UC Riverside
UC Riverside Electronic Theses and Dissertations

Title
Thermal and Power Estimation and Reliability Management for Commercial Multi-Core Processors

Permalink
https://escholarship.org/uc/item/63r4g0kv

Author
Zhang, Jinwei

Publication Date
2023

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Thermal and Power Estimation and Reliability Management for Commercial Multi-Core Processors

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Jinwei Zhang

March 2023

Dissertation Committee:

Dr. Sheldon X-D Tan, Chairperson
Dr. Daniel Wong
Dr. Hyoseung Kim

Copyright by
Jinwei Zhang
2023

The Dissertation of Jinwei Zhang is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

This dissertation represents the culmination of work that would not have been possible if I had not received the support, mentorship, and collaboration from numerous individuals. I wish to thank many people who gave me the support, help and assistance, both academically and mentally, throughout the journey of my Ph.D. program.

Firstly, I would like to convey my sincere gratitude to my Ph.D. primary advisor Dr. Sheldon X-D Tan for giving me the precious opportunity to pursue this degree and for all his guidance and support that were invaluable to my work. I would also like to thank my committee members, Dr. Daniel Wong and Dr. Hyoseung Kim, for their insightful and invaluable advice. Additionally, I would like to thank my fellow researchers at the VLSI Systems and Computation Lab (VSCLab) for their collaboration, feedback, and friendship. Namely, I would like to thank Dr. Sheriff Sadiqbatcha, Dr. Liang Chen, Dr. Han Zhou, Dr. Shaoyi Peng, Dr. Zeyu Sun, Wentian Jin, Shuyuan Yu, Jincong Lu, Sachin Sachdeva, Yibo Liu, Subed Lamichhane, Maliha Tasnim, Mohammad Amir Kavousi and Chinmay Raje. In addition to my mentors and colleagues at VSCLab, I would also like to thank a few scholars from other research groups who have contributed to some of my research projects. Namely, Dr. Nanpeng Yu, Dr. Yuanqi Gao, and Dr. An Zou with Washington University in St. Louis. In addition to my mentors and colleagues at UC Riverside, I would like to thank a few additional individuals who have had an enormous impact on my personal and professional life. Namely, I want to thank my parents, Bin and Yunchuan, and my girl friend Xiaohan for their love and encouragement.

The content of this thesis is fully or partially rewritten from these published materials. Please note that the contents of Chapter 3, which were rewritten from the 5th work in the following

publication list, were contributed equally by Jincong Lu and myself, where I mainly focused on the initial idea of the framework (Section 3.1) and data preparation (Section 3.2), and Jincong Lu focused more on the construction and Training of machine-learning-based model (Section 3.3), and we both have equal contributions to the experiments and results of this work.

1. Jinwei Zhang, Sheriff Sadiqbatcha, Wentian Jin and Sheldon X-D Tan, ”Accurate Power Density Map Estimation for Commercial Multi-Core Microprocessors,” 2020 Design, Automation and Test in Europe Conference and Exhibition (DATE), Grenoble, France, 2020, pp. 1085-1090.

2. Jinwei Zhang, Sheriff Sadiqbatcha, Michael O'Dea, Hussam Amrouch and Sheldon X-D Tan, ”Full-Chip Power Density and Thermal Map Characterization for Commercial Microprocessors Under Heat Sink Cooling,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 5, pp. 1453-1466, May 2022, doi: 10.1109 / TCAD.2021.3088081.

3. Jinwei Zhang, Sheriff Sadiqbatcha, Yuanqi Gao, Michael O'Dea, Nanpeng Yu, and Sheldon X-D Tan. 2020. HAT-DRL: Hotspot-Aware Task Mapping for Lifetime Improvement of Multicore System using Deep Reinforcement Learning. In Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD'20). Association for Computing Machinery, New York, NY, USA, 77-82.

4. Jinwei Zhang, Sheriff Sadiqbatcha and Sheldon X-D Tan, ”Hot-Trim: Thermal and Reliability Management for Commercial Multi-core Processors Considering Workload Dependent Hot Spots,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, doi: 10.1109/TCAD.2022.3216552.

