). This is because the cost

of a sensor grid $i$ is 0 regardless of the presence of other sensors. This placement algorithm will not put more sensors to improve the prediction accuracy at another sensor location since its contribution to the overall cost function is 0 anyway. The efficiency of this algorithm is very high since it avoided the costly matrix operations but instead uses a very simple objective function as the optimization target. The experimental results presented in section 3.6 will demonstrate the effectiveness of this cost function. In general the results produced by this algorithm are not always as good as the first algorithm but they are quite close.

**Algorithm 3** Placement 2

**Require:** The desired number of sensors $m$, all grid location set $T$ and the thermal correlations between all pairs of grids

**Ensure:** Sensor location set $S$ with $|S| = m$

1: $S \leftarrow \emptyset$
2: **while** $|S| < m$ **do**
3: **for all** $i \in T \setminus S$ **do**
4: $S_{new} \leftarrow S \cup i$
5: Based on $S_{new}$, calculate the new cost (eqn. (3.27)):
$$cost = \sum_{\forall i \in T} \max \left( 0, \quad 1 - \sum_{\forall j \in S_{new}} c_{ij}^2 \right)$$
6: **end for**
7: Select $i_{min}$ which results in the minimum *cost* among all $i$
8: $S \leftarrow S \cup i_{min}$
9: **end while**

Note the two heuristics described in Algorithms 2 and 3 ignore the compression error at this stage. The technique for incorporating the compression error will be discussed in the next section.

### 3.3.3 Incorporating Fusion Center and Sensor Design Considerations

Once a sensor placement is decided, the techniques presented in section 3.2 are used to control the granularity of the information transmitted from these sensors to

the central register. The finite size $M$ of this register and the finite wiring space as well as the lack of informativeness of the located sensors force us to perform this compression. After compression, some sensors may be compressed to a very high degree. This is because while placing sensors we did not account for these physical limitations. Also the placement algorithm is a heuristic and may not reach the global optimal solution. We would like to use compression rates of the placed sensors to replace some sensors with the target of improving the quality of the solution. Basically, we would like to account for the compression error while performing the placement. We present a simple yet effective feedback system by defining a scaling factor $Scale_i$ for each placed sensor $i$ which is proportional to the degree of compression (small $Scale_i$ implies high compression). We get rid of all sensors (say $k$) whose scaling factor is below a threshold. If doing this removes all the sensors then the threshold is too high. Now, for all the leftover sensors, we change $c_{i,j}$ (the thermal correlation between a sensor grid $i$ and a non-sensor grid $j$) to $c_{i,j} \times Scale_i$. This implies that due to the compression, the information provided by sensor $i$ gets reduced as well. We recompute the placement cost using the new correlations in equation (3.27) (after removing $k$ sensors). Now we add exactly $k$ new sensors in the same greedy fashion as described above. We then re-evaluate the compression policy and iterate if necessary.

To choose an appropriate threshold for the scaling factor, let us first define a simple scaling factor for illustrative purposes:

$$Scale_i = \frac{2^{s_i}}{2^{b_i}} \tag{3.28}$$

Here $s_i$ is the number of bits allocated to sensor $i$. $b_i$ is the maximum possible number of bits for this sensor. The above equation defines a scaling factor for the $i$-th sensor and it satisfies the general requirement. Note that if a sensor is allocated 0 bits then $Scale_i$ will evaluated to a value close to zero (e.g. $1/2^8$, which intuitively means instead of reporting 256 potential temperature values with all 8 bits in the non-compressed case, the sensor can now report only one single value - the sensor is useless). If the sensor is not compressed at all then $Scale_i$ is equal to 1 (full

information available). Now the threshold could be chosen, for example, to be 0.1 which means all sensors with less than 10% of the full information should be removed to give room in favor of other potentially better sensor locations. This means with a maximum of 8 bits for each sensor, all those with less than 5 bits allocated should be removed to give space for new (potentially better) sensors for the next iteration of the placement algorithm. In practice this threshold could be further tuned based on the experience of the designer.

## 3.4 The Complete Design Flow

In order to develop this sensing infrastructure, we first generate all the necessary statistical information for the grids and sensors [68, 69]. Then we generate an initial sensor placement while ignoring the compression factors of sensors. Using this sensor placement, we decide the sensor compression factors. The maximum number of bits that a sensor can get ($b_i$ in equation (3.12)) depends on how much space we have to route its data. In general, sensor locations which are further from the central register and/or in congested areas could be given a smaller $b_i$. In this way the area and routing overhead can easily be incorporated by appropriate choice of $M$ and $b_i$. The compression algorithm distributes the bits to sensors globally and also decides the compression policy locally at each sensor (which observations to compress), thereby giving design specifications to the encoder stage of the sensor. This information is then fed back to the placement engine which accounts for the compression by scaling the correlation information of the sensors appropriately. Then it generates a new placement while accounting for the compression. This process is repeated until the solution converges.

It is noteworthy that our approach accounts for sensor design, compression, fusion and sensor placement in one unified perspective. It also accounts for the space available for placing sensors, central register and routing the data wires. The complete design flow is illustrated in figure 3.5 below.

In some cases it is also possible that the thermal correlation map could change

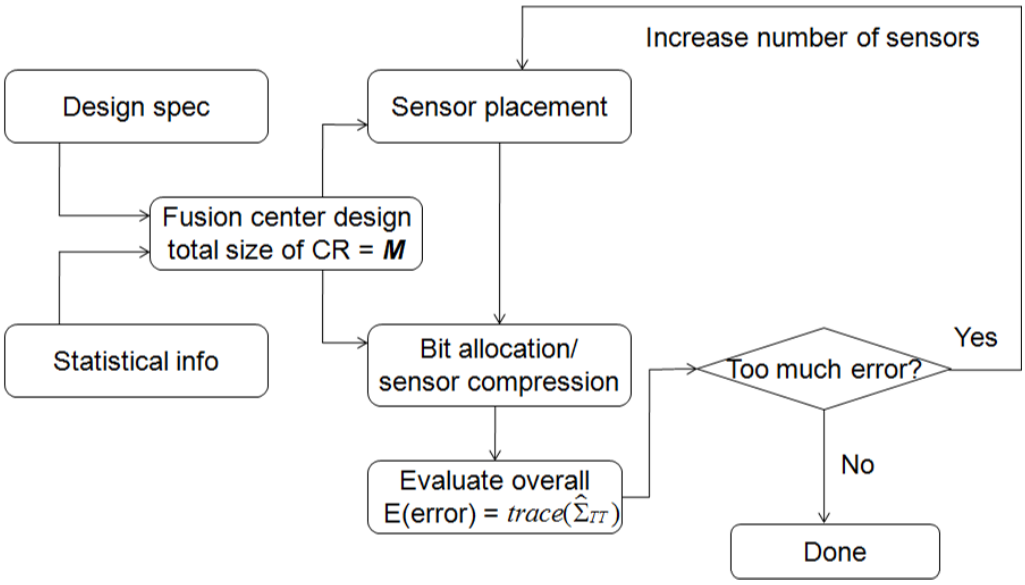



Figure 3.5: The complete design flow (CR: central register)

at runtime. One possible solution for this scenario is to summarize the system thermal behavior using several correlation maps (for example, each representing a different configuration of a reconfigurable system, or each representing a different workload cluster for a multi-core system). Our framework could then be applied to generate a design for each such potential correlation map. Every design generated this way should be evaluated on all possible correlation maps to determine its overall performance. The selection criteria is a choice of the designer. For example, we can select the desirable design based on either its average performance or worst-case performance across all correlations maps. We leave the possibilities of other potential better solutions for future work.

## 3.5 Implementation Overhead

Based on MATLAB implementation, our statistical framework could generate a complete thermal sensing infrastructure (placement, compression, data fusion) in less than 10 seconds. Note that this process is needed only once when designing the infrastructure. Once implemented the runtime overhead is very small: the sensor compression is done in a distributed fashion at each sensor locally (as a simple

encoder stage which consists of 2-level gate logic). The hypothesis testing part can be implemented using look-up tables. Fusion center only needs to process the simple linear equation (3.1) which can be computed in about 0.006 seconds for $16 \times 16$ granularity with our MATLAB implementation (a C/C++ implementation should be even faster). Data parity check is not necessary since all communication happens in an on-chip environment which means the transmission error is unlikely. Even if there are rare occurrence of such errors, its impact on the overall estimation accuracy would not be too high since the estimation is based on thermal data collected from multiple sensors. In reality, using the results generated by such a carefully designed sensing system would improve the control decisions made by the DTM unit.

## 3.6 Experimental Results

In this section we present our experimental results. One practical way of generating the required statistical information (mean, variance, correlation and prior probability of hypotheses) is through simulation of typical chip workloads, though other approaches are also possible. For example, there are existing well-accepted power and thermal simulation tools which can be utilized to achieve this purpose. In this thesis we used Wattch and HotSpot respectively. First, by feeding the typical chip workloads or programs into Wattch, we can obtain module-wise power dissipation information of the chip for each potential workload/program. Then we can feed such power information (in proportion to how often they are executed in practice) into the HotSpot model to simulate the chip thermal dynamics. The dynamic thermal map can then be sampled at fixed intervals to provide us a sample set of realistic thermal maps of the chip from which we can calculate the sample mean, sample variance and correlation information etc. They can also be used to obtain the relative frequency of appearance for each hypothesis (which essentially represents prior probabilities of the hypotheses). Alternatively such information can be obtained by having a test chip running typical applications and having infrared photos taken (more on this please see [39] [26]). Such photos would be more realistic

thermal data from which we can also extract the statistical information in a similar fashion. For the following experiments we used the simulation-based approach. To make our experiments more practical we separated the training set and the testing set. An important point to note is that even if such estimated statistical information is not 100% accurate, we could still generate a better thermal estimate of the chip than the range-based method (see table 3.1).

Next we will describe our experimental settings: Firstly, we used Wattch with Alpha binary to generate the power consumption data for each functional unit. We configured a high performance aggressive processor with pipeline width of 8 instructions and an instruction window of 128 instructions. Level 1 caches (both instruction cache and data cache) are 32KB 4-way set associative. All the caches in the hierarchy are using LRU replacement policy and a block size of 64 bytes. The physical dimension of the chip is $10mm$ (length)$\times 10mm$ (width)$\times 0.5mm$ (thickness). The average overall power dissipation of this processor is $60W$ (similar to Pentium 4). We assumed a simplified floorplan of the processor core for this illustrative experiment (see figure 3.6).

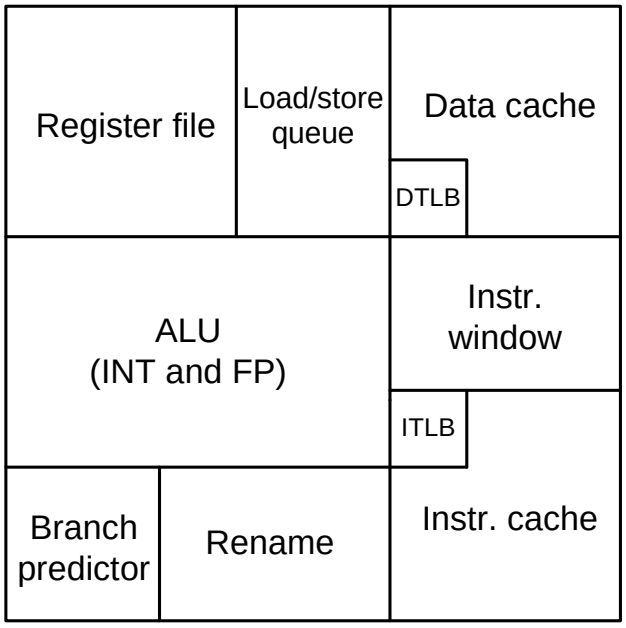



Figure 3.6: A simplified floorplan used in our experiments

For benchmarks, we arbitrarily chose 10 benchmarks from SPEC-2000 suite as

our training set (facerec, ammp, applu, bzip2, crafty, art, apsi, eon, equake, fma3d). They are compiled with the default parameters provided with the suite and are simulated for 30 minutes duration. In this process we randomly choose and run a different benchmark every 5 seconds (to make our simulation more realistic). Then we fed the obtained module-wise power consumption data into a HotSpot thermal-RC model to simulate the temperature profile of the chip for the entire 30 minutes duration. The thermal dynamics for the entire chip are then sampled to provide us with a sample set from which we could generate an estimate for the system statistical information required by our approaches. The same simulation process is carried out for the testing set (gap, gcc, gzip, lucas, mcf, mesa, mgrid, parser, perl, sixtrack, swim, twolf, vortex, vpr, wupsie) to generate the actual temperature of the chip (based on which we could calculate the RMS error in table 3.1). The model parameters are in line with the values shown in [53]: the thermal conductivity (inverse of thermal resistance) per unit volume is $100W/(m \cdot K)$. The thermal capacitance per unit volume is $1.75 \times 106J/(m^3 \cdot K)$. For the overall effect of heat sink the thermal resistance is $0.026K/W$ and the thermal capacitance is $8.8J/K$. For the effect of convection the equivalent thermal resistance is $1.0K/W$ and the thermal capacitance is $140.4J/K$.

We conducted a set of experiments for both the noiseless and the noisy sensor cases while assuming the sensors can be either infinitely precise (each sensor is assigned 10 bits to approximate this assumption) or are bounded by a total number of 16 bits altogether. We compared the solution (placement, compression and fusion) given by our statistical design framework and that of a range-based method similar to those described in [33, 36] (when the total number of bits are limited we uniformly distribute the bits among all sensors). We used a 16x16 grid granularity for both methods. For fair comparison, we only calculate and compare the average RMS error for all hotspot locations whose temperature is above a certain threshold (say, 80% of the peak temperature across the chip). This is because most of the range-based methods only allocate sensors to monitor the temperatures at such hot chip areas. In our experiments, for range-based method we look at each of these

hotspot grid cells and use the sensor observation in its closest vicinity to approximate the hotspot temperature. For our methods we use our design framework to obtain sensor placement/compression and then use our fusion algorithm to estimate the temperatures at the same set of hotspot locations. The sensor noise characteristics were obtained using Monte Carlo simulation assuming 3% standard deviation (normally distributed) in sensor parameters $W, L, V_{th}$ and $t_{ox}$ and this noise is superimposed to the actual temperatures to obtain noisy sensor readings. The RMS error is computed by comparing the thermal estimates (based on limited sensor readings) to the actual samples we obtained earlier from simulation. The average error (across all hotspots) are reported in table 3.1. Note that in this table we compared three different experimental settings:

1. NC/NL: The sensors are assumed to be infinitely precise (no sensor compression, abbr. NC) and noiseless (NL).

2. C/NL: The sensors are assumed to be noiseless but are compressed so that their combined readings can be fitted into the central register of 16 bits.

3. C/N: The sensors are assumed to be both noisy and compressed so that their combined readings can be fitted into the central register of 16 bits.

Table 3.1: RMS error and runtime for different experimental settings

| #sensors | RMS for range-based method (°C) | | | RMS for our method (placement1) (°C) | | | RMS for our method (placement2) (°C) | | | Improvement (for C/N case) | | Runtime (s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NC/NL | C/NL | C/N | NC/NL | C/NL | C/N | NC/NL | C/NL | C/N | placemt.1 | placemt.2 | placemt.1 | placemt.2 |
| 2 | 14.75 | 14.78 | 16.55 | 11.81 | 11.92 | 12.13 | 11.85 | 11.90 | 13.31 | 26.71% | 19.58% | 3.98 | 2.17 |
| 3 | 13.45 | 13.49 | 15.11 | 10.53 | 10.61 | 11.24 | 11.02 | 10.18 | 12.24 | 25.61% | 19.00% | 5.46 | 2.86 |
| 4 | 12.84 | 13.06 | 14.63 | 9.46 | 9.57 | 9.93 | 9.79 | 10.01 | 11.37 | 32.12% | 22.28% | 5.82 | 3.57 |
| 5 | 11.81 | 12.58 | 14.09 | 8.62 | 8.99 | 9.41 | 8.86 | 9.42 | 11.15 | 33.22% | 20.87% | 6.96 | 4.23 |
| 6 | 11.41 | 13.15 | 14.73 | 7.89 | 8.81 | 9.23 | 8.18 | 9.26 | 10.90 | 37.33% | 26.00% | 8.10 | 5.30 |
| 7 | 11.32 | 13.84 | 15.50 | 7.34 | 8.73 | 8.92 | 8.04 | 9.23 | 10.98 | 42.45% | 29.16% | 9.19 | 6.36 |

Table 3.1 highlights the RMS error comparison for various experimental settings. We can see from this table that as we increase the number of sensors the error generally goes down. Another important observation is that our methodology can achieve higher accuracy even with fewer compressed sensor observations when

compared to the range-based method (with no compression and more sensors). Our method requires at most 9 seconds to generate the overall solution and thus can be easily implemented at design time with little extra effort. The worst-case estimation error for our method and the ranged-based method (with 5 sensors and 16 bits size constraint of central register) are 18.35°C and 13.24°C respectively. These worst-case values could be used when considering temperature guardbands for a design. In table 3.2 we also report our runtime for designing the complete sensing infrastructure for different chip granularities.