5. Jincong Lu, Jinwei Zhang, Wentian Jin, Sachin Sachdeva, and Sheldon X-D Tan. 2023. Learning Based Spatial Power Characterization and Full-Chip Power Estimation for Commercial TPUs. In Proceedings of the 28th Asia and South Pacific Design Automation Conference (ASPDAC'23). Association for Computing Machinery, New York, NY, USA, 98-103, doi: 10.1145/3566097.3568347.

To my parents for all the support.

ABSTRACT OF THE DISSERTATION

Thermal and Power Estimation and Reliability Management for Commercial Multi-Core Processors

by

Jinwei Zhang

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, March 2023
Dr. Sheldon X-D Tan, Chairperson

Power, thermal and related reliability issues are among the major limiting factors for today's high performance multi-core processors. This is especially true after the breakdown of the so-called Dennard scaling, since power density starts to increase as IC technology advances. To enhance reliability, researchers have proposed many power/thermal regulation or dynamic management methods, including clock gating, power gating, dynamic voltage and frequency scaling (DVFS), and task migration. In this thesis, we present our findings to address the challenges of post-silicon power and thermal characterization, and dynamic thermal managements for lifetime reliabilities. We first address the problem of accurate full-chip power and thermal map estimation for commercial off-the-shelf multi-core processors. The novel scheme is developed to generate the true 2D power density maps based on the thermal measurements of the processor with backside cooling and facilitated with an advanced infrared (IR) thermal imaging system. the proposed method achieves both higher resolution and considerable speedup than a recently proposed state-of-art method. Then the second, we propose a novel approach for the real-time estimation of chip-level spatial power maps for commercial TPU chips based on a machine-learning technique for the first

time. In detail, we achieve estimating the spatial power for commercial TPUs from the hyperparameters of the neural networks (workloads) that are deployed on the TPUs in real-time. Thirdly, processors operating with heat sink cooling remains a challenging problem due to the difficulty in direct measurement. We build an FEM model to reconstruct the full-chip thermal maps for commercial processors while they are under heat sinks. Lastly, based on the spatial power characterization, we propose a new dynamic thermal and reliability management framework via task mapping and migration to improve the thermal performance and lifetime reliability of commercial multi-core processors. Compared to the existing works, the new approach is the first to optimize VLSI reliabilities by exploring workload-dependent power hot spots. The advantages of the proposed method over the Linux baseline task mapping and the temperature-based mapping method are demonstrated and validated on real commercial processors.

Contents

List of Figures **xiii**

List of Tables **xvi**

1 Introduction **1**
1.1 Post-Silicon Full-Chip Power Map Modeling 1
1.1.1 Contributions 4
1.2 Full-Chip Power Map Modeling for TPUs 6
1.2.1 Contributions 8
1.3 Thermal Characterization Under Heat Sink Cooling 8
1.3.1 Contributions 9
1.4 Thermal and Reliability Management Considering Hot Spots 11
1.4.1 Contributions 13
1.5 Organization of This Thesis 14

2 Post-Silicon Full-Chip Power Map Modeling **15**
2.1 Related Work and Motivation 15
2.2 The Power Map Modeling Setup 18
2.2.1 The Power Density Modeling Framework 18
2.2.2 Thermal Imaging System 19
2.3 New Power Density Map Estimation Method 22
2.3.1 Proposed Power Density Map Based on Laplace Operation 22
2.3.2 Estimation of Real Thermal Conductivity 26
2.3.3 Thermal Conductivity Estimation for Real Chip Die Area 30
2.3.4 FEM Model Architecture to Imitate Real Experiment Setup 32
2.4 Experimental Results and Discussions 35
2.4.1 Power Map Estimation and Comparison Results 36
2.5 Summary 40

3 Power Estimation for Commercial TPUs **42**
3.1 Power Map Estimation Framework 42
3.1.1 Estimation Flow Overview 42

3.1.2 Thermal IR Imaging System . . . 44
3.2 Data Preparation and Feature Selection . . . 45
3.2.1 Offline Power Map Acquisition . . . 46
3.2.2 Feature Selection Considering TPU Workloads . . . 48
3.3 CGAN-Based Estimation Model . . . 49
3.3.1 Review of CGAN . . . 49
3.3.2 Proposed CGAN-Based Power Estimation Framework . . . 51
3.4 Experimental Results and Discussions . . . 54
3.4.1 Validation of the Total Power Consumption . . . 54
3.4.2 Power Map Estimation Accuracy . . . 55
3.4.3 Computational Efficiency . . . 57
3.5 Summary . . . 58

4 Full-Chip Thermal Characterization With Heat Sink Cooling 60
4.1 Identify Exact Locations of On-Chip Thermal Sensors . . . 60
4.2 FEM Thermal Modeling With Heat Sink . . . 63
4.3 Ensuring the Same Power Density Maps for Both Cooling Conditions . . . 66
4.4 Application for Different Workloads . . . 69
4.5 Thermal Map Results Under Real Working Conditions . . . 70
4.6 Summary . . . 75

5 Thermal and Reliability Management Considering Hot Spots 76
5.1 Related Work . . . 76
5.2 Reliability Models . . . 77
5.2.1 EM Model . . . 78
5.2.2 NBTI Model . . . 79
5.2.3 HCI Model . . . 80
5.2.4 Summary of Reliability Models . . . 80
5.3 Observation and Motivation . . . 81
5.3.1 Thermography System Setup . . . 81
5.3.2 A Glance of Hot Spots . . . 82
5.4 Proposed Hotspot-Aware Task Allocation Framework . . . 87
5.4.1 Learning-Based Hot Spot Modeling and Detection . . . 87
5.4.2 Task Management Controller . . . 90
5.4.3 Proposed Mapping Control Algorithm . . . 94
5.5 Experimental Results and Discussions . . . 98
5.5.1 Comparison in System Performance . . . 101
5.5.2 Thermal and Reliability Improvement . . . 105
5.6 Summary . . . 113

6 Conclusions 115
6.1 Post-Silicon Full-Chip Power Map Modeling . . . 115
6.2 Power Estimation for Commercial TPUs . . . 116
6.3 Full-Chip Thermal Characterization With Heat Sink Cooling . . . 117
6.4 Thermal and Reliability Management Considering Hot Spots . . . 117

Bibliography **119**

List of Figures

1.1 Measured temperature of a hot spot versus the nearest sensor reading 12

2.1 Framework overview: (a) power modeling flow; (b) power inference flow 20
2.2 Thermal imaging system 21
2.3 A negative-Laplacian map (raw power map) example of experimental thermal measurements in 3D view 23
2.4 Simple ideal cases: (a) homogeneous heat source in orange region with total power 3W; (b) linear heat source in orange region (10×2mm), with areal power density $0.05(x-2)$ W $\cdot$ mm^{-2}, $x \in [2, 12]$ and total is 5W. 24
2.5 Comparison between estimated power density maps and exact ones for simple ideal examples. (a) and (b) are original power density maps for homogeneous heat source and linear heat source, respectively; (c) and (d) are the corresponding estimated power density maps of (a) and (b). 28
2.6 Estimated thermal conductivity of die area with respect to multiple workloads in time domain 31
2.7 Thermal structure created to imitate the real experiment setup. 33
2.8 Setting proper boundary conditions for thermal simulation model using measurements of CPU's idle status 34
2.9 The power map estimation results, from first to last row are experiment measured thermal maps, estimated power maps in 3D view, and simulation generated thermal maps, respectively. Each column is related to one workload at one steady state. . . 36
2.10 Comparison of the estimated power maps between the BPI (27×41 resolution) and the proposed method for three benchmarks (*cachebench*, *phpbench* and *cyclictest*) with respect to a specific time step during their full execution, respectively. 37
2.11 (a) Intel i7-8650U processor die floor-plan [Wik]; (b) an estimated power density map; (c) projection of power density map onto the processor die floor-plan. 39

3.1 Framework and data acquisition flow 43
3.2 (a) Thermal Imaging system setup (b) TPU chip under-test, Coral M.2 TPU module. TPU module is shown in the blue box. 44
3.3 (a) Thermal image and (b) power map of TPU for MobileNet-V2-224-1.0 network. 48
3.4 Architecture of CGAN model 51