Table 3.2: RMS error and runtime comparison for different chip granularities ("Placement 1" algorithm, 5 sensors, M=16, noisy sensor)

| granularity | $8 \times 8$ | $16 \times 16$ | $32 \times 32$ |
|---|---|---|---|
| Runtime | 4.27s | 6.36s | 10.48s |
| RMS error (°C) | 10.13 | 8.56 | 8.24 |

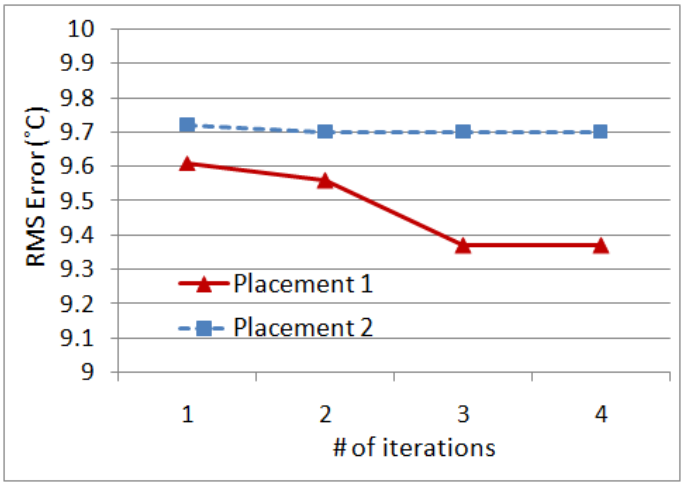



Figure 3.7: Accuracy improvements by refining sensor placement

The overall estimation accuracy obtained using both of our placement heuristics are compared in figures 3.7, 3.8 and 3.9. As explained in section 3.3.3, if we incorporate the sensor compression information from the later design stage into the placement engine, we can re-allocate some of the sensors and achieve better overall accuracy. Such a placement refinement process is repeated until convergence. It can

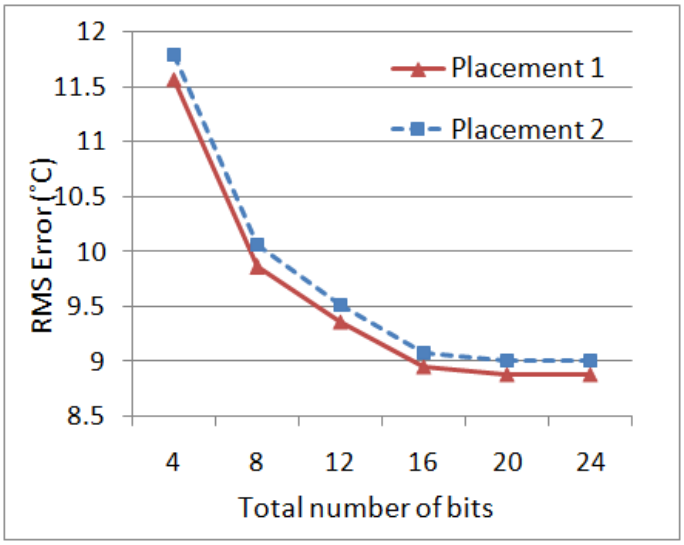



Figure 3.8: Error comparison for different central register size (M))

be observed from figure 3.7 that we can indeed reduce the overall RMS error by cooperatively designing the sensor placement and compression schemes (the compression is required by the limited central register size). The improvement usually converge in less than 5 iterations. In figure 3.8 we highlight how the RMS error changes as we fix the sensor placement scheme (number and location) while increasing the total number of bits at the central register. It can be seen the error goes down as each sensor are allocated more bits. Note that the RMS error for the range-based placement scheme is 12.84 °C even with no compression. The accuracy improvement clearly demonstrate the effectiveness of our thermal sensing infrastructure. This becomes more clear as we plot all data in the same graph (see figure 3.9). It can be observed that if we increase the total number of sensors the error generally goes down. The only exception is the case for range-based method with 16 bits central register size constraint. The error actually increases a little bit after 5 sensors. This is because there is a fundamental limitation on how precise each sensor can be posed by the central register size constraint. As the number of sensor increases, the precision of each sensor degrades since the bits are uniformly distributed among all sensors. This is not a problem for our method since we have an explicit scheme for compressing sensors. Specifically we look at the relative importance of each sensor and determine

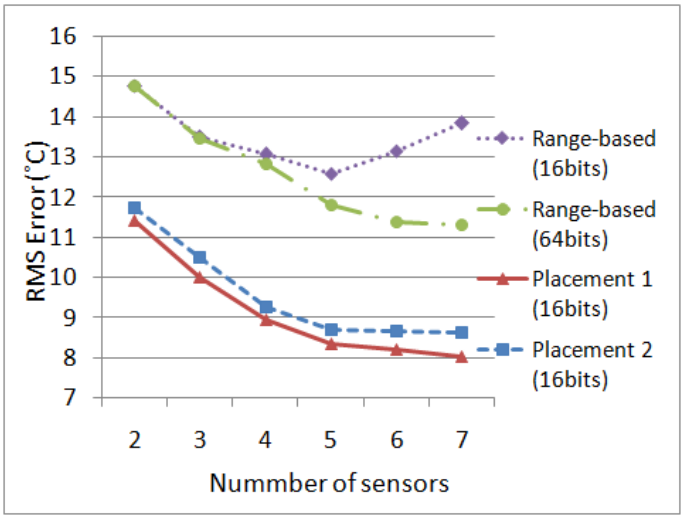



Figure 3.9: Error comparison for different settings

how they should be compressed in order to achieve the best accuracy. The sensors which provide more information than others get more bits allocated whereas the less important sensors are compressed further. Thus if we increase the number of sensor beyond 5, the error tends to converge instead of increasing. This signifies that no more information can be obtained given the 16 bits limitation in the central register size.

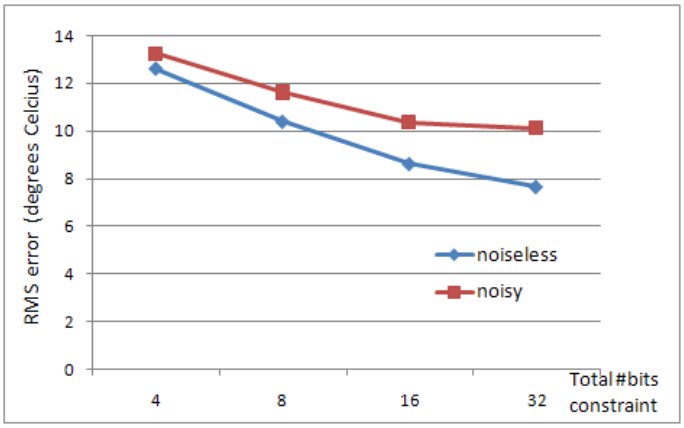



Figure 3.10: Error comparison for noiseless and noisy cases (varying M constraints)

Figure 3.10 highlights how the RMS error changes when we increase the central register size constraint M from 4 to 32 while fixing the number of sensors to 6 (with a fixed placement scheme). We can observe a steady decrease in the RMS error for

the noiseless case and a first-decrease-then-converge curve for the noisy case. This is because in the noiseless case the sensors can accurately reflect its local temperatures, the more bits we have the more accurate our estimates will be. However in the noisy sensor case, because the sensor observations are corrupted by noise, more bits at each sensor do not necessarily help us gain extra information and hence the error tends to converge after a certain point.

In the following we demonstrate how our thermal sensing infrastructure performs when applied to the dynamic chip system. We used the popular thermal-RC model (with parameters set similar to those used in HOTSPOT tool [53]) in the following experiment. In this model each grid cell $i$ has an associated thermal resistance $R_i$ (or equivalently a thermal conductance $G_i = 1/R_i$). Each grid cell $i$ also has a thermal capacitance $C_i$. Between any two neighboring grid cells $i$ and $j$ there is a cross thermal resistance $R_{ij}$ (or $G_{ij} = 1/R_{ij}$). The grid temperature is modeled by the node voltage. The differential equation governing the transient temperature $T_i(t)$ at grid cell $i$ can be expressed as follows ($N_i$ is the set of all neighboring grids for $i$):

$$\sum_{j \in N_i} (T_i(t) - T_j(t)) G_{ij} + C_i \frac{dT_i(t)}{dt} - P_i(t) = 0 \tag{3.29}$$

We simulated an randomly chosen sequence of SPEC 2000 benchmarks using Wattch and then feed the power dissipation data into equation (3.29). By solving this equation we could obtain the dynamic temperature change of the system [63]. This is used as the reference for comparing the accuracy of our methods. We applied both our schemes (including sensor placement, compression and fusion) and the ranged-based method (with uniform sensor compression) and report the results in figure 3.11. The data was obtained for a randomly chosen hotspot chip location (which does not coincide with the sensor locations). The actual temperatures (solid blue curve in the figure) are obtained from the solution of equation (3.29). For our method we placed 7 sensors on the chip and for range-based method 10 sensors are deployed. A total number of 32bits is assumed to be the size constraint of the central register. As a result the sensors get compressed and the observed temperatures

become discrete. Such discrete and compressed temperature observations are sampled from sensors every second and then combined into the central register. The data from this register is subsequently used to generate thermal estimates based on our fusion algorithm (see [74] and equation (3.1)). As can be seen, our results (green segmented curve) follows the real temperature more closely as compared to the results generated by the range-based method (red dotted curve). This demonstrates the effectiveness of our statistical framework in estimating the dynamic chip temperature.

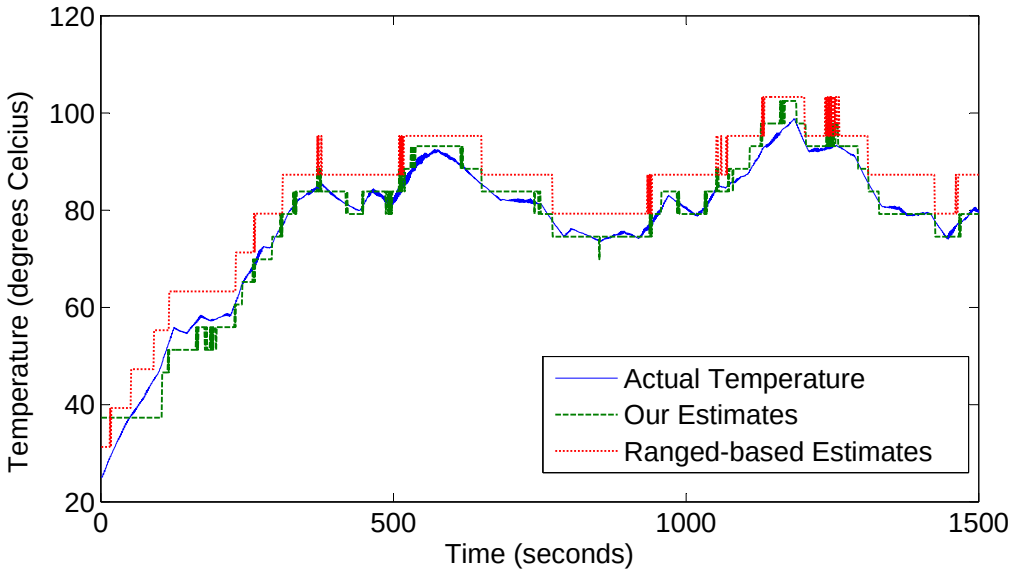



Figure 3.11: Dynamic temperature tracking: actual vs estimated

Though the results demonstrated in this section are generated on a single-core processor, our approaches can be applied to multi-core processors as well. Our framework exploits the thermal correlation map to drive sensor placement and data fusion. In a multi-core scenario, the correlation are likely to be stronger within each core and weaker between different cores. Thus, driven by our placement algorithm, each core will essentially get its own sensors which will then be placed at the most informative locations within the core.

# Chapter 4

# Adaptive and Autonomous Thermal Tracking

## 4.1 Motivation

In this chapter, we propose a methodology for adaptive and autonomous thermal tracking. Many existing work on the general topic of temperature tracking focus on estimating chip thermal state with a few given sensors. Any change in the underlying workload characteristics are largely ignored. Some of the existing effort are discussed below. Cochran et al. used spectral methods [15] to characterize the thermal profile of a chip. Zhang et al. exploited the logical/spacial correlations in temperature between sensors and other chip locations to estimate the thermal profile. Sharifi et al. used Kalman filter (KF) based method [50] to track the dynamic change in chip temperature. Jung et al. also developed a Kalman filter based approach to issue alerts for potential thermal hot spots [24]. Among all these methods, Kalman filter based techniques are quickly gaining popularity since they are capable of exploiting the statistical power characteristics to improve the estimation accuracy (note that a chip's temperature is a strong function of its power dissipation). They also explicitly account for sensor noise when generating thermal estimates. Though the standard Kalman filter based approach is quite effective, it also has several drawbacks which we will discuss below.

First of all, despite the fact that the statistical workload characteristics may change at runtime (which will in turn cause the statistical power characteristics to change), the standard Kalman filter could not detect or adapt to any such changes. This can lead to erroneous results or degraded performance of the filter. In practice, the change in statistical power characteristics (mean, variance and etc.) is often caused by varying workload characteristics (e.g. switching from multimedia processing to word processing, or from integer-heavy operations to floating-point heavy

operations). Such changes can also be caused by various dynamic power management policies taking effect at runtime (e.g. switching from low-power/sleep mode to full-speed mode, etc.). In such a scenario, the traditional Kalman filter based approach could no longer produce the desired estimation accuracy. In this thesis, we address the problem of real-time adaptive thermal tracking by accounting for dynamic changes in the statistical power characteristics caused by various factors. Using Kalman filter as the core estimation engine, we first develop schemes for autonomous detection of changes in the underlying power characteristics. We then propose adaptive techniques to tune our filter at runtime in order to continuously produce optimal estimation results. Such awareness of system's power state can prove to be critical in tracking the thermal state of today's general-purpose processors. To achieve our goal, we proposed two adaptive schemes which are based on the concept of residual whitening and hypothesis testing respectively. The former is more suitable for relatively stable systems (the statistical characteristics of power do not change too quickly). The latter is designed for systems that switch among different applications very frequently. Experimental results showed that our adaptive method is capable of achieving 60% accuracy improvement over the traditional approaches which do not adapt to changes in system's power characteristics. The implementation overhead for our adaptive filter is also very small. For the residual whitening based method only a set of linear operations are involved. The hypothesis testing based approach can be implemented using table lookup strategy.

Another drawback of the standard Kalman filter is that it is built upon a linear system model which ignores the nonlinear dependency of leakage power on temperature. As technology continues to scale down, this nonlinear effect of leakage is no longer negligible. It is reported in [25, 46] that the leakage power can contribute to about 50% of the total power consumption under the current technology node. Note that leakage power has the undesirable characteristic that it could increases exponentially with temperature [22, 56]. More leakage power will in turn lead to higher chip temperature, thus making today's silicon chip a positive feedback system. Ignoring this important phenomenon could lead to underestimation of the

real temperature experienced by a chip. In this chapter we introduce the extended Kalman filter theory to explicitly model and account for the nonlinear leakage effect in our Kalman filter formulation. By doing so the estimation accuracy could be improved significantly compared to the traditional approach. Such a leakage-aware extended Kalman filter could be easily combined with our adaptive approaches to further improve the thermal tracking accuracy.

## 4.2 Preliminary

In this section we first explain how to model the dynamic thermal behavior of a silicon chip as well as its thermal sensors. We then introduce the Kalman filter based estimation framework.