3.5 Measured power maps (row #1), estimated power maps (row #2), and error maps (row #3). The numbers in the first row indicate the *Power Density RMSE* | *Average Power Density* (unit: $\mathrm{mW/mm^2}$). And numbers in the second row indicate the *Total Power Percentage Error* | *Total Power* (unit: W). 56

4.1 The correlation between pixel blocks and sensor values of each core. In each correlation map the pixel block that has the highest correlation to the core sensor measurement is marked in green dots, which infers to the on-chip sensor location. . . . 62
4.2 The sensor locations of CPU cores are identified in colored squares on an Intel i7-8650U quad-core processor [Wik]. 63
4.3 Processor without heat sink (left) and with heat sink (right) 64
4.4 (a) Thermal structure created to imitate the real experiment setup (without heat sink); (b) transparent view of processor area when with heat sink mounted on. . . . 64
4.5 The computed temperature of processor with heat sink under idle status. The small red boxes mark the on-chip thermal sensor locations. 66
4.6 Thermal sensor values when processor uses the back-side liquid cooling with respect to workloads 67
4.7 Thermal sensor values when processor is mounted with heat sink with respect to workloads 68
4.8 Total processor power with back-side liquid cooling and with heat sink during the time line 68
4.9 Full-chip thermal maps of processor under real working conditions with heat sink mounted on with respect to various workloads 70
4.10 Comparison between the HotSpot computed steady state temperature at sensor locations and the real sensor readings for (a) *cachebench* and (b) *phpbench* in a time duration 73

5.1 Infrared thermography system 82
5.2 Measured temperature of a hot spot versus the nearest sensor reading 83
5.3 (a) On-chip sensor readings (one sensor per core). (b) Measured temperature at the sensor locations (blue dots). 84
5.4 Power patterns of PARSEC-3.0 and Splash-2 benchmark workloads on a real Intel Core-i7 processor at the core scale (within the core). 85
5.5 Power detector network architecture 88
5.6 Example of task queue, waiting thread queue and the initial instant mapping. 92
5.7 Task management workflow. 93
5.8 Task migration in a management cycle. 97
5.9 Example of mapping a waiting thread to the available cores. 98
5.10 Power density detector neural network learning curves. 99
5.11 Power density ($\mathrm{W/mm^2}$) V.S. time steps (60 Hz) for workloads. Estimated power density compared to the measured power density at the identified hot spots. 100
5.12 Temperature at the identified true hot spot location HS3 for each core under three mapping techniques (Case 4). 106

5.13 Core frequencies (GHz) V.S. time steps when processor under different mapping policies: (a) Linux baseline (b) Temperature-based (c) Hot-Trim. Red lines are the mean lines of each frequency series (Case 4). 108
5.14 Normalized MTTF considering (a) EM, (b) NBTI and (c) HCI reliability effects regarding test case 4 where task release intervals satisfy $\Delta t \sim U(4, 12)$. 110
5.15 Normalized MTTF considering (a) EM, (b) NBTI and (c) HCI reliability effects regarding test case 5 where task release intervals satisfy $\Delta t \sim U(10, 20)$. 112

List of Tables

2.1 Computational cost comparison for power map estimation 38
2.2 Estimated power example for processor component (i7-8650U) 39

3.1 Selected Workload Features (Coral M.2 TPU, Google Edge) 50
3.2 Architecture and Parameters 53
3.3 Total Power Comparison 55

4.1 FEM computed temperature at sensor locations VS. real sensor values with heat sink 71

5.1 Average Power Density ($\mathrm{W/mm^2}$) at Hot Spots for Various Workloads 86
5.2 High-level Performance Metrics (Intel PCM) 89
5.3 Exemplary Ordering of Tasks and Cores and Migration 96
5.4 Test Cases of Task Series 102
5.5 Linux V.S. Hot-Trim: Performance and Temperature 104
5.6 Mean of Temperature over Time at the Worst Stressed Hot Spot Location HS3 of Each Core 107

Chapter 1

Introduction

1.1 Post-Silicon Full-Chip Power Map Modeling

Power, thermal and related reliability issues are among the major limiting factors for today's high performance multi-core processors. This is especially true after the breakdown of the so-called Dennard scaling, since power density starts to increase as IC technology advances [EBSA+12, Tay13]. To enhance reliability, researchers have proposed many power/thermal regulation or dynamic management methods, including clock gating, power gating, dynamic voltage and frequency scaling (DVFS), and task migration [BM01, HV14, LTHW15, WMT+16].

However, one important aspect of those works depends on how to correctly estimate the full-chip temperature map. Currently, the on-chip temperature is mainly obtained by performing the thermal analysis based on the run-time functional unit (or component-wise) power estimation of the processor. Estimating component power inputs, however, still remains challenging for commercial off-the-shelf microprocessors.

To obtain accurate on-chip temperature, we need to look at two important aspects of this problem: the accurate estimation of the input power and the accurate calculation of on-chip temperature from the thermal model and the input power.

Traditional power estimation methods focus on the functional unit (component-wise or core-wise) power estimation based on the measured temperature and total power [JM01, IM03, WJY+07, DNR13]. But those methods require understanding of the architecture details and functional units of each chip and many approaches are still ad-hoc, involving manual turning. At the same time, post-silicon full-chip power (density) map estimation is also important for power verification and package design. Power map is a 2D spatial distribution of heat dissipation in an IC chip. This problem was also coined as the *inverse thermal map to power map* problem as temperature can be more easily measured either directly or indirectly. Many approaches have been investigated in the past [WFMS09, CNR10, PSSK13, NWR13, BBVB16, RDB18a]. Most of the proposed methods tried to frame the problem as a nonlinear optimization problem (deterministically or statistically) once the thermal models are known. However, those methods do not work for general off-the-shelf commercial multi-core processors where only core-level power can be obtained [RDB18a]. Many of those only work for specialized silicon such as FPGAs [CNR10, PSSK13, NWR13, RDB18a]. In addition, they suffer from high computing cost and measurement noise although some mitigation techniques have been proposed such as using AC power [NWR13]. Recently, a new heat source identification method based on the measured temperature and 2D spatial Laplace transformation was proposed for general commercial multi-core processors [SZA+19], but this is not enough for full-chip power or thermal characterization.

Once we know the power inputs, thermal models are needed to compute the temperature outputs. Many power-based thermal models have been proposed including equivalent thermal RC networks [LVRW95, GW02], architecture level thermal modeling speed up [LLJ+06, LTPT09], finite difference based methods, such as HotSpot [HGV+06], and finite element based methods (FEM) [GJK+08].

Most existing power modeling methods and related thermal models, however, do not work well for commercial multi-core processors as mentioned before. It is even more challenging for modeling of commercial multi-core processors running in the normal working environment with heat sink cooling as it is difficult to directly measure the temperature of the chip's surface. On the other hand, we notice that commercial multi-core processors have many on-chip sensors, for instance, the Intel i7-8650U has one sensor for each core. One can leverage those sensor readings to validate the proposed thermal models. However, the exact locations of these temperature sensors are generally not known, neither provided by the processor's manufacturer.

The obtained full-chip power maps and thermal maps are instrumental for many applications. For instance, once power density maps are obtained, component power can be easily obtained by area integration over the chip layout. The estimated power map also provides many insights into power consumption of different modules, cores and uncore blocks in a microprocessor. Additionally, the power map and thermal maps obtained in real-time can be instrumental in exploring many power/thermal management techniques with various package and cooling solutions.

To mitigate the aforementioned problems, in this work, first, we try to obtain the full-chip power density map from the measured thermal maps/images of the commercial multi-core microprocessor when heat sink is removed. Second, we provide a new methodology to accurately estimate

the thermal map and hot spots of commercial multi-core processors running in the normal working environment with heat sink cooling. The obtained full-chip power and thermal maps under normal heat sink cooling can provide many insightful hot spot information, which can't be obtained by physical sensors and will enable new applications for dynamic thermal/power/reliability management.