### 4.2.1 Thermal RC Model

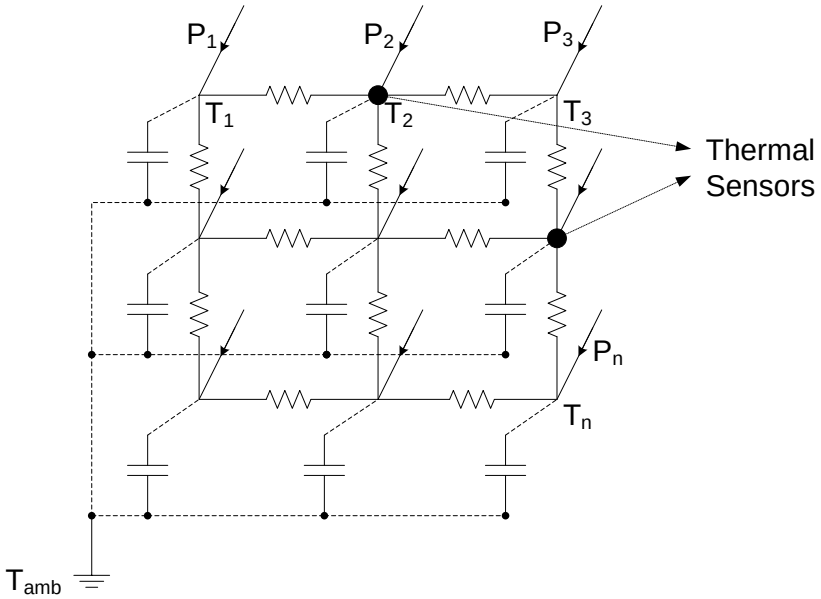



Figure 4.1: Equivalent thermal-RC model of the chip with on-chip thermal sensors

Due to the well-known duality between heat flow and electrical phenomena, the thermal dynamics of a chip system can be modeled by a thermal-RC circuit [63, 53], see figure 4.1. Each node in this circuit corresponds to a chip grid location (there

are also nodes representing the heat sink/package). The temperature $T_i$ at node $i$ can be represented by the voltage at that node. Similarly, the power dissipation $P_i$ can be modeled by a current inflow at node $i$. There is also a thermal capacitance $C_i$ associated with each node which represents its ability to hold heat. Between any neighboring nodes $i$ and $j$ (and between any node and the package) there is also thermal resistance $R_{ij}$ (or conductance $G_{ij} = 1/R_{ij}$) which captures how quickly heat can be transferred between the two nodes. In this way, the thermal dynamics at each grid location $i$ can be described by the following differential equation (see details in [63, 53]):

$$\sum_{j \in N_i} G_{ij}(T_i(t) - T_j(t)) + C_i \frac{dT_i(t)}{dt} - P_i(t) = 0 \tag{4.1}$$

In equation (4.1), $t$ represents the time and $N_i$ is the set of all neighbors of node $i$. $T_i(t)$ and $P_i(t)$ are respectively the thermal state and the power consumption of node $i$ at time $t$. Converting to discrete-time state space, the above differential equation can be approximated by the following difference equation (see [63, 50]):

$$T[n] = AT[n-1] + BP[n-1] \tag{4.2}$$

Here $T[n]$ and $P[n]$ are vectors representing the temperature and power values for all chip grids at time $n$. The coefficient matrices $A$ and $B$ are determined by the circuit parameters ($G_{ij}$ and $C_i$) and the chosen length of time steps. They can be computed as shown by [53]. Equation (4.2) essentially captures how the system's thermal state at time $n$ depends on its thermal state and power dissipation at an earlier time $n-1$.

Due to unpredictability of workloads (vector $P[n]$ is unknown until runtime) and fabrication/environmental variabilities, the exact value of $T[n]$ at runtime can be very hard to predict. This has motivated designers to place sensors at various chip locations to monitor chip temperature at runtime (see figure 4.1). These on-chip sensors can provide us with an observation vector $S[n]$ which is essentially a subset of $T[n]$ plus some sensor noise $v[n]$:

$$S[n] = HT[n] + v[n] \tag{4.3}$$

In equation (4.3), $H$ is simply a transformation matrix determined by the sensor placement. It has dimension $s \times m$ if vectors $S[n]$ and $T[n]$ each has $s$ and $m$ elements respectively. As explained earlier, sensors can be quite noisy, especially in on-chip environments. $v[n]$ is a Gaussian random vector with zero mean which is used to represent sensor noise [50]. Note that sensors come with power and area overheads. This means their number and placement are highly constrained ($s \ll m$). The problem of estimating the entire thermal profile (vector $T[n]$) based on very limited sensor observations ($S[n]$) at runtime is rather complex.

Recently, many techniques have been proposed to solve the above problem [15, 74, 50, 24]. Among these techniques, Kalman filter based method is a very promising research direction: it is capable of generating thermal estimates for all chip locations while countering sensor noise. It is also quite efficient and could be applied to real-time thermal tracking problems. In the next section we briefly introduce the Kalman filter methodology.

### 4.2.2 Kalman Filter Based Thermal Tracking

Typically, the power dissipation $P[n]$ can be viewed as a random quantity which are determined by many factors. For example, $P[n]$ depends on the application being executed, input data patterns, task scheduling, dynamic frequency scaling and etc. The sensor noise $v[n]$ is also affected by various factors such as supply voltage fluctuation, cross coupling, fabrication variability and etc. According to Central Limit Theorem, the collective effect of many different and independent sources of randomness will approximate a Gaussian distribution. The Kalman filter theory [17] states that if the randomness in $P[n]$ and $v[n]$ has Gaussian nature, Kalman filter could generate optimal estimates for $T[n]$ given the linear model described in equations (4.2) and (4.3).

In Kalman filter two sequential steps are carried out repetitively: *predict* & *correct*. They are described by equations (4.4) to (4.8) below:

*predict:*

$$T[n|n-1] = AT[n-1|n-1] + B\mu^p \tag{4.4}$$

$$C[n|n-1] = AC[n-1|n-1]A^T + BQB^T \tag{4.5}$$

*correct:*

$$T[n|n] = T[n|n-1] + K[n](S[n] - HT[n|n-1]) \tag{4.6}$$

$$K[n] = C[n|n-1]H^T(R + HC[n|n-1]H^T)^{-1} \tag{4.7}$$

$$C[n|n] = (I - K[n]H)C[n|n-1] \tag{4.8}$$

---

In the above equations, $R$ is the covariance matrix of sensor noise $v[n]$. $C[n|n-1]$ and $C[n|n]$ are the error covariance matrices associated with $T[n|n-1]$ and $T[n|n]$ respectively. $I$ is the identity matrix. $K[n]$ is the Kalman gain which is chosen to minimize the expected estimation error at each time step.

The Kalman filter works in the following way: in the *"predict"* stage the filter uses the average power dissipation $\mu^p$ and the temperature history to predict the current temperature ($T[n|n-1]$ in equation (4.4)). As soon as a new sensor observation becomes available, the filter adjusts its prediction in the *"correct"* stage using the new sensor input $S[n]$ to produce a more accurate thermal estimate $T[n|n]$ (see equation (4.6)). These two stages are carried out iteratively as time progresses, thus continuously tracking the dynamic thermal profile of the system. The filter also updates the error covariance matrices $C[n|n-1]$ and $C[n|n]$ associated with the estimates $T[n|n-1]$ and $T[n|n]$ at each step based on the variance of power and sensor noise (see equations (4.5) and (4.8)). Such error covariance matrices are indicators of the potential error of the filter at each step. They are also used to calculate the optimal Kalman gain $K[n]$ in equation (4.7).

Note that usually $R$ (the covariance of sensor noise) does not change for a relatively long period of time. It is also noteworthy that if the system is only running a set of similar applications, the statistical characteristics of $P[n]$ stay relatively stable as well. In such a scenario, the Kalman filter will quickly stabilize which

means $C[n|n-1], C[n|n]$ and $K[n]$ will all converge to static values. This is called the steady-state of Kalman filter. During the steady state, even though our thermal estimates may change time to time, the error covariance $C[n|n]$ stays the same. In a highly dynamic chip environment, however, this assumption becomes problematic for reasons we explained earlier. In the next section we discuss how to overcome this drawback of standard Kalman filter using our adaptive thermal tracking techniques.

## 4.3 Adaptive Tracking Based on Residual Whitening

### 4.3.1 Autonomous Detection

In this subsection we first explain how we can autonomously detect any potential change in system's statistical power characteristics. Let us start by defining a power state $k$ as the state of the system in which the statistical power parameters of the chip (mean $\mu^P$ and covariance $Q$) stay stable. This implies that $\mu_k^P$ and $Q_k$ are both constants for a certain power state $k$. As explained earlier, each state $k$ essentially captures the system's power behavior for a set of similar applications. We further assume that the statistical parameters $\{\mu_k^P, Q_k\}$ for each state can be pre-characterized by simulating typical chip workloads/benchmarks (for example, as what we did for the experiments in section 4.7). Such statistical parameters are provided as input to our methods. Our *residual whitening* based approach takes advantage of a useful property of the residual process $e[n]$ which is defined below:

$$e[n] = S[n] - HT[n|n-1] \tag{4.9}$$

$$= HT_{real}[n] + v[n] - HT[n|n-1] \tag{4.10}$$

$$= -Hx[n] + v[n] \tag{4.11}$$

Here (4.10) is obtained by plugging (4.3) into (4.9). $x[n] = (T[n|n-1] - T_{real}[n])$ represents the error of our thermal prediction $T[n|n-1]$. The autocorrelation function of the residual process is defined as

$$ac_i = E(e[n] \cdot e[n-i]^T) \tag{4.12}$$

Essentially $ac_i$ is the autocorrelation of the error process between time $n$ and $n-i$.

Our *residual whitening* based approach described in this section is only appropriate for the case where the system stays in each power state for relatively long period of time, so that the Kalman filter can enter its steady-state between successive power state switch. There is a useful property of Kalman filter which states that once the filter has entered steady state, the autocorrelation $ac_i$ of the residual process will be zero for all $i$ except $i = 0$ as long as the filter is operating with the correct statistical parameters $\mu^P$ and $Q$. This is called the whitening of the residual process (interested readers can refer to chapter 5 of [17] for detailed proof).

Now, the autocorrelation of the residual process can be easily estimated using a combination of the Kalman filter outputs and the sensor observations based on the following equation:

$$\hat{ac}_i = \frac{1}{N}\sum_{n=1}^{N}(e[n] \cdot e[n-i]^T) \tag{4.13}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left((S[n] - HT[n|n-1]) \cdot (S[n-i] - HT[n-i|n-i-1])^T\right) \tag{4.14}$$

Thus, during the filter operation, we can keep record of a running window of past sensor observations & Kalman filter predictions and then use them to estimate the autocorrelation of the residual process in an online fashion [70]. If the estimated $ac_i$ is sufficiently close to zero, then the residual-whitening property is satisfied and we know the statistical power parameters $\mu^P$ and $Q$ that were used to operate Kalman filter is correct. Otherwise a power state switch must have occurred and we must update the filter with new parameters. Based on this principle of *residual whitening* we can design our adaptive Kalman filter in the following way: the filter stores the most recent $N+1$ data samples (if we choose $i = 1$ in equation (4.14)) and keeps updating the autocorrelation function $ac_i$ based on (4.14). As soon as the value of $ac_i$ exceeds a certain threshold $ac_t$, the filter knows that a power state change has happened and can react accordingly. This approach does not require any external interference and the filter can detect the change in power state autonomously.

### 4.3.2 Adaptive Tracking Algorithms

Once our adaptive filter detects the power state change, there are two things that need to be done: 1) we need to be able to predict the correct new state $k^*$ and 2) the filter parameters $(\mu^p, Q)$ need to be updated to their correct values $(\mu^P_{k^*}, Q_{k^*})$. To predict the new power state we could do the following: for each potential power state $k$, we rewind the filter to the point where the power state change happened and re-compute $ac_i^k$ using $\mu^p_k, Q_k$. Then we could simply choose the state $k^*$ which results in the smallest $|ac_i^k - 0|$:

$$k^* = \underset{k}{\operatorname{argmin}} |ac_i^k - 0| \tag{4.15}$$

In the simplest case $i$ can be chosen as 1. Essentially this scheme chooses the state that minimizes $ac_i^k$ (the one closest to zero) as our predicted new state. There is one more complication though. If we only know that a switch in power state has occurred but do not know where exactly it happened in time, we would not be able to compute the correct value of $ac_i^k$ for each state $k$. As long as we know the exact state switching time $t_{sw}$, rewinding Kalman filter to this point and computing $ac_i^k$ for for each state $k$ is relatively easy. Next we give a divide-and-conquer algorithm for detecting the switching time of the power state change (see Algorithm 4). Given the most recent $N + 1$ data samples (within which the state change has been detected) and the previous power state $l$ before the switch happened, the algorithm predicts the switching point to fall within the first half or the second half of the $N + 1$ data samples based on the autocorrelation computed on each half respectively. We recursively divide this sub-range in half until the switching point is predicted to fall within a reasonably small range (*minlength* in the algorithm). Algorithm 5 gives the overall adaptive thermal tracking procedure. Note that for these algorithms we assume Kalman filter is used as our underlying estimation engine.

**Algorithm 4** *Autonomous Detection*

**Require:** sequential data samples $\{S[n]\}$ and $\{T[n|n-1]\}$ for $n = t_b$ to $t_e$, desired detection resolution *minlength*

**Ensure:** the approximate switching time $t_{sw}$

$t_m \leftarrow \lfloor \frac{t_e - t_b}{2} \rfloor$ // *compute the middle point*

/* minlength: desired detection resolution */

**if** $t_e - t_b < minlength$ **then**

  **return** $t_{sw} = t_m$

**end if**

**for** $n = t_b$ to $t_m$ **do**

  compute residual $e[n] = S[n] - HT[n|n-1]$

**end for**

estimate $\hat{ac}_1 = \frac{1}{(t_m - t_b)} \sum_{n=t_b+1}^{t_m} (e[n] \cdot e[n-1]^T)$ for the first half data samples

**if** $\hat{ac}_1 > threshold$ **then**

  /* $\hat{ac}_1 \not\approx$ *0: the switching point falls in* $[t_b, t_m]$ */

  **return** $swdetect(t_b, t_m)$

**else**

  /* $\hat{ac}_1 \approx$ *0: the switching point falls in* $[t_m + 1, t_e]$ */

  **return** $swdetect(t_m + 1, t_e)$

**end if**

## 4.4 Adaptive Tracking Based on Hypothesis Testing

The previous approach was based on computing the autocorrelation value for the residual process for all the potential power states and choosing the one which was closest to 0. In this section we present an alternative approach which is based on hypothesis testing. In this framework, we treat each power state $k$ of the $K$ potential states as a hypothesis $H_k$. It has a probability $p_k$ of occurrence. we always check the probability of each power state using the most recent sensor observations to see if the assume power state has changed. We would like to choose the power state which has the highest probability of occurrence based on the current sensor observations.

**Algorithm 5** *Adaptive Tracking Algorithm Based on Residual Whitening*

**Require:** The sequential sensor inputs $S[n]$ and the potential power statistics $\{(\mu_k^p, Q_k)\}$ for $k = 1$ to $K$

**Ensure:** The adaptive Kalman filter thermal estimates

initialize $\mu^p \leftarrow \mu_k^p, Q \leftarrow Q_k$ with any $k$
**while** (1) **do**
  operate Kalman filter as normal
  **if** $(n\%N == 0)$ **then**
    compute $ac_1 = \frac{1}{N}\sum_{i=n-N+1}^{n}(e[i] \cdot e[i-1]^T)$
    *// detect power state change*
    **if** $|ac_1| > threshold$ **then**
      $t_{sw} = swdetect(n-N, n)$ *// switching time*
      *// predict the new system state*
      **for** $k = 1$ to $K$ **do**
        $\mu^p \leftarrow \mu_k^p, Q \leftarrow Q_k$
        rewind Kalman filter and re-compute $ac_1^k$ assuming state $k$ is true
      **end for**
      $k^* = \underset{k}{\text{argmin}}|ac_1^k - 0|$
      update filter parameters: $\mu^p \leftarrow \mu_{k^*}^p, Q \leftarrow Q_{k^*}$
    **end if**
  **end if**
  $n \leftarrow n + 1$
**end while**

Given the sensor reading $S[n]$, the posterior probability for each hypothesis $H_k$ is defined as $prob(H_k|S[n])$. In this framework, we would like to choose the power state or hypothesis with maximum posterior probability.

$$H_p = \delta(S[n]) = \underset{H_k \in H_1..H_K}{\text{argmax}} \{prob(H_k \big| S[n])\} \tag{4.16}$$

Here $\delta(S[n])$ is our decision rule and it is a function of the sensor observation $S[n]$ only: Given a certain $S[n]$, we evaluate $prob(H_k \big| S[n])$ for each potential $H_k$

and then choose the one with the highest probability. The problem is really how we can obtain the value of $prob(H_k \big| S[n])$. Note that according to Bayes' theorem we have the following relationship:

$$\begin{aligned} prob(H_k \big| S[n]) &= \frac{prob(H_k, S[n])}{prob(S[n])} \\ &= \frac{prob(S[n]|H_k) \times prob(H_k)}{prob(S[n])} \end{aligned} \tag{4.17}$$

where $prob(H_k)$ is the prior probability of $H_k$ ($prob(H_k) = p_k$). Note that the denominator in the above equation stays the same for each different hypothesis $H_k$. This means we can focus on the numerator only and simplify our decision rule to the following:

$$H_p = \delta(S[n]) = \underset{H_k \in \mathbb{H}}{\operatorname{argmax}} \{prob(S[n] \big| H_k) \times p_k\} \tag{4.18}$$

where $\mathbb{H} = \{H_1, \ldots, H_K\}$. Now we can use the following process to compute the value of $prob(S[n] \big| H_k)$. Let us assume that until time $n-1$ we knew exactly which power state we were in. Hence we know $T[n-1|n-1]$ and $C[n-1|n-1]$ accurately. At time $n$, we detect a change in power state and have an associated new sensor sample $S[n]$. Since the power state has changed we do not know $T[n|n-1]$ and $C[n|n-1]$ accurately. For each hypothesis $k$, we can use equations (4.4) to (4.8) in time step $n$ to compute new potential $T_k[n|n-1]$ and $C_k[n|n-1]$ for each hypothesis $k$. Now let the error be $x_k[n] = T[n] - T_k[n|n-1]$ for each hypothesis $k$. Thus $C_k[n|n-1]$ is the covariance matrix of $x_k[n]$.

Based on equation (4.3) we have the following:

$$\begin{aligned} S[n] &= HT[n] + v[n] \\ &= H(x_k[n] + T_k[n \big| n-1]) + v[n] \end{aligned} \tag{4.19}$$

where $v[n]$ is the sensor noise (normally distributed with zero mean and covariance $R$). Since $S[n]$ is a linear combination of Gaussian random variables, it should be normally distributed as well whose mean and covariance can be derived as follows

(assuming $x_k$ and $v$ are uncorrelated):

$$\mu_k^S = E\left[H(x_k[n] + T_k[n\big|n-1]) + v[n]\right] = HT_k[n\big|n-1] \tag{4.20}$$

$$\Sigma_k^S = Hcov(x_k[n])H^T + cov(v[n]) = HC_k[n\big|n-1]H^T + R \tag{4.21}$$

Here $cov(\cdot)$ represents the covariance of a certain random vector. Now that (given a certain hypothesis $H_k$) we know $S[n]$ has Gaussian distribution with mean $\mu_k^S$ and covariance $\Sigma_k^S$, the probability $prob(S[n]\big|H_k)$ can be easily obtained based on the following multivariate Gaussian density function:

$$\begin{aligned} &prob(S[n]\big|H_k) \\ &= \frac{1}{(2\pi)^{N/2}|\Sigma_k^S|^{1/2}} exp\{-\frac{1}{2}(S[n]-\mu_k^S)^T(\Sigma_k^S)^{-1}(S[n]-\mu_k^S)\} \end{aligned} \tag{4.22}$$

Once we have $prob(S[n]\big|H_k)$ the decision rule in equation (4.18) can be easily evaluated (note that $prob(H_k) = p_k$ is the prior probability).

Though the above described method is effective, its accuracy relies heavily on the amount of noise in the system and also how accurately we can determine the switching time of the power states. This is because the above scheme depends too much on $S[n]$: the information sampled at a single time instance. It tends to make mistakes if such a $S[n]$ is erroneous due to noise. We can improve the accuracy of this method by considering $m$ sequential sensor observations $\{S[n], S[n-1], \ldots, S[n-m+1]\}$. By exploiting the information sampled at multiple time instances, the noise associated with each one gets canceled out and a more accurate prediction can be achieved. In such a scenario our decision rule is similar:

$$\begin{aligned} H_p &= \delta(S[n], S[n-1], \ldots, S[n-m+1]) \\ &= \underset{H_k \in \mathbb{H}}{\text{argmax}}\{prob(H_k \mid S[n], \ldots, S[n-m+1])\} \end{aligned} \tag{4.23}$$

$$= \underset{H_k \in \mathbb{H}}{\text{argmax}}\{prob(S[n], \ldots, S[n-m+1] \mid H_k) \times prob(H_k)\} \tag{4.24}$$

Note equation (4.24) is obtained by applying the Bayes' theorem and noting the fact that the denominator stays the same for each hypothesis $H_k$. To evaluate

this decision rule we can expand the conditional probability as follows:

$$
\begin{aligned}
& prob(S[n], S[n-1], \ldots, S[n-m+1] \mid H_k) \\
&= prob(S[n-m+1] \big| H_k) \times prob(S[n-m+2] \big| S[n-m+1], H_k) \\
&\quad \times \ldots \times prob(S[n] \mid S[n-1], \ldots, S[n-m+1], H_k) \qquad (4.25) \\
&= \prod_{i=n-m+1}^{n} prob(S[i] \mid T_k[i \big| i-1], C_k[i \big| i-1], H_k) \qquad (4.26)
\end{aligned}
$$

Note that at a certain time $i$, Kalman filter uses all previous sensor observations up to time $i-1$, combined with the statistical information $H_k \sim \mathcal{N}(\mu_k^p, Q_k)$ to generate $T_k[i \big| i-1]$ and $C_k[i \big| i-1]$, which is then used to derive the conditional probability of $S[i]$ (see equations (4.20), (4.21) and (4.22)). This leads to the simplification in the last step. Note that each term in (4.26) can be computed in exactly the same way as we showed earlier when computing the conditional probability of a single observation ($prob(S[n] \big| H_k)$). Once we know how to compute $prob(S[n], S[n-1], \ldots, S[n-m+1] \mid H_k)$, the decision rule for multiple sequential sensor observations (equation (4.24)) can be easily obtained.

## 4.5 Qualitative Comparison

In general the hypothesis-testing based scheme operates at a much finer granularity in time than the residual-whitening based method. The latter method usually has lower overhead: it is activated every $N$ steps and only tries to predict the new system state if an actual change has been detected). It also has higher accuracy (more sensor observations are used to test the whiteness of the residual process: $N >> m$). However it is constrained to the case where the system stays in each state for relatively long period of time so that Kalman filter has entered steady-state. The hypothesis-testing based method has the flexibility of changing the number of sequential observations exploited (the value $m$). Thus it can adapt to any switching frequency of the system. However it has higher implementation overhead: the probability in equation (4.24) needs to be updated and decision rule needs to be evaluated at each time step. We are providing both methods here so that in practice

the most appropriate scheme can be selected based on the nature of the application (yet another level of adaptivity).

## 4.6 Leakage-aware Kalman Filter

### 4.6.1 Problem Description

In this section we discuss how our Kalman filter based adaptive-and-autonomous thermal tracking schemes can be extended to explicitly account for the leakage effect [72]. As shown by equation (4.2), the entire standard Kalman filter theory is based on a linear system model. Under this assumption, Kalman filter theory is theoretically sound and can generate optimal thermal estimates efficiently using equations (4.4) to (4.8). However, as VLSI fabrication technology continues to scale down, leakage power is contributing more and more to the total power dissipated by a chip. As reported in [25, 46] leakage power can take up to 50% of the total chip power consumption. Note that leakage has the non-linear nature that it can increase exponentially with the chip temperature. Therefore, in reality, the standard Kalman filter tends to under-estimate the actual chip temperature due to the assumed linear model. It needs to be modified or extended to account for the nonlinear leakage-temperature dependency. It is also desirable that doing so will not incur too much computational overhead since thermal tracking is usually done in real-time. In this section, we use extended Kalman filter to explicitly model the nonlinear leakage power. We then modify the standard Kalman filter formulation accordingly to account for this change. The techniques are primarily based on linearization schemes in order to keep the computational overhead to a minimum. Next, let us first explain the nonlinear leakage-temperature relationship in more detail.

There are several leakage model proposed in the existing literature. Some models use a quadratic approximation [56]. Here we use a more accurate leakage

model as described in [22]:

$$\begin{aligned} P_L &= N_{gate} \cdot V_{dd} \cdot I_{avg} \\ &= N_{gate} \cdot V_{dd} \cdot I_s(T_0, V_0) \cdot T^2 \cdot e^{((614.98 \cdot V_{dd} - 3528.43)/T)} \quad (4.27) \\ &= L \cdot T^2 \cdot e^{(K/T)} \quad (4.28) \end{aligned}$$

In the above model, $P_L$ represents the leakage power. $N_{gate}$ is the total number of gates within a certain chip area under consideration. $V_{dd}$ is the supply voltage. $T_0$ and $V_0$ are reference temperature and voltage, respectively. $I_s(T_0, V_0)$ is the saturation current at $T_0$ and $V_0$. $L = N_{gate} \cdot V_{dd} \cdot I_s(T_0, V_0)$ is a design/technology dependent constant. $K = 614.98 \cdot V_{dd} - 3528.43$ is also a technology-dependent constant for a fixed supply voltage. As can be seen, the leakage power has a rather complex dependency on temperature. The overall effect is approximately an exponential increase in leakage as temperature rises. There are other models for leakage power as well. We found the model described above quite effective in capturing the leakage-temperature dependency and we will use it to explain our method. Note that our approach is general and can handle other nonlinear leakage-temperature models as well [72].

### 4.6.2 Extended Kalman Filter

Now let us take leakage power into the picture and derive the new system model (equations (4.29) to (4.32)). Note that $T[n]$ still depends on $T[n-1]$ and $P[n-1]$ (the temperature and power history). However power $P[n-1]$ has two components now: dynamic power $P_D[n-1]$ and leakage power $P_L[n-1]$. The latter is a nonlinear function in temperature.

$$\begin{aligned} T[n] &= AT[n-1] + BP[n-1] \quad (4.29) \\ &= AT[n-1] + B(P_D[n-1] + P_L[n-1]) \quad (4.30) \\ &= AT[n-1] + BP_D[n-1] + BL \cdot elem\left(T[n-1]^2 e^{K/T[n-1]}\right) \quad (4.31) \\ &= f(T[n-1]) + BP_D[n-1] \quad (4.32) \end{aligned}$$

where

$$f(T[n-1]) = AT[n-1] + BL \cdot elem\left(T[n-1]^2 e^{K/T[n-1]}\right) \tag{4.33}$$

For simplicity in notation we used $elem(\cdot)$ to denote the element-wise operations (the computation is carried out in an element-by-element fashion for the entire vector or matrix; for example, $elem([1\ \ 2\ \ 3]^2) = [1\ \ 4\ \ 9]$). This is because the leakage power generated at a certain grid location only depends on the local temperature at that location. In the above equations $f(T[n-1])$ is a nonlinear function. $P_D$ and $P_L$ represent the dynamic and leakage power respectively. Equation (4.31) is obtained by substituting (4.28) into (4.30).

Based on this new nonlinear system model, we should now update $T[n|n-1]$ (the projected/predicted temperature at time $n$) as follows:

$$T[n\big|n-1] = E(T[n] \mid T[n-1|n-1]) \tag{4.34}$$

$$= f(T[n-1\big|n-1]) + B\mu^{P_D} \tag{4.35}$$

where $\mu^{P_D}$ represents the average value of dynamic power. Since $T[n|n-1]$ is our prediction for $T[n]$ before any sensor observations are made at time $n$, the best guess we have is simply the expected value of $T[n]$ given the observations up to time $n-1$, hence the above equation. Note that (4.35) is derived based on the nonlinear model (4.32) and should now replace equation (4.4) in the standard Kalman filter to account for the leakage effect.

We have shown that computing $T[n|n-1]$ in the *predict* stage of Kalman filter is relatively easy. However, given a nonlinear system described in (4.32), computing the error covariance matrix $C[n\big|n-1]$ is much harder. Note that with a linear model (which is the case when ignoring the leakage), a Gaussian input (vector $T[n-1]$ and $P[n-1]$ in equation (4.2)) will always produce a Gaussian output ($T[n]$ in (4.2)). Thus $C[n\big|n-1]$ can be easily computed using equation (4.5). However this does not hold true any more due to the nonlinear function $f(\cdot)$ in equation (4.32) which makes computing the covariance matrix $C[n\big|n-1]$ much more difficult. This covariance matrix is critical because the Kalman gain (see equation (4.7)) and therefore the accuracy of the entire filter depends critically on the correct value of

it. Various linearization techniques can be used to help us approximate the value of this covariance matrix. Next we introduce two such schemes.

Extended Kalman filter is a popular approach that has been successfully applied to many nonlinear systems over the past many years. Here we introduce this technique to the realm of thermal tracking to address the nonlinear leakage effect. The fundamental concept of this method revolves the notion that, at each time step, the true temperature is sufficiently close to the estimated temperature. Therefore, we can perform a first-order Taylor expansion of the nonlinear term in equation (4.32) at each time step around the most recent temperature estimate. Based on this linearized model, the error covariance matrix $C[n\big|n-1]$ can be computed accordingly.

Notice that in equation (4.32), the nonlinear function $f(\cdot)$ is differentiable. Thus if we perform a Taylor expansion of the term $f(T[n-1])$ around a certain point $T_c$ (note that $T_c$ is a vector), we can approximate $T[n]$ as follows:

$$T[n] = f(T[n-1]) + BP_D[n-1] \tag{4.36}$$

$$\approx f(T_c) + f'\big|_{T_c}(T[n-1] - T_c) + BP_D[n-1] \tag{4.37}$$

$$= f(T_c) + A'_c(T[n-1] - T_c) + BP_D[n-1] \tag{4.38}$$

Here the matrix $A'_c = f'\big|_{T_c}$ is the Jacobian of $f$ evaluated at $T_c$. Both $A'_c$ and $f(T_c)$ are constants whose values are determined once $T_c$ is chosen. To ensure the accuracy of this approximation, $T_c$ should be chosen as close to $T[n-1]$ as possible. Note that the estimate $T[n-1\big|n-1]$ produced by the filter at time $n-1$ is the conditional expectation of $T[n-1]$ given all sensor observations up to time $n-1$; it is naturally our best guess (statistically) for $T[n-1]$ and should be sufficiently close to it in value as long as the filter is operating correctly. Thus, to approximate $C[n\big|n-1]$, we can choose $T_c = T[n-1|n-1]$ in equation (4.38) and then substitute

it into the definition of covariance matrix, as shown by the following equations:

$$
\begin{aligned}
& C[n\big|n-1] \\
=& E\left((T[n\big|n-1]-T[n])\cdot(T[n\big|n-1]-T[n])^T\right) && (4.39)\\
=& E\Bigg(\left(A'_{n-1}(T[n-1]-T[n-1|n-1])+B(P_D[n-1]-\mu^{P_D})\right) \\
& \cdot\left(A'_{n-1}(T[n-1]-T[n-1|n-1])+B(P_D[n-1]-\mu^{P_D})\right)^T\Bigg) && (4.40)\\
=& E\big(A'_{n-1}(T[n-1]-T[n-1|n-1]) \\
& \qquad \cdot(T[n-1]-T[n-1|n-1])^T A'^T_{n-1}\big) \\
& + E\left(B(P_D[n-1]-\mu^{P_D})(P_D[n-1]-\mu^{P_D})^T B^T\right) && (4.41)\\
=& A'_{n-1}C[n-1\big|n-1]A'^T_{n-1}+BQB^T && (4.42)
\end{aligned}
$$

where

$$A'_{n-1}=f'\big|_{T[n-1|n-1]} \tag{4.43}$$

Here equation (4.40) is obtained by substituting (4.35) and (4.38) (with $T_c = T[n-1|n-1]$) into (4.39). Equation (4.41) can be obtained by noting the fact that the randomness in estimation error ($T[n-1]-T[n-1|n-1]$) and the randomness in dynamic power ($P_D[n-1]-\mu^{P_D}$) are generally independent. The former is usually caused by sensor noise and modeling error. The latter is usually caused by workload variability. Thus the expectation of the cross products in equation (4.40) evaluates to zero, hence the equation (4.41). Finally (4.42) can be obtained using the definitions of $Q$ (covariance matrix of dynamic power) and $C[n-1\big|n-1]$ (see equation (4.44)):

$$
\begin{aligned}
& C[n-1\big|n-1] \\
=& E\left((T[n-1\big|n-1]-T[n-1])\cdot(T[n-1\big|n-1]-T[n-1])^T\right) && (4.44)
\end{aligned}
$$

The remaining parts (correct stage: equation (4.6) – (4.8)) of standard Kalman filter stay the same due to the fact that they handle the sensor observations only (which have nothing to do with leakage). Thus, to incorporate nonlinear effect of leakage power we simply need to replace the predict stage equations (4.4) and

(4.5) with the new equations (4.35) and (4.42). The extended Kalman filter can be summarized as follows:

---

*predict:*

$$T[n\big|n-1] = f(T[n-1\big|n-1]) + B\mu^{P_D} \tag{4.45}$$

$$C[n\big|n-1] = A'_{n-1}C[n-1\big|n-1]A'^{T}_{n-1} + BQB^T \tag{4.46}$$

*correct:*

$$T[n|n] = T[n|n-1] + K[n](S[n] - HT[n|n-1]) \tag{4.47}$$

$$K[n] = C[n|n-1]H^T(R + HC[n|n-1]H^T)^{-1} \tag{4.48}$$

$$C[n|n] = (I - K[n]H)C[n|n-1] \tag{4.49}$$

---

## 4.7 Experimental Results

In this section we demonstrate the effectiveness of our adaptive and leakage-aware Kalman filters. To show the impact of each approach, we first ignore the leakage power and report the accuracy improvement achieved by our adaptive methods alone. We then take leakage into consideration and compare the accuracy of (1) standard Kalman filter; (2) leakage-aware standard Kalman filter and (3) leakage-aware adaptive Kalman filter. The simulations setup used in these experiments are described below.