1.1.1 Contributions

The specific contributions of Chapter 2 are as follows:

- First, different than all the existing power estimation methods, the new method, based on the first principle of heat transfer, performs a much more efficient 2D spatial Laplace operation on a given thermal map to obtain the so-called *raw* power density map. This consists of both positive and negative values due to the steady-state nature and boundary conditions of the microprocessors. We study two motivation cases to provide many insights into the relationship between raw power density maps and real power density maps. Our work is enabled by an advanced thermal measuring platform with a high-precision thermal camera and a cooling system installed on the back side of the CPU. This allows us to take explicit temperature images (thermal maps) of CPU die while the CPU is under load.
- Then based on the total power of the microprocessor obtained using an online CPU monitoring tool, we develop a novel scheme to generate the true positive-only power density map from the measured raw power density map. At the same time, we develop a novel method to compute effective thermal conductivity of the microprocessor die, which is an important parameter for the subsequent thermal modeling.

- To validate the power density map and the estimated actual effective thermal conductivity of the microprocessors, we construct a thermal model with COMSOL Multiphysics [Mul14]. The model mimics the real experimental setup (without heat sink), with the same boundary conditions used in the IR imaging system. We use the thermal measurements when CPU is in idle status to determine the boundary conditions of the thermal simulation model. Then we use FEM method to compute the thermal map based on the estimated power density map to ensure the computed thermal maps match the measured thermal maps using the FEM method.
- Numerical results show that the proposed power map estimation method is not only more than $100\times$ faster but also more fine-grained than the state-of-art blind power identification (BPI) method [RDB18a].

Experimental results on an Intel i7-8650U 4-core processor with back side cooling technique demonstrate 96% similarity (2D correlation) between the measured thermal maps and the computed thermal maps, which are computed using the estimated power maps and accurately built FEM thermal model.

Chapter 2 is organized as follows. Section 2.1 reviews the existing relevant work. Section 2.2 shows the power modeling framework and IR thermography setup used in this study. Section 2.3 presents the proposed power density map estimation method and the effective thermal conductivity estimation method. Section 2.4 presents the experimental results and comparisons with the current state-of-art method. Section 2.5 summarizes the work of this chapter.

1.2 Full-Chip Power Map Modeling for TPUs

With the continuing trend of rapid integration and technology scaling, today's high-performance processors have become more thermally constrained than ever before. An increase in temperature has been shown to exponentially degrade the reliability of the semiconductor chips [ITR03], and hence has become one of the leading concerns in the industry today. To address this trend, runtime power and thermal control schemes are being implemented in most, if not all new generations of processors and are crucial in any modern processor [EBSA+12, Tay13]. However, these control schemes require accurate real-time thermal information, and essentially the power information, ideally the spatial power density map of the entire chip area, in order to be effective [KSH+06, KCS12]. On-chip temperature sensors alone cannot provide the full-chip temperature information since the number of sensors that are typically available is very limited due to their high area and power overheads [SZZ+20b]. Furthermore, power characterization for commercial tensor processors (TPUs) is rarely studied and reported.

To obtain precise thermal and power control, we need to look at two important aspects of this problem: the accurate estimation of the on-chip power and the accurate calculation of temperature from the thermal model and the on-chip power inputs. Traditional power estimation methods focus on the functional unit (component-wise or core-wise) power estimation based on the measured temperature and total power [JM01, IM03, WJY+07, DNR13]. But those methods require an understanding of the architectural details and functional units of each chip and many approaches are still ad-hoc, involving manual turning. At the same time, post-silicon (no prior layout information is needed) spatial power map estimation from thermal information has been widely studied. This problem was also coined as the *inverse thermal map to power map* problem as the temperature can

be more easily measured either directly or indirectly. Many approaches have been investigated in the past [WFMS09, CNR10, PSSK13, NWR13, BBVB16, RDB18a]. Most of the proposed methods tried to frame the problem as a nonlinear optimization problem (deterministically or statistically) once the thermal models are known. However, those methods do not work for general off-the-shelf commercial processors where only core-level power can be obtained [RDB18a]. Many of those methods only work for specialized silicon such as FPGAs [CNR10, PSSK13, NWR13, RDB18a]. Recently, new spatial power map estimation methods based on the measured spatial temperature, 2D spatial Laplace transformation, and processor's performance monitors were proposed for general commercial multicore processors [ZSO+21]. Specifically, the machine-learning based power source hot spot estimation [SZZ+20b] and full chip thermal map estimation [SZAT21] have been proposed. Those methods estimate the hot spot or the full chip thermal maps based on the real-time on-chip performance information such as Intel's Performance Counting Monitor (IPCM) [Int].