### 4.7.1 Autonomous and Adaptive Kalman Filter

**Simulation Settings:** The test processor we used in our simulation is a high performance aggressive out-of-order processor with pipeline width of 8 instructions and an instruction window of 128 instructions. Both instruction cache and data cache are 32KB 4-way set associative. All the caches in the hierarchy use LRU

replacement policy and have a block size of 64 bytes. The physical dimension of the chip is $10mm \times 10mm \times 0.5mm$. We used a simplified floorplan (see figure 4.2) and $16 \times 16$ grid granularity in all our following experiments.

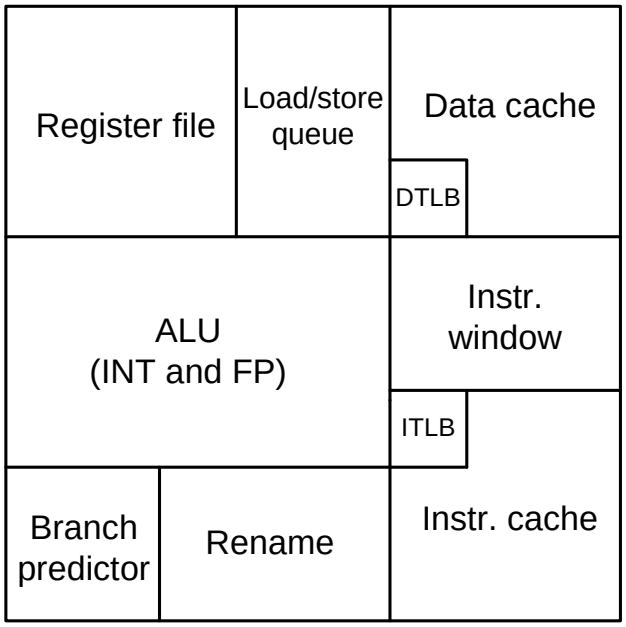



Figure 4.2: A simplified floorplan used in our experiments

As a basis for comparison, we first simulated the actual temperature trace of the test processor based on equation (4.2). In this equation the power consumption $P[n]$ at any given time $n$ only consists of the dynamic power component $P_D[n]$. We will ignore leakage power for the time being in order to demonstrate the advantage of our adaptive approach alone. To obtain the dynamic power trace of the test chip, we simulated the SPEC-2000 CPU benchmark suite using Wattch [6]. These benchmarks are scheduled in a random sequence and each benchmark is executed for a random time duration within range $0s \sim 60s$. The coefficient matrices $A$ and $B$ in equation (4.2) can be derived (see [63]) using the HotSpot thermal-RC model. The model parameters are in line with the values given in [53]: the thermal conductivity (inverse of thermal resistance) per unit volume is 100 $W/(m \cdot K)$. The thermal capacitance per unit volume is $1.75 \times 10^6$ $J/(m^3 \cdot K)$. For the overall effect of heat sink the thermal resistance is 0.026 $K/W$ and the thermal capacitance is 8.8 $J/K$. For the effect of convection the equivalent thermal resistance is 1.0 $K/W$ and the

thermal capacitance is 140.4 $J/K$. Now given equation (4.2), coefficient matrices $A$ & $B$ and the dynamic power trace ($P[n]$, for $\forall n$), we simulated the dynamic thermal profile of the test processor for a duration of 1500 seconds (at startup, the chip is assumed to have the ambient temperature of 25°C). This simulated thermal trace is assumed to be the real chip temperature and is used to measure the estimation accuracy.

For thermal tracking, we assumed that 5 sensors are uniformly scattered on the chip. The matrix $H$ in equation (4.3) can be easily determined based on sensor placement. These sensors can report the actual temperature (generated from the above simulation) of the grid cells that they reside in. In reality sensors are always affected by various types of noise such as supply voltage fluctuation, fabrication variability, cross coupling and etc. To reflect this reality, we superimposed 5% Gaussian random noise onto sensor readings. Given such noise-corrupted observations from the five sensors (sampled at 1 second intervals), our goal is to estimate the entire thermal profile of the chip as accurately as possible.

To compare our adaptive filter with the standard Kalman filter, we extracted the statistical power characteristics $(\mu_k^p, Q_k)$ for each benchmark $k$. Each pair $(\mu_k^p, Q_k)$ essentially represents a potential power state (hypothesis) that our system could be in. To obtain this information, we simply simulated each benchmark separately for a representative $250M$ instructions to obtain its runtime dynamic power trace. We then superimposed 5% variation onto such simulated power data to reflect the runtime uncertainties such as supply voltage fluctuation, environmental randomness and etc. Based on this data, parameters $\mu_k^p$ and $Q_k$ can be easily computed from the sample mean and sample covariance. The statistical power information for all benchmarks form a base set that represents all potential system power states. Each state is assumed to have the same prior probability. Given this information as well as the runtime sensor input, our adaptive filter is capable of autonomously detecting any underlying power state change and automatically adjust the filter parameters to adapt to such changes. Awareness of the varying system power state, and furthermore adapting to each new state in an online fashion can

prove to be very beneficial in improving the thermal tracking accuracy. For the traditional non-adaptive Kalman filter, we simply used the average $(\mu^p_{avg}, Q_{avg})$ across all benchmarks to operate the filter. All other inputs to these two different filters are the same. The tracking results for the ALU unit are reported in figures 4.3 to 4.6.

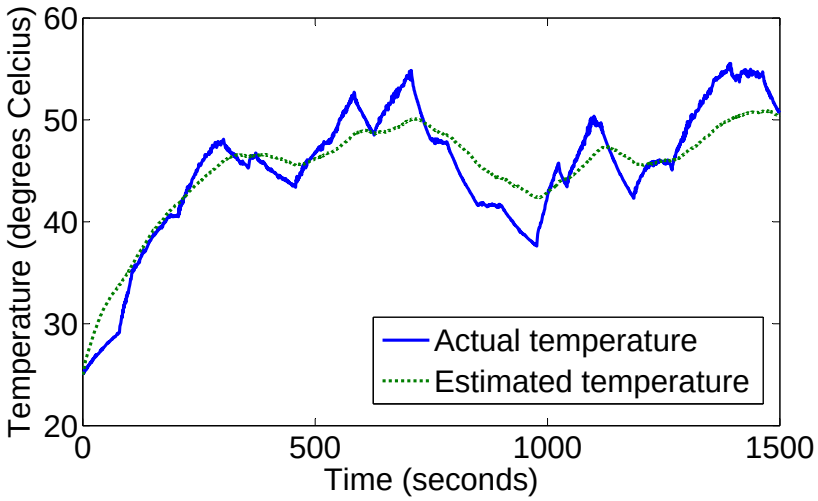



Figure 4.3: Actual vs estimated temperature using standard Kalman filter

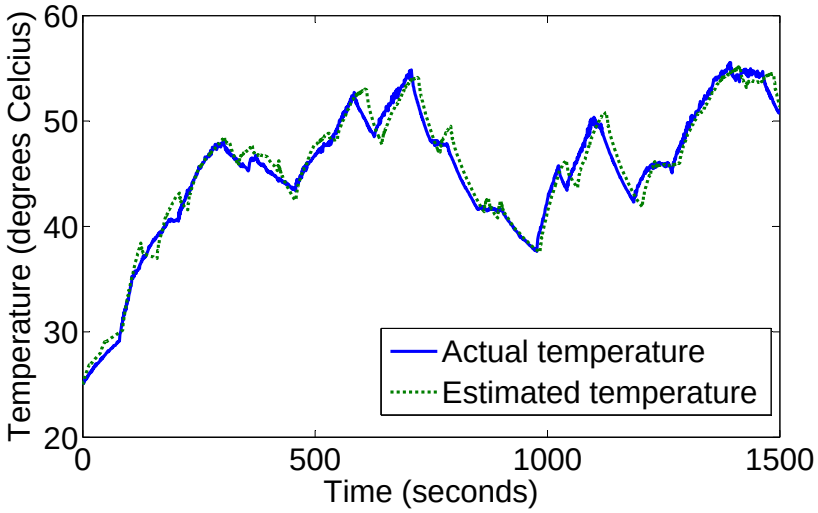



Figure 4.4: Actual vs estimated temperature using hypothesis testing (multi-sample)

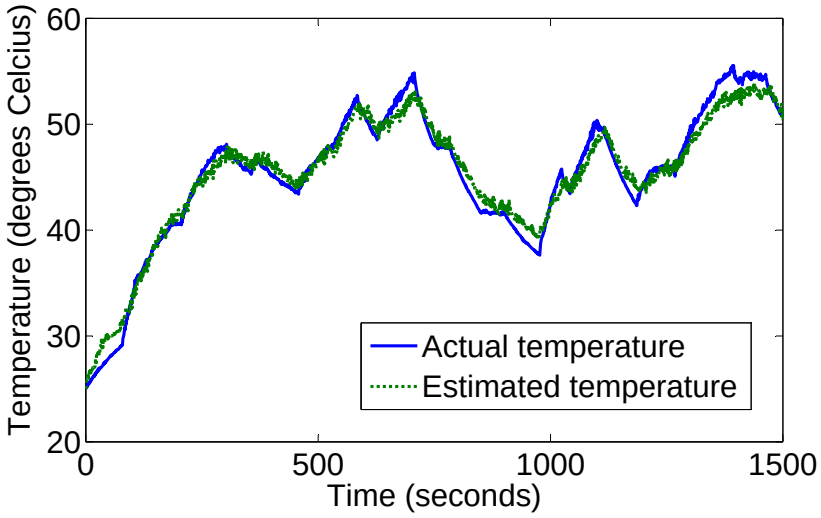



Figure 4.5: Actual vs estimated temperature using hypothesis testing (single sample)

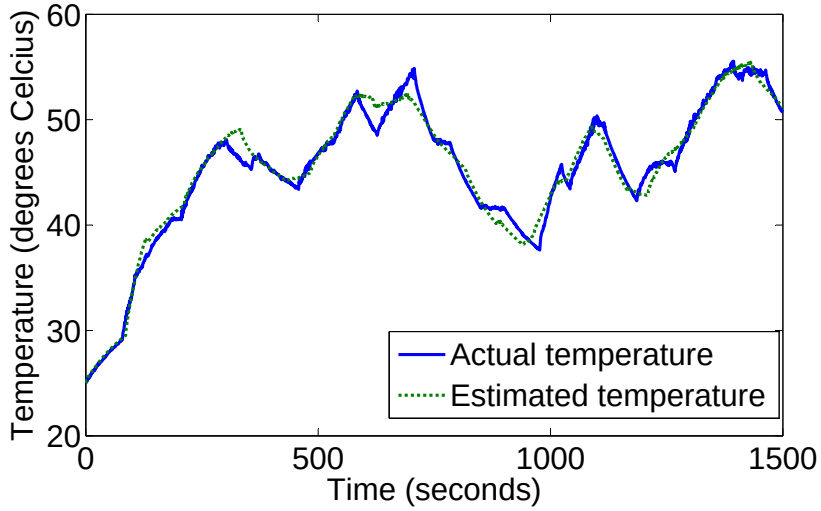



Figure 4.6: Actual vs estimated temperature using residual whitening

It can be seen from these figures that our adaptive approaches can predict and adapt to the system power state at runtime and can always generate highly accurate thermal estimates compared to the traditional non-adaptive Kalman filter. The average RMS error across the chip for the entire 1500 second duration are compared in table 4.1. Note that this is average error, the worst-case error for non-adaptive KF can be as high as 8.29°C which would require significant safety

guard-band from the DTM unit. On the other hand our adaptive filter produced a much better worst-case error of 3.05°C.

Table 4.1: Average RMS error for standard Kalman filter and adaptive filters

| Settings | Average RMS error (°C) |
|---|---|
| Standard Kalman filter | 2.66 |
| Hypothesis test (single sample) | 1.33 |
| Hypothesis test (multi-sample) | 1.28 |
| Residual whitening | 1.09 |

### 4.7.2 Leakage-aware Adaptive Kalman Filter

The experimental settings used for testing the leakage-aware adaptive Kalman filter is largely the same as those described in section 4.7.1, with the only exception being the inclusion of the leakage power component. For this purpose we used the system model shown in equation (4.30) to obtain the real temperature of the chip. In this equation the power consumption $P$ has two component $P_D$ and $P_L$. The former is still generated by simulating SPEC-2000 benchmarks using Wattch. For leakage $P_L$, we use the nonlinear leakage-temperature model as described in [22] (see equations (4.27) and (4.28)). The supply voltage $V_{dd}$ is assumed to be 1 $V$ to determine the parameter $K$. Parameter $L$ is scaled such that on average, leakage power consists of around 40% of the total power consumption. We applied our leakage-aware Kalman filter as described in section 4.6. In figures 4.7 to 4.11 we compare the thermal tracking results of the following four different settings: (1) non-adaptive standard Kalman filter with no leakage consideration; (2) adaptive Kalman filter with no leakage consideration; (3) leakage-aware non-adaptive Kalman filter and (4) leakage-aware adaptive Kalman filter.

It can be seen from figures 4.7 to 4.11 that after the inclusion of leakage power, the chip temperature increased about 15°C. The traditional Kalman filter based estimations schemes do not consider the nonlinear leakage-temperature dependency

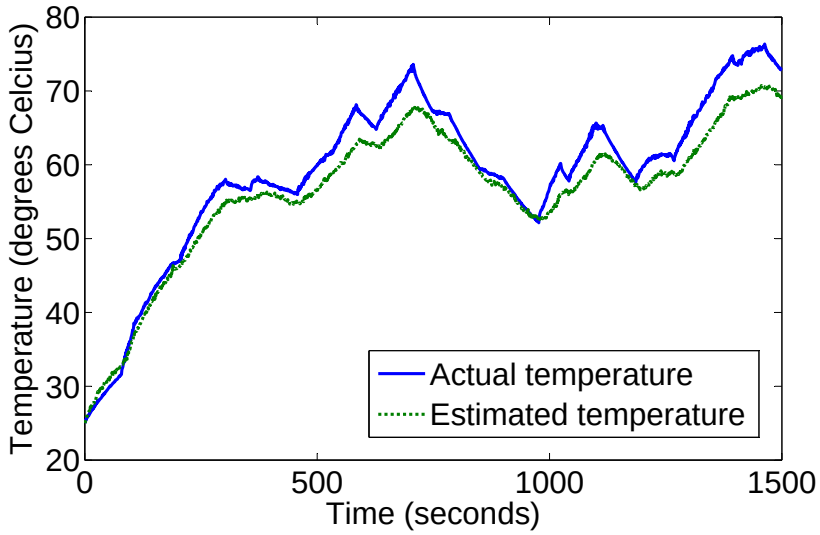



Figure 4.7: Actual vs estimated temperature using standard Kalman filter (ignoring leakage)

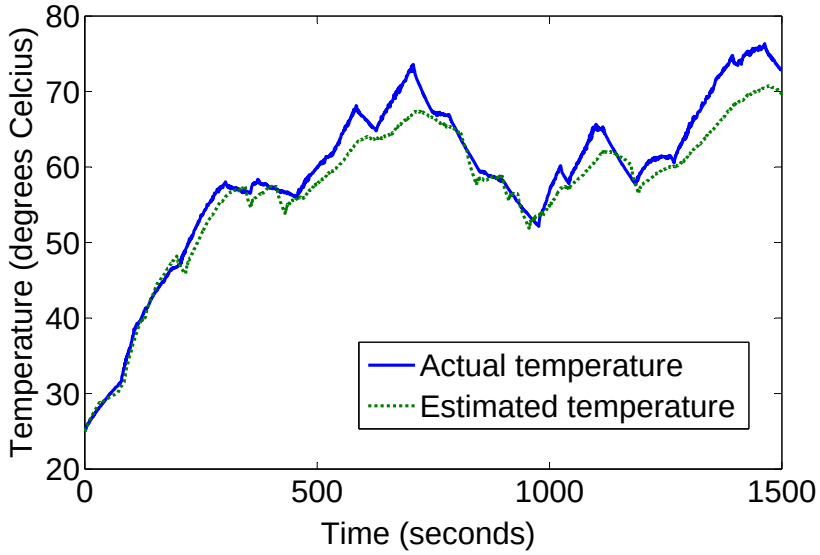



Figure 4.8: Actual vs estimated temperature using adaptive Kalman filter (hypothesis test with multi-sample) but ignoring leakage

and therefore tends to under-estimate the true chip temperature. With the extended leakage-aware adaptive Kalman filter we can achieve a much higher thermal tracking accuracy. The average RMS error across the chip for each setting is compared in

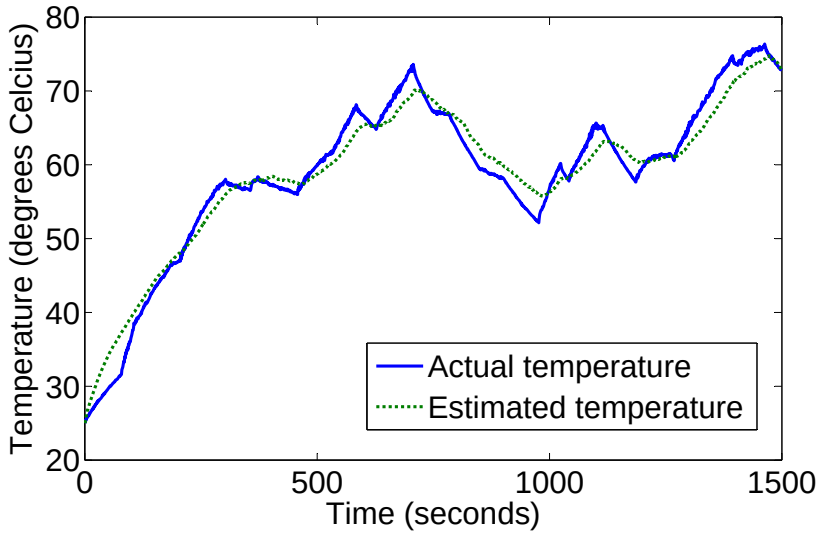



Figure 4.9: Actual vs estimated temperature using leakage-aware nonadaptive Kalman filter

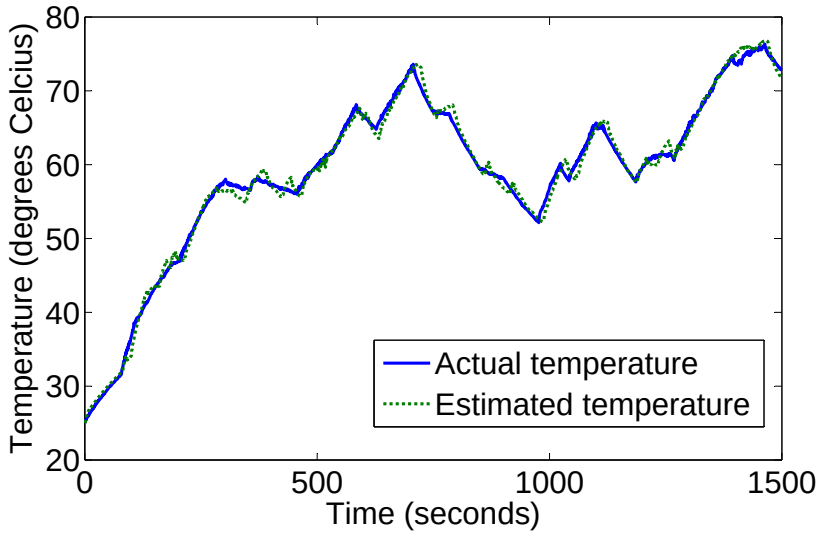



Figure 4.10: Actual vs estimated temperature using leakage-aware adaptive Kalman filter (hypothesis test with multi-sample)

table 4.2. The worst-case estimation error of the standard KF is 10.23°C whereas the worst-case error of our leakage-aware adaptive filter is only 3.27°C. It can be seen the improvement is more significant once we include the leakage effect. This

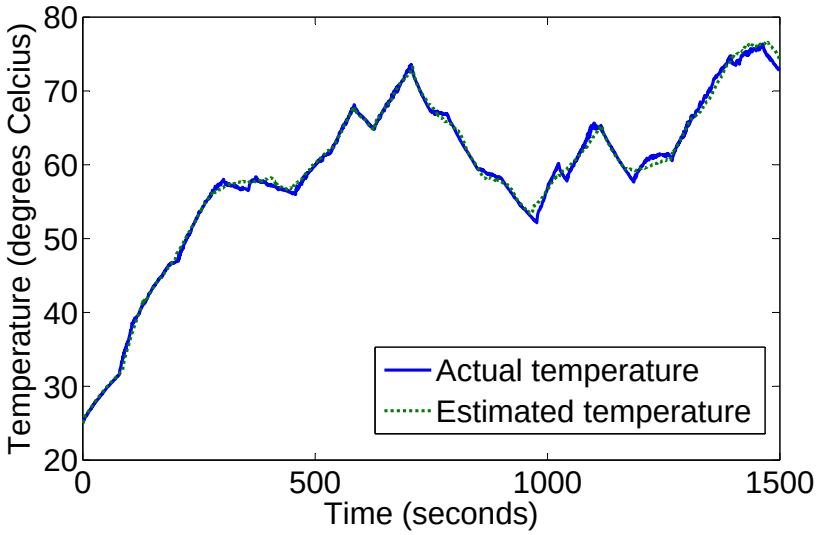



Figure 4.11: Actual vs estimated temperature using leakage-aware adaptive Kalman filter (residual whitening)

clearly demonstrate the effectiveness of our methods.

Table 4.2: Average RMS error comparison for different combinations of filters (HT: hypothesis testing; RW: residual whitening)

| Settings | Average RMS error (°C) |
|---|---|
| Traditional standard Kalman filter | 3.09 |
| Leakage-unaware adaptive KF | 2.73 |
| Leakage-aware non-adaptive KF | 2.56 |
| Leakage-aware adaptive KF (using HT) | 1.24 |
| Leakage-aware adaptive KF (using RW) | 1.15 |

# Chapter 5

# Statistical Characterization of Chip Power Behavior

## 5.1 Motivation

In earlier chapters, we explained why thermal sensing and estimation are necessary for efficient thermal control schemes. However, most of existing temperature estimation approaches depend heavily on the availability of the statistical power characteristics. Such required statistical information includes the mean, variance and correlation of the power consumption in different chip modules. In most of the previously proposed estimation schemes, such statistical power information is assumed to be known ([50, 70, 74, 24]). Many have argued that such information can be obtained through simulations or experimenting with benchmarks. However, to the best of our knowledge, no systematic approach has been proposed so far for modeling and extracting the statistical characteristics of the system power behavior. We propose an methodology in this chapter to fill such a gap. Moreover, the extracted statistical power information is not limited to only this application. It can also be used to provide guidelines to many other design considerations. For example, such information can be helpful in computing the expected life span of the battery life for embedded systems, or to estimate the severity and probability of the appearance of on-chip hot spots, just to name a few.

In this chapter we propose two approaches for extracting the statistical power characteristics using realistic workloads at a post-fabrication testing stage [73]. To extract the statistical model of the chip power behavior we need some way of observing the system power consumption. Our approach is based on taking infra-red thermal pictures of the chip as it is running realistic workloads with dynamic power management unit in effect. The feasibility of using infra-red camera to monitor the temperature map of the chip at a test stage has been demonstrated and verified in several works [39, 26]. Other ways of capturing the thermal profile of the chip are

also possible and can be used as an input for our learning algorithms. In our first approach we present an analytical solution for the case where the system statistical characteristics stay relatively stable (for example data center servers) and hence can be represented by a single Gaussian distribution. Our second approach is inspired from machine learning techniques and is based on the Expectation-Maximization (EM) algorithm. This algorithm can be used to characterize systems that are running highly heterogenous applications (for example desktop computers) and hence are best represented by a mixture Gaussian model. To motivate and provide a background for our approach, we will use the popular Kalman filter based temperature tracking framework as an example to demonstrate: (1) why obtaining the statistical power information is important and (2) how such information can be used in runtime power and temperature estimation algorithms. Kalman filter based estimation scheme is very popular due to its optimality and convergence properties [50, 24, 70, 73]. By exploiting the statistical power characteristics of a chip obtained using our methods, a Kalman filter based approach is capable of generating accurate thermal and power estimates for all chip modules based on limited and possibly noise-corrupted sensor observations, thus enabling (or improving the performance of) many sophisticated runtime power and temperature optimization schemes.

## 5.2 Problem Definition and Challenges

### 5.2.1 Joint Temperature/Power Estimation

This chapter will again use some of the preliminary knowledge of thermal RC modeling and Kalman Filter. Please refer to section 4.2 for details. The thermal estimate for $T[n]$ generated using sensors and a-priori knowledge of the statistical characteristics of $P[n]$ could be used to manage the hotspots and improve the overall reliability of the system. On another front we would also be interested in knowing the runtime power dissipation profile $P[n]$ and tracking the same. Using system thermal model which relates temperature and power, as well as the KF generated thermal estimates of $T[n]$, we can estimate the runtime power profile by solving the

following equation:

$$T[n|n] = AT[n-1|n-1] + BP[n-1] \tag{5.1}$$

Using the current and previous thermal profile estimate $T[n|n], T[n-1|n-1]$ (which was generated using thermal sensor measurements as discussed earlier) we can estimate $P[n-1]$. The chip power profile $P[n-1]$ is an important parameter we would like to estimate during runtime. Knowledge about which parts of the chip is dissipating how much power could be used by the OS to make more informed decisions about the scheduling of tasks, dynamic thermal and power management policies etc. Unusual distribution of power across the chip which is significantly different that expected could also be used to detect the presence of trojan horses. Thermal sensors and the underlying KF theory could be used to estimate both the current thermal and power state of the chip.

### 5.2.2 Modeling the Random Power Behavior

Much of the sensor based approaches assume a-priori knowledge of statistical characteristics of the power dissipation profile $P[n]$ to perform estimation. If the underlying characteristics are known, a KF [50] or adaptive KF [70] based approach has been shown to perform reasonable well for estimating the dynamic chip thermal profile $T[n]$ (and even the power profile $P[n]$). However, in reality, it is extremely tough to obtain the statistical characteristics of $P[n]$ a-priori. The fidelity of thermal and power tracking critically depends on how good our a-priori estimates of $P[n]$ statistical characteristics are. In the context of KF based thermal/power tracking, knowing the $\mu^p$ and $Q$ is sufficient. In the context of adaptive KF based tracking which supports dynamic changes in the $P[n]$ statistical characteristics, knowing the mixture Gaussian model parameters $\{(\mu_0^p, Q_0), (\mu_1^p, Q_1), \ldots, (\mu_K^p, Q_K)\}$ is sufficient. The problem is that this a-priori knowledge is extremely hard to generate since it is highly dependent on the runtime characteristics of the application, data etc.

In this thesis we present a learning algorithm for generating the required information pertaining to the statistical characteristics of $P[n]$ which could then be

used by either a KF or adaptive KF based approach. We assume that at the post-fabrication characterization stage, the infra-red camera can provide thermal pictures (i.e. the complete temperature vector $T[n]$) of the test chip at fixed time intervals. In formal mathematical terms, our methods seek to learn the following parameters of the underlying statistical power model from the thermal pictures of the chip as it is running typical workloads.

$$\Theta_K = \begin{cases} BGD_1 & \sim \mathcal{N}(\mu_1^p, Q_1): \quad \pi_1 \\ BGD_2 & \sim \mathcal{N}(\mu_2^p, Q_2): \quad \pi_2 \\ \cdots & \\ BGD_K & \sim \mathcal{N}(\mu_K^p, Q_K): \quad \pi_K \end{cases} \tag{5.2}$$

Here K is the number of potential statistical power states (i.e. base Gaussian distributions, BGD) the system could be in. Also $\mu_i^p, Q_i$ is the mean and co-variance of the $BGD_i$. Conceptually each of these different BGDs could represents the power behavior of a cluster of similar application.The parameter $\pi_i$ represents the probability of system being in the $i-th$ BGD at a given time. Thus $\sum_{i=1}^{K} \pi_i = 1$ since the system must be in one of the BGDs. If the system are relatively stable and its workloads are homogeneous then one BGD is enough which means $K = 1$. Once the appropriate parameters for the model represented in equation (5.2) have been learned, they could be provided as input to any statistical estimation approaches for thermal/power estimation.

### 5.2.3 Problem Formulation

As mentioned earlier the system we are working with can be described as follows:

$$T[n] = AT[n-1] + BP[n-1] \tag{5.3}$$

$$S[n] = T[n] + v[n] \tag{5.4}$$

Note that we know the complete observation vector $S[n]$ (from infra-red thermal pictures), the dimension of $S[n]$ is the same as $T[n]$. Equations (5.3) and (5.4)

describe the system dynamics and the observation vector. $v[n]$ represents the potential error in the infra-red pictures. Such noise characteristics do not depend on the workloads and are therefore relatively stable. We assume the camera noise vector $v[n]$ follows a Gaussian distribution with known mean $\mu^S$ and known covariance matrix $R$. Now given such mean and variance information, we would like to dynamically learn the power characteristics of the system, *i.e.* the mixture Gaussian model illustrated in equation (5.2), using the observed thermal pictures (*i.e.* $S[n]$ for all $n$). It can be expected that the optimal set of model parameters would be the ones that best fit the observed data. Thus this problem can be formulated in a optimization framework in which the target function we want to maximize is the probability of getting the set of observation vectors ($S[0]$ to $S[n]$) given the underlying mixture Gaussian model. The variables that need to be determined are the statistical parameters of this model presented in equation (5.2). The problem can be formulated mathematically as follows:

$$\text{maximize: } F = prob(S[n], S[n-1], \ldots, S[1] \big| S[0], \Theta_K) \tag{5.5}$$

$$\text{variables: } \Theta_K = \{\mu_1^p, \ldots, \mu_K^p, Q_1, \ldots, Q_K, \pi_1, \ldots, \pi_K\} \tag{5.6}$$

$$\text{subject to: } \sum_j \pi_j = 1 \tag{5.7}$$

where $\Theta_K$ represents the set of parameters of the $K$ overlapping BGDs in our mixture Gaussian model. The set of temperature observations is represented by $\{S[0], \ldots, S[n]\}$ where $S[0]$ is our initial observation (when we just started taking thermal pictures) and $S[n]$ is the sensor observations at time $n$. Since one of the distribution must always be true we have $\sum_j \pi_j = 1$. There is no other constraint on the parameters of $\Theta_K$ except the fact that the mean power dissipation vectors should be positive. One important point to note is that different model parameters would result in different overall probability $F$ because the temperature change are mainly caused by the underlying power behavior. In other words thermal maps are the observed surface of the system and the power behavior is the underlying driving force behind the surface phenomenon. Once we maximized the target function $F$ by adjusting the model parameters we would have captured an optimal model for the

underlying statistical power characteristics. This relationship between observation vectors and the underlying power model will become more obvious once we make the following transformations to $F$:

$$F = prob(S[n], S[n-1], \ldots, S[1] \big| S[0], \Theta_K) \tag{5.8}$$

$$= prob(S[n] \big| S[n-1], \Theta_K) \times prob(S[n-1] \big| S[n-2], \Theta_K) \times \ldots \times prob(S[1] \big| S[0], \Theta_K) \tag{5.9}$$

$$= prob(D[n] \big| S[n-1], \Theta_K) \times prob(D[n-1] \big| S[n-2], \Theta_K) \times \ldots \times prob(D[1] \big| S[0], \Theta_K) \tag{5.10}$$

where

$$D[n] = S[n] - AS[n-1] \tag{5.11}$$

$$= T[n] + v[n] - A(T[n-1] + v[n-1])$$

$$= AT[n-1] + BP[n-1] + v[n] - A(T[n-1] + v[n-1])$$

$$= BP[n-1] + v[n] - Av[n-1] \tag{5.12}$$

We can transform equation (5.8) to (5.9) because of the markovian nature of the system dynamics in equation (5.3). Essentially $S[n]$ depends only on $S[n-1]$ and the model $\Theta_K$. Next we transform equation (5.9) to (5.10). This is because it can easily be shown that $prob(S[n]|S[n-1], \Theta_K) = prob(D[n]|S[n-1], \Theta_K)$. This is because there is a one to one mapping between $D[n]$ and $S[n]$ for a given $S[n-1]$. After the above transformation, the target function $F$ is described by the probability of getting $\{D[1], \ldots, D[n]\}$ given $\{S[0], S[1], \ldots, S[n]\}$ and $\Theta_K$. Equation (5.12) is obtained by plugging equations (5.3) and (5.4) into (5.11). It can be seen in (5.12) that $D[n]$ is a linear combination of Gaussian random variables. Hence $D[n]$ is Gaussian as well. Now based on the linear relationship shown in (5.12), if we are given any mixture model $\Theta_K$ for $P[n]$ as described by (5.2) we can easily find out the corresponding mixture model for $D$:

$$\Theta_K^D = \begin{cases} BGD_1 & \sim \mathcal{N}(B\mu_1^p, BQ_1B^T + R + ARA^T): \quad \pi_1 \\ BGD_2 & \sim \mathcal{N}(B\mu_2^p, BQ_2B^T + R + ARA^T): \quad \pi_2 \\ \dots & \\ BGD_K & \sim \mathcal{N}(B\mu_K^p, BQ_KB^T + R + ARA^T): \quad \pi_K \end{cases} \tag{5.13}$$

For any model parameters $\Theta_K$ we know the corresponding $\Theta_K^D$ as shown above which describes the probabilistic distribution for $D$. Based on $\Theta_K^D$ we can easily compute the target function $F$ based on (5.10). Thus the optimal power model can be obtained by maximizing the new target function (5.10). As would become clear subsequently, this transformation is essentially for seamless optimization of our objective function. We rewrite this objective for the sake of clarity.

$$\begin{aligned} \text{maximize: } & prob(D[n]\big|S[n-1], \Theta_K) \times prob(D[n-1]\big|S[n-2], \Theta_K) \\ & \times \dots \times prob(D[1]\big|S[0], \Theta_K)) \end{aligned} \tag{5.14}$$

## 5.3 Power Characterization

In this section we present techniques for automatically learning the $\Theta_K$ from thermal pictures. We present two techniques which are applicable in two different scenarios. The first one deals with the case where the statistical characteristics of $P[n]$ do not change very often. Under this scenario, the underlying power dissipation profile and therefore the thermal observations are generated from a single BGD associated with $P[n]$. Basically $K = 1$ in $\Theta_K$. This scenario is suitable for cases where the application class does not change too often (data centers for example). The other scenario represents the case where the statistical characteristics of $P[n]$ change quickly. For example, in a general purpose desktop processor, the characteristics of $P[n]$ would change quickly if heterogeneous application are scheduled at a fast pace. Hence $\Theta_K$ would represent the set of $K$ BGDs that $P[n]$ could be in along with the associated prior probability of each BGD. The observation vector $S[n]$ collected over a large period of time would represent these different statistical

power states and the problem is to learn the parameters of $\Theta_K$ from this data. The next two subsections develop methods for learning the power characteristics in these two distinct situations.