However, these methods can hardly be applied to TPUs (like the Google Coral M.2 TPU used in this paper) as there is no real-time utilization information such as IPCM from the TPU chips. As a result, the existing full-chip power map estimation methods cannot be applied to commercial TPUs.

In Chapter 3, we try to address the aforementioned issues and propose a novel machine-learning based approach to estimate the full-chip power density distribution of commercial TPU chips.

1.2.1 Contributions

The key contributions of this chapter are as follows:

- We developed a generalized full-chip power map estimation method that is based on the hyperparameters of the TPU's workloads (i.e., neural networks inferencing on the TPU), without the knowledge of TPU's performance monitors or supply power.
- We treat the full-chip power density map estimation problem as an image generation problem, where the input features are given number of hyperparameters and TPU resource information (generated by the TPU compiler). We propose to use the Conditional Generative Adversarial Networks (CGAN) to generate such power map images from the given features.
- Experimental results show that the predictions of power maps are quite accurate, with the RMSE of only $4.98\text{mW}/\text{mm}^2$, or 2.6% of the full-scale error. The speed of deploying the proposed approach on an Intel Core i7-8650U is as fast as 6.9ms, which is suitable for real-time estimation.

Chapter 3 is organized as follows. Section 3.1 shows the power modeling framework and IR thermography setup used in this study. Section 3.2 models the spatial power from the workload features that are available in real time. Section 3.3 describes the architecture of the proposed CGAN-based neural net model for power map estimation. Section 3.4 presents the experimental results and comparisons. Section 3.5 summarizes the work of this chapter.

1.3 Thermal Characterization Under Heat Sink Cooling

Most existing power modeling methods and related thermal models, however, do not work well for commercial multi-core processors as mentioned before. It is even more challenging for

modeling of commercial multi-core processors running in the normal working environment with heat sink cooling as it is difficult to directly measure the temperature of the chip's surface. On the other hand, we notice that commercial multi-core processors have many on-chip sensors, for instance, the Intel i7-8650U has one sensor for each core. One can leverage those sensor readings to validate the proposed thermal models. However, the exact locations of these temperature sensors are generally not known, neither provided by the processor's manufacturer.

To mitigate the aforementioned problems, in Chapter 4, first, we try to obtain the full-chip power density map from the measured thermal maps/images of the commercial multi-core microprocessor when heat sink is removed. Second, we provide a new methodology to accurately estimate the thermal map and hot spots of commercial multi-core processors running in the normal working environment with heat sink cooling. The obtained full-chip power and thermal maps under normal heat sink cooling can provide many insightful hot spot information, which can't be obtained by physical sensors and will enable new applications for dynamic thermal/power/reliability management.

1.3.1 Contributions

In this chapter, we aim to address the aforementioned issues with the novel contributions summarized below:

- To develop the thermal models with heat sink cooling, we first try to identify the exact locations of on-chip sensors of commercial multi-core processors based on the correlation analysis of measured thermal map traces and on-chip sensor readings for the first time.

- Next, we construct the second thermal model with COMSOL Multiphysics that mimics the real set up of multi-core processors with heat sinks under real working conditions. The model is validated by ensuring that the computed thermal maps using the estimated power density maps match the temperature values obtained from the real sensors of the chip with heat sink. We also manage to keep power maps consistent for both the back cooling and heat sink cooling to minimize the leakage impacts on power.
- Numerical results show that the proposed power map estimation method is not only more than $100\times$ faster but also more fine-grained than the state-of-art blind power identification (BPI) method [RDB18a].

Experiments were conducted on an Intel i7-8650U 4-core processor by taking advantage of the power maps obtained in Chapter 2. In detail, FEM simulations with COMSOL Multiphysics [Mul14] were implemented based on such estimated power maps and the accurately built FEM thermal model with heat sink that imitates the real setup. Results show that under the real working condition with heat sink obscured, the average absolute error is only 2.2°C over a 56°C dynamic temperature range and about 3.9% percentage error between the computed thermal maps and the real thermal maps at the sensor locations.