### 5.3.1 Single BGD Characterization

In this subsection we discuss the situation where the power statistical characteristics stay relatively stable. Note that the system power can still vary from time to time but its mean and variance $\mu^P$ and $Q$ stays the same (which is what we wish to estimate from the thermal pictures). In other words the mixture Gaussian model $\Theta_K$ is reduced to a simple Gaussian model where only one BGD is present (essentially $K = 1$). In such a scenario, we could derive the model parameters analytically. As explained earlier in section 5.2 (equations (5.11) to (5.12)), once we have the observation vectors $S[0]$ to $S[n]$ we can easily compute the corresponding $D[1]$ to $D[n]$ as follows:

$$\begin{aligned} D[i] &= S[i] - AS[i-1] \qquad \text{for } i = 1 \text{ to } n \\ &= BP[i-1] + v[i] - Av[i-1] \end{aligned} \tag{5.15}$$

Based on (5.15) and the fact that $P[i-1], v[i-1]$ and $v[i]$ are independent (camera noise is assumed to be independent of the chip power dissipation), the statistical characteristics of $D$ are related to the statistical characteristics of $P[n]$ in the following ways:

$$\mu^D = B\mu^P + (I - A)\mu^S \tag{5.16}$$

$$\Sigma^D = BQB^T + R + ARA^T \tag{5.17}$$

Since we know the noise characteristics $\mu^S$ and $R$, if we know $\mu^D$ and $\Sigma^D$, we can analytically compute $\mu^P$ and $Q$. Using equation (5.15), consecutive observation vectors $S[i]$ and $S[i-1]$ can be used to compute samples of $D[i]$. Sample of $S[i]$ and therefore samples of $D[i]$ collected over a period of time could then be used to

estimate the sample mean $\mu^D$ and sample covariance$\Sigma^D$ as follows.

$$\mu^D = \frac{1}{n}\sum_{i=1}^{n} D[i] \tag{5.18}$$

$$\Sigma^D = \frac{1}{n-1}\sum_{i=1}^{n}(D[i]-\mu^D)(D[i]-\mu^D)^T \tag{5.19}$$

This estimate can then be used to estimate $\mu^P$ and $Q$ as follows.

$$\mu^P = B^{-1}(\mu^D - (I-A)\mu^S) \tag{5.20}$$

$$Q^P = B^{-1}(\Sigma^D - R - ARA^T)(B^T)^{-1} \tag{5.21}$$

### 5.3.2 Multiple BGDs Characterization

In this section we propose a heuristic algorithm for learning the parameters of $\Theta_K$ from the thermal observations when the statistical characteristics of $P[n]$ are from several overlapping BGDs. In this case the observation vectors are caused by multiple statistical power states in the mixture Gaussian model. As indicated earlier $\Theta_K$ represents the set of possible statistical power states that the system switches between. Once $\Theta_K$ is known, methods such as [70] could be used to jointly track the power and temperature. The heuristic is inspired from the fields of machine learning and artificial intelligence.

Let us first assume that we know the number of BGDs $K$ in such a mixture model $\Theta_K$. Usually by looking at the typical workloads information we know how many different BGDs are needed to model the power behavior of the system. If $K$ is not known we can simply try $K = 1, 2, 3, \ldots$ until satisfactory accuracy is reached (i.e. target function $F$ does not change much as $K$ is further increased). Now assuming $K$ is fixed our goal is to determine the set of model parameters based on the thermal map observations. Our algorithm is an Expectation-Maximization (EM) algorithm which is a popular algorithm used in machine learning [57]. At the beginning of the algorithm, all model parameters (see equation (5.2)) are initialized (in a systematic way). The convergence rate of our algorithm is a function of this initialization. Usually one method of initialization could be to uniformly split the

potential range of power uniformly into $K$ slots and assign each to a specific BGD. Let us suppose we have collected $n+1$ observation vectors. Then the algorithm iteratively carries out the following two steps until convergence:

*Expectation:*

$$r_{ij} = prob(BGD_j \big| D[i]) \tag{5.22}$$

$$= \frac{prob(D[i] \big| BGD_j) \cdot \pi_j}{\sum_{m=1}^{K} prob(D[i] \big| BGD_m) \cdot \pi_m} \tag{5.23}$$

*Maximization:*

$$\mu_j^D = \frac{1}{\sum_{i=1}^{n} r_{ij}} \sum_{i=1}^{n} r_{ij} D[i] \tag{5.24}$$

$$\Sigma_j^D = \frac{1}{\sum_{i=1}^{n} r_{ij}} \sum_{i=1}^{n} r_{ij} (D[i] - \mu_j^P)(D[i] - \mu_j^P)^T \tag{5.25}$$

$$\mu_j^P = \mu_j^D - \mu^S \tag{5.26}$$

$$Q_j^P = B^{-1}(\Sigma_j^D - R - ARA^T)(B^T)^{-1} \tag{5.27}$$

$$\pi_j = \frac{1}{n} \sum_{i=1}^{n} r_{ij} \tag{5.28}$$

**Expectation Step:** In equation (5.23), the parameter $r_{ij}$ represents the probability that the system is in statistical state represented by $BGD_j$ given the $i$-th observation $D[i]$. Note that using the observation vectors $S[i]$ and $S[i-1]$, $D[i]$ can be computed easily (see equation (5.15)). Our algorithm is an iterative process where we start with some parametric assumption on the values of $\Theta_K$. These values are then refined to fit the observed thermal data. To compute $r_{ij}$ for all observations $D[i]$ in the current iteration we use equation (5.23) which simply is an application of Bayes theorem. The parameters of $\Theta_K$ in the current iteration can be used to compute $prob(D[i] \big| BGD_j)$ for all the BGDs. This computation is performed as follows. For $BGD_j$ with parameters $\mu_j^P$ and $Q_j^P$ for the current iteration are known. Then using equations (5.16) and (5.17), we can obtain the mean and variance of the parameter $D$ assuming $BDG_j$ is the underlying power statistic. From here, we can compute $prob(D[i] \big| BGD_j)$. Hence, using the $\Theta_K$ parameters in the current iteration we can compute $r_{ij}$ for all data samples $D[i]$.

**Maximization Step:** In this step, the estimated $r_{ij}$ values are used to refine the $\Theta_K$ parameters to fit the observed data. Let us suppose we have collected $n+1$ samples of $S[i]$ and therefore have $n$ samples for $D[i]$. Equation (5.24) computes $\mu_j^D$ which represents the mean of $D$ assuming $BGD_j$ is the underlying power state. Basically $\sum_{i=1}^{n} r_{ij}$ represents the total probability that all $D[i]$ samples belong to $BGD_j$. Similarly $\Sigma_j^D$ in equation (5.25) represents the associated covariance matrix. This is computed for all BGDs in $\Theta_K$. Using this information and equations (5.20) and (5.21) we can refine $\mu_j^P$ and $Q_j^P$. This is represented in equations (5.26) and (5.27). Finally equation (5.28) refines of the prior probabilities of BGDs. These refined estimates are once again improved in the next iteration of the expectation and maximization loops.

This process is repeated until an acceptable convergence criterion is met. Usually if the change in the $\Theta_K$ parameters is small then further iterations are not needed.

### 5.3.3 Overall Framework and Computational Complexity

**$K = 1$:** When the power characteristics do not change too much then it is reasonable to assume $K = 1$. A KF based thermal/power tracking approach as presented in [50, 24] needs to know $\mu^P$ and $Q^P$. Such information could be generated using our approach presented in section 5.3.1. The number of data samples directly controls the complexity of our approach. More data samples implies more accurate characterization but higher complexity as well. The error in estimating the parameters of a Gaussian distribution usually decreases at the rate of $1/n$ where $n$ is the number of samples. In our results we experimented with about 500 data samples and observed an error within 3%.

**$K > 1$:** In this case, the system can run highly heterogeneous workloads. Basically, $P[n]$ follows a particular BGD for a while and then changes to another BGD at a fast pace (which depends on how often different applications are switched). The approach presented in [70] develops adaptive techniques for thermal tracking when the underlying statistical power characteristics follow a mixture Gaussian model. It

assumes a library of choices (basically $\Theta_K$) is available. Our methods can generate $\Theta_K$ using based on the observed thermal data. The algorithm presented above is an iterative process that starts from an initial estimation of $\Theta_K$ which is then refined. The complexity of the algorithm is a function of the number of observation vectors and also the number of iterations needed. For 500 data samples, we needed only 5 iterations to converge.

Overall, there is certainly some computational overhead imposed by the algorithms that automatically learn $\Theta_K$. This complexity is acceptable since the characterization only needs to be performed once at a post-fabrication stage. The complexity and accuracy of our approach can be controlled by the number of observation samples used.

## 5.4 Experimental Results

Table 5.1: Estimation error (unit: $W$) of our methods for various benchmarks.

| | apsi/eon | bzip2/crafty | art/crafty | art/apsi | bzip2/fma3d | applu/eon |
|---|---|---|---|---|---|---|
| avg absolute error in $\mu^P$ (fast) | 0.11 | 0.06 | 0.34 | 0.05 | 0.88 | 0.05 |
| avg absolute error in $Q$ (fast) | 0.07 | 0.98 | 1.19 | 0.10 | 1.27 | 0.06 |
| | applu | bzip2 | crafty | art | apsi | eon |
| avg absolute error in $\mu^P$ (slow) | 0.06 | 0.03 | 0.04 | 0.04 | 0.03 | 0.07 |
| avg absolute error in $Q$ (slow) | 0.06 | 0.07 | 0.11 | 0.05 | 0.23 | 0.28 |

Table 5.2: Comparison between the actual $\mu^P$ and the estimated $\mu^P$ (unit $W$) for the APSI benchmark.

| | Initialization | 1st iter. | 2nd iter. | 3rd iter. | 4th iter | 5th iter. | actual $\mu^P$ |
|---|---|---|---|---|---|---|---|
| rename | 0.56 | 0.84 | 0.82 | 0.81 | 0.80 | 0.79 | 0.78 |
| instr. cache | 1.38 | 2.52 | 2.46 | 2.41 | 2.36 | 2.35 | 2.32 |
| ALU | 2.72 | 5.28 | 5.23 | 5.19 | 5.14 | 5.14 | 5.03 |
| runtime (s) | 0.02 | 1.54 | 2.04 | 2.06 | 2.05 | 2.08 | – |

To carry out our experiments, we used Wattch with Alpha binary [6] to generate the power consumption data for each functional unit. We configured a high

performance aggressive processor with pipeline width of 8 instructions and an instruction window of 128 instructions. Both instruction cache and data cache are 32KB 4-way set associative. All caches are using LRU replacement policy and a block size of 64 bytes. The physical dimension of the chip is $15mm$(length) $\times$ $15mm$(width) $\times$ $0.5mm$(thickness). For benchmarks, we simulated all the SPEC 2000 CPU benchmark suite compiled with the default parameters provided with the suite. The power consumption data obtained through simulation for each benchmark are then superimposed 30% variation to represent the collective effect of supply voltage fluctuation and the impact of runtime DPM actions, etc. Such random power consumption data generated for each benchmark is used to compute the actual statistical parameters (mean and variance) for each underlying BGD in our mixture Gaussian model $\Theta_K$. These parameters are used as the foundation for accuracy comparison in our experiments. Our goal is to learn the model parameters of $\Theta_K$ based on only the infra-red pictures observed at a post-fabrication testing stage. To obtain the dynamic temperature change of the chip, we used the popular thermal RC model described in HotSpot [53]. The model parameters we used are similar to the values shown in [53].

Our experiments are carried out as follow: 1) we randomly executed the two arbitrarily chosen benchmarks (from the SPEC 2000 pool) in an interleaved fashion for a total of 500-second duration. 2) We then superimpose 30% variation to the power data generated in the previous step to represent the runtime variability (voltage fluctuation and DPM actions, etc.). Such random power consumption data are then provided to the thermal-RC model (equation (5.3)) to generate the dynamic temperature trace of the chip. 3) We then assume the infra-red cameras can provide thermal map observations of the chip at 1 second intervals. A white Gaussian noise with zero mean and 2 degrees of standard deviation are superimposed onto the observation vectors to model the observation error. 4) We use both of our proposed techniques in section 5.3 to learn the statistical power characteristics (*i.e.* model parameters of $\Theta_K$) (500 data samples are used). The results are reported in table 5.1.

As can be seen from table 5.1, both of our methods generates accurate estimates for the statistical power characteristics (note that the average power in each module is around $6.7W$ which means our results are within 3% of error). Table 5.2 reports the convergence rate of our EM algorithm in the estimated mean of the BZIP2 benchmark. The convergence rate for other chip modules and benchmarks are similar. The results in table 5.2 demonstrated that within 5 iterations the error of our EM algorithm was able to converge to 0.12W from the initial values which has an error of 3.43W. The runtime for each iteration is also listed in the table. It can be seen our algorithm took a total of less than 10 seconds to learn the power characteristics of the system.

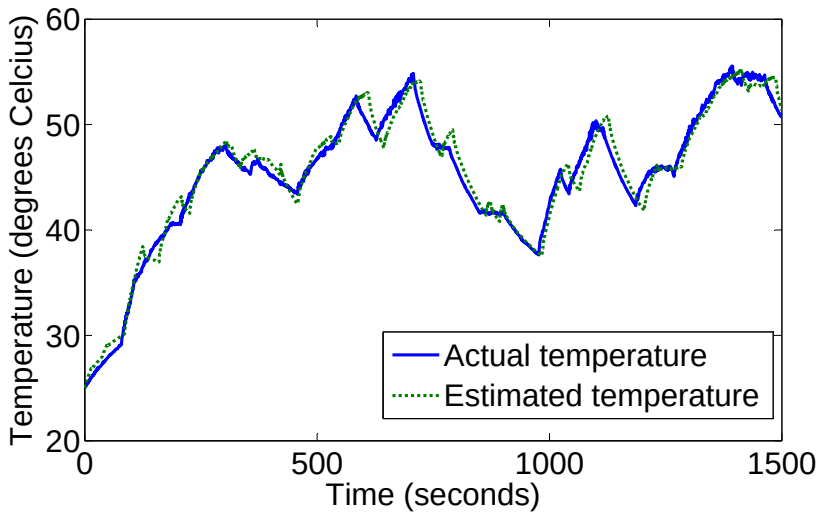



Figure 5.1: Temperature tracking results using our dynamically learned model

In figure 5.1 we demonstrate the thermal tracking results obtained using a Kalman filter (see [70]) based on our dynamically learned power statistical characteristics. Our test system switched among several benchmarks and the KF based approach can give consistently accurate thermal tracking results. A adaptive KF based tracking system could take advantage of the mixture model and adapt to the BGD switches, thus continuously providing accurate thermal estimates.

# Chapter 6

# Conclusion

In this thesis, we discussed several different areas under the general topic of temperature tracking and estimation in an on-chip environment by exploiting thermal sensors and the probabilistic characteristics of workload and randomness in fabrication variability, and *etc*. We presented our own approaches for addressing each different issue and compared their effectiveness with the state of the art research effort in each specific field. In chapter 2, we described an approach for chip-level thermal profile estimation based on limited sensor readings by exploiting the correlation in power density exhibited by different chip modules. In chapter 3, we went further and introduced a statistical framework for designing a complete and accurate thermal tracking infrastructure which can be instantiated on any chip and assist dynamic thermal management schemes to make optimal thermal control decisions. We discussed how to do sensor placement, how to counter sensor noise, how to account for the space and power constraint locally and globally by strategically compressing each sensor, and combining their readings into a central register in order to achieve global optimality. In chapter 4, we extended the popular Kalman filter based dynamic thermal tracking framework to specifically tackle the problem of changing workload characteristics (a property often observed for today's general purpose high performance processors), therefore further improving the accuracy and adaptability of any Kalman filter based thermal tracking system. In addition, we extended the standard Kalman filter to address the nonlinear effect of leakage power. In chapter 5, we proposed an methodology for extracting the chip power statistical characteristics automatically. The experimental results clearly demonstrate the effectiveness of each proposed methodology.

# Bibliography


[1] Moore's law backgrounder, intel corp.

[2] International technology roadmap for semiconductors, semiconductor industry assoc., 2011.

[3] Raid Ayoub and Tajana Rosing. Predict and act: Dynamic thermal management for multi-core processors. *Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 99–104, 2009.

[4] J. R. Black. Electromigration - a brief survey and some recent results. *IEEE Transactions on Electron Devices*, 16:338–347, 1969.

[5] David Brooks and Margaret Martonosi. Dynamic thermal management for high-performance microprocessors. *Proc. of IEEE/ACM International Symposium on High-Performance Computer Architecture*, pages 171–182, Janurary 2001.

[6] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. *Proc. of International Symposium on Computer Architecture*, 2000.

[7] Hongliang Chang and Sachin S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single pert-like traversal. *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pages 621–625, November 2003.

[8] Thidapat Chantem, X. Sharon Hu, and Robert Dick. Online work maximization under a peak temperature constraint. *Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 105–110, 2009.

[9] Poki Chen, Chun-Chi Chen, Chin-Chung Tsai, and Wen-Fu Lu. A time-to-digital-converter-based cmos smart temperature sensor. pages 1642–1648, 2005.

[10] Qikai Chen, M. Meterelliyoz, and K. Roy. A cmos thermal sensor and its applications in temperature adaptive design. *Proc. of IEEE/ACM International Symposium on Quality Electronic Design*, pages 6 pp.–248, March 2006.

[11] Lerong Cheng, Jinjun Xiong, and Lei He. Non-linear statistical static timing analysis for non-gaussian variation sources. *Proceedings of the 44th annual Design Automation Conference*, pages 250–255, 2007.

[12] Ihtesham Chowdhury, Ravi Prasher, and et. al. On-chip cooling by superlattice-based thin-film thermoelectrics. *Nature Nanotechnology*, pages 235–238, January 2009.

[13] Joachim Clabes, Joshua Friedrich, Mark Sweet, Jack DiLullo, Sam Chu, Donald Plass, James Dawson, Paul Muench, Larry Powell, Michael Floyd, Balaram Sinharoy, Mike Lee, Michael Goulet, James Wagoner, Nicole Schwartz, Steve Runyon, Gary Gorman, Phillip Restle, Ronald Kalla, Joseph McGill, and Steve Dodson. Design and implementation of the power5 microprocessor. pages 670–672, 2004.

[14] Brian Cline, Kaviraj Chopra, David Blaauw, and Yu Cao. Analysis and modeling of cd variation for statistical static timing. *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 60–66, 2006.

[15] Ryan Cochran and Sherief Reda. Spectral techniques for high-resolution thermal characterization with limited sensor data. *Proc. of IEEE/ACM Design Automation Conference*, pages 478–483, July 2009.

[16] Ayse Kivilcim Coskun, Tajana Simunic Rosing, and Kenny C. Gross. Proactive temperature management in mpsocs. *Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 165–170, 2008.

[17] John L. Crassidis and John L. Junkins. *Optimal Estimation of Dynamic Systems*. CRC Press, 2004.

[18] Thomas Ebi, Mohammad Abdullah Al Faruque, and Jörg Henkel. Tape: thermal-aware agent-based power economy for multi/many-core architectures. *Proceedings of the 2009 International Conference on Computer-Aided Design*, pages 302–309, 2009.

[19] Lei He Fei Li, Yan Lin and Jason Cong. Low-power fpga using pre-defined dual-vdd/dual-vt fabrics. *Proceedings of the ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, pages 42–50, 2004.

[20] Greg Hamerly, Erez Perelman, Jeremy Lau, and Brad Calder. Simpoint 3.0: Faster and more flexible program analysis. *Workshop on Modeling, Benchmarking and Simulation*, June 2005.

[21] Vinay Hanumaiah, Ravishankar Rao, Sarma Vrudhula, and Karam S. Chatha. Throughput optimal task allocation under thermal constraints for multi-core processors. *Proc. of IEEE/ACM Design Automation Conference*, pages 776–781, 2009.

[22] Lei He, Weiping Liao, and Mircea Stan. System level leakage reduction considering the interdependence of temperature and leakage. *Proc. of IEEE/ACM Design Automation Conference*, pages 12–17, 2004.

[23] W-L. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T. Theocharides, and M. J. Irwin. Thermal-aware floorplanning using genetic algorithms. *Proceedings of the 2005 International Symposium on Quality of Electronic Design*, pages 634–639, 2005.

[24] Hwisung Jung and Massoud Pedram. A stochastic local hot spot alerting technique. *Proc. of the IEEE Asia and South Pacific Design Automation Conference*, pages 468–473, 2008.

[25] Nam Kim, Todd Austin, David Blaauw, Trevor Mudge, Krisztian Flautner, Jie Hu, Mary Irwin, Mahmut Kandemir, and Vijaykrishnan Narayanan. Leakage current: Moore's law meets static power. *Computer*, pages 68–75, 2003.

[26] Eren Kursun and Chen-Yong Cher. Variation-aware thermal characterization and management of multi-core architectures. *ACM International Conference on Computer Design*, pages 280–285, 2008.

[27] ByungHyun Lee and Taewhan Kim. Optimal allocation and placement of thermal sensors for reconfigurable systems and its practical extension. *Proc. of the IEEE Asia and South Pacific Design Automation Conference*, pages 703–707, 2008.

[28] Kong Hoon Lee and Ook Joong Kim. Simulation of the cooling system using thermoelectric micro-coolers for hot spot mitigation. *26th International Conference on Thermoelectrics*, pages 279 –282, june 2007.

[29] Hang Li, Pu Liu, Zhenyu Qi, Lingling Jin, Wei Wu, Sheldon X. D. Tan, and Jun Yang. Efficient thermal simulation for runtime temperature tracking and management. *Proceedings of the 2005 International Conference on Computer Design*, pages 130–136, 2005.

[30] J. Long, S. Ogrenci Memik, G. Memik, and R. Mukherjee. Thermal monitoring mechanisms for chip multiprocessors. *ACM Transactions on Architecture and Code Optimization*, 5(2), August 2008.

[31] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. 1:281–297, 1967.

[32] Ravi Mahajan, Chia-Pin Chiu, and Greg Chrysler. Cooling a microprocessor chip. *Proc. of the IEEE*, pages 1476–1486, 2006.

[33] S. Ogrenci Memik, R. Mukherjee, and J. Long M. Ni. Optimizing thermal sensor allocation for microprocessors. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 27(3):516–527, March 2008.

[34] Gordon Moore. Cramming more components onto integrated circuits. *Electronics*, pages 114–117, 1965.

[35] R. Mukherjee, S. Mondal, and S. Ogrenci Memik. Thermal sensor allocation and placement for reconfigurable systems. *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pages 437–442, November 2006.

[36] Rajarshi Mukherjee and Seda Ogrenci Memik. Systematic temperature sensor allocation and placement for microprocessors. *Proc. of IEEE/ACM Design Automation Conference*, pages 542–547, July 2006.

[37] Ali M. Niknejad, Ranjit Gharpurey, and Robert G. Meyer. Numerically stable green function for modeling and analysis of substrate coupling in integrated circuits. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 17:305–315, 1998.

[38] Vidyasagar Nookala, David J. Lilja, and Sachin S. Sapatnekar. Temperature-aware floorplanning of microarchitecture blocks with ipc-power dependence modeling and transient analysis. *Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, pages 298–303, 2006.

[39] Abdullah Nazma Nowroz, Ryan Cochran, and Sherief Reda. Thermal monitoring of real processors: techniques for sensor allocation and full characterization. *Proc. of IEEE/ACM Design Automation Conference*, pages 56–61, 2010.

[40] Massoud Pedram and Shahin Nazarian. Thermal modeling, analysis, and management in vlsi circuits: Principles and methods. *Proc. of the IEEE*, pages 1487–1501, 2006.

[41] M.A.P. Pertijs, K.A.A. Makinwa, and J.H. Huijsing. A cmos smart temperature sensor with a 3 sigma inaccuracy of plus/minus 0.1 degrees from -55 degrees to 125 degrees celcius. pages 2805–2815, 2005.

[42] Zhouyuan Li Robert P. Dick Li Shang Hai Zhou Xianlong Hong Pingqiang Zhou, Yuchun Ma and Qiang Zhou. 3d-staf: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits. *Pro-*

*ceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design*, pages 590–597, 2007.

[43] H. Vincent Poor. *An Introduction to Signal Detection and Estimation*. Springer-Verlag, 2nd edition, 1998.

[44] Ravishankar Rao and Sarma Vrudhula. Efficient online computation of core speeds to maximize the throughput of thermally constrained multi-core processors. *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pages 537–542, 2008.

[45] Ravishankar Rao, Sarma Vrudhula, Chaitali Chakrabarti, and Naehyuck Chang. An optimal analytical solution for processor speed control with thermal constraints. *Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 292–297, 2006.

[46] Stefan Rusu. Trends and challenges in vlsi technology scaling towards 100nm. *Proc. of the European Solid-State Circuits Conference*, pages 194–196, 2001.

[47] Karthik Sankaranarayanan, Sivakumar Velusamy, Mircea Stan, Charles L, and Kevin Skadron. A case for thermal-aware floorplanning at the microarchitectural level. *Journal of Instruction Level Parallelism*, 7, 2005.

[48] Adel S. Sedra and Kenneth C. Smith. *Microelectronic Circuits*. Oxford University Press, 5th edition, 2004.

[49] S. Sharifi, ChunChen Liu, and T.S. Rosing. Accurate temperature estimation for efficient thermal management. *Proc. of IEEE/ACM International Symposium on Quality Electronic Design*, pages 137–142, March 2008.

[50] Shervin Sharifi, Chun Chen Liu, and Tajana Simunic Rosing. Accurate temperature estimation for efficient thermal management. *Proc. of IEEE/ACM International Symposium on Quality Electronic Design*, pages 137–142, March 2008.

[51] Bing Shi, Yufu Zhang, and Ankur Srivastava. Dynamic thermal management for single and multi-core processors under soft thermal constraints. *Proc. ACM International Symposium on Low Power Electronics and Design*, pages 165 – 170, August 2010.

[52] Bing Shi, Yufu Zhang, and Ankur Srivastava. Accelerating gate sizing using graphics processing units. *IEEE Trans. on Computer-aided Design of Integrated Circuits and Systems*, pages 160 – 164, Jan. 2012.

[53] Kevin Skadron, Mircea R. Stan, Karthik Sankaranarayanan, Wei huang, Sivakumar Velusamy, and David Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, pages 94–125, March 2004.

[54] Ankur Srivastava, Sachin Sapatnekar, Bing Shi, and Yufu Zhang. Encyclopedia of thermal packaging: Vol 4 - thermally-informed design of microelectronic components. 2014.

[55] M. Strasser, R. Aigner, M. Franosch, and G. Wachutka. Miniaturized thermoelectric generators based on poly-si and poly-sige surface micromachining. *Sensors and Actuators A: Physical*, 97-98:535–542, 2002.

[56] Haihua Su, Frank Liu, Anirudh Devgan, Emrah Acar, and Sani Nassif. Full chip leakage estimation considering power supply and temperature variations. *Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 78–83, 2003.

[57] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Introduction to data mining. 2005.

[58] James Tschanz Liqiong Wei Steven Burns Venkatesh Govindarajulu Vivek De Tanay Karnik, Yibin Ye and Shekhar Borkar. Total power optimization by simultaneous dual-vt allocation and device sizing in high performance micro-

processors. *Proc. of the IEEE/ACM Design Automation Conference*, pages 486–491, July 2002.

[59] R.A. Taylor and G.L. Solbrekken. Comprehensive system-level optimization of thermoelectric devices for electronic cooling applications. *IEEE Transactions on Components and Packaging Technologies*, 31(1):23 –31, march 2008.

[60] Peng Wang, Avram Bar-Cohen, Bao Yang, Gary L. Solbrekken, and Ali Shakouri. Analytical modeling of silicon thermoelectric microcooler. *Journal of Applied Physics*, 100(1):014501 –014501–13, 2006.

[61] Qi Wang and Sarma B. K. Vrudhula. Static power optimization of deep submicron cmos circuits for dual vt technology. *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 490–496, 1998.

[62] Ting-Yuan Wang and Charlie Chung-Ping Chen. 3-d thermal-adi - a linear-time chip level transient thermal simulator. *IEEE Transaction on Computer-Aided Design Integrated Circuits System*, 2002.

[63] Yonghong Yang, Changyun Zhu, Zhenyu Gu, Li Shang, and Robert P. Dick. Adaptive multi-domain thermal modeling and analysis for integrated circuit synthesis and design. *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pages 575–582, November 2006.

[64] Jing-Jia Liou Ying-Yen Chen. Extraction of statistical timing profiles using test data. *Proc. of the IEEE/ACM Design Automation Conference*, pages 509–514, July 2007.

[65] Yong Zhan and Sachin S. Sapatnekar. A high efficiency full-chip thermal simulation algorithm. *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pages 635–638, November 2005.

[66] Yanliang Zhang, Yunfei Chen, Changmeng Gong, Juekuan Yang, Ruiming Qian, and Yujua Wang. Optimization of supelattice thermoelectric materials

and microcoolers. *Journal of microelectromechanical systems*, pages 1113–1119, 2007.

[67] Yufu Zhang, Bing Shi, and Ankur Srivastava. Statistical framework for designing on-chip thermal sensing infrastructure in nano-scale systems. *IEEE Trans. on Very Large Scale Integration Systems*, pages 270 – 279, April 2013.

[68] Yufu Zhang and Ankur Srivastava. Accurate temperature estimation using noisy thermal sensors. *Proc. of IEEE/ACM Design Automation Conference*, pages 472–477, July 2009.

[69] Yufu Zhang and Ankur Srivastava. Accurate temperature estimation using noisy thermal sensors for gaussian and non-gaussian cases. *IEEE Trans. on Very Large Scale Integration Systems*, pages 1617 – 1626, June 2010.

[70] Yufu Zhang and Ankur Srivastava. Adaptive and autonomous thermal tracking for high performance computing systems. *Proc. of IEEE/ACM Design Automation Conference*, pages 68 – 73, June 2010.

[71] Yufu Zhang and Ankur Srivastava. A statistical framework for designing on-chip thermal sensing infrastructure in nano-scale systems. *Proc. of IEEE International Symposium on Physical Design*, pages 169 – 176, March 2010.

[72] Yufu Zhang and Ankur Srivastava. Leakage-aware kalman filter for accurate temperature tracking. *Proc. of IEEE International Green Computing Conference*, pages 1 – 7, July 2011.

[73] Yufu Zhang and Ankur Srivastava. Statistical characterization of chip power behavior at post-fabrication stage. *Proc. of IEEE International Green Computing Conference*, pages 1 – 6, July 2011.

[74] Yufu Zhang, Ankur Srivastava, and Mohamed Zahran. Chip level thermal profile estimation using on-chip temperature sensors. *Proceedings of IEEE International Conference on Computer Design*, pages 432–437, October 2008.

[75] Yufu Zhang, Ankur Srivastava, and Mohamed Zahran. On-chip sensor driven efficient thermal profile estimation algorithms. *ACM Transactions on Design Automation of Electronic Systems*, pages 25:1–25:27, May 2010.