Chapter 4 is organized as follows. Section 4.1 describes the new method to identify the locations of physical thermal sensors of commercial off-the-shelf processors. Section 4.2 presents the FEM architecture that imitates the real working situation when processor is under the heat sink. Section 4.3 details how to ensure the power maps obtained with back-side cooling match the power maps with heat sink cooling. Section 4.4 describes the proposed FEM thermal model with heat sink

can be applied to different workloads once it has been built. Section 4.5 presents the experimental results. Section 4.6 summarizes the work of this chapter.

1.4 Thermal and Reliability Management Considering Hot Spots

Power density increases with technology scaling, which can cause severe thermal and reliability problems in high performance multi-core systems [EBSA+12]. Temperature and power has significant impacts on all major long-term reliability effects such as electro-migration (EM) for interconnects, bias-temperature-instability (BTI) and hot-carrier-injection (HCI) for CMOS devices [AvE+14]. As a result, many research works have been investigated to find efficient methods to improve both system performance and reliability via dynamic thermal/reliability management (DTM/DRM) methods, which control the thermal and reliability behavior of multi-core systems by online control such as task migration strategies [CRW07, GMQ10, LFQ12, LTHW15, PMJH20].

However, existing DTM techniques either using DVFS or task migration are highly dependent on the on-chip location-fixed temperature sensors. Due to high design overheads, currently only a limited number of on-chip digital temperature sensors (DTS) can be allocated on a silicon chip. A recent study shows that the number of hot spots on a typical commercial processor far exceeds the amount of embedded sensors [SZZ+20b]. Consequently, thermal and reliability management algorithms that solely depend on the sensors become insufficient for modern multi-core systems, as power and thermal hot spots distinguish within cores under different workloads while having the same sensing temperature.

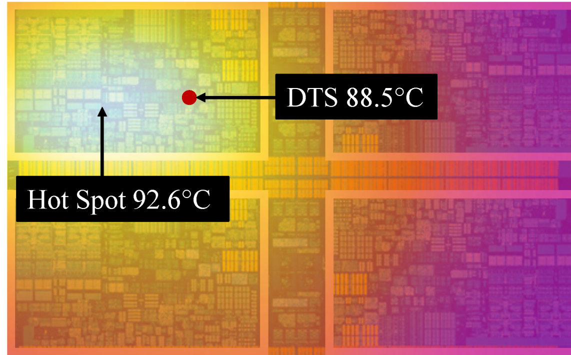


Figure 1.1: Measured temperature of a hot spot versus the nearest sensor reading

Fig. 1.1 shows a significant temperature difference[1] between a hot spot and the nearest sensor location on an Intel Core i7 quad-core processor under the SPLASH-2 workload *radiosity* (only displaying the quad-core area). Therefore, as the reliability of a core is mainly determined by the thermal hot spots, temperature per-core information alone is insufficient for DTM/DRM techniques. On the other hand, recent studies [RDB18a, ZSJT20] show that one can identify the power density distribution of a multi-core processor with advanced thermal characterization.

Based on this observation, in this article, we introduce a new efficient and scalable task mapping algorithm for the thermal and reliability management for **commercial multi-core processors** via machine learning based modeling for power density at the true hot spots[2]. Our work is facilitated by an advanced thermal imaging system for measuring the spatial temperature across the full processor. Once temperature maps are measured, one can obtain the power density maps (the corresponding heat sources or hot spots) through the thermal-to-power technique using thermal measurements [ZSJT20]. After that, we build a learning-based model for power density at the major hot spots in cores. We remark that the power or hot spot identification for commercial multi-core

[1]Temperatures are measured with a calibrated thermal imaging system (see Section 5.3.1).

[2]In the paper, hot spot is designated for power density hot spot instead of the traditional thermal hot spot. Power density hot spots are a superset of thermal hot spots and can be viewed as the potential thermal hot spots in general.

processors under different workloads can also be carried out on chips with heat sink cooling in practical work settings [ZSO+21].

1.4.1 Contributions

The following summarizes key contributions of this chapter.

- First, we show that the existing task mapping techniques, which solely depend on per-core sensor temperature, may lead to subpar quality solution for chip reliability as the true hot spots of cores can be stressed unevenly.
- Second, based on this observation, we employ a fast, run-time accurate machine learning model to estimate the exact spatial hot spots from the given workloads. With this, we propose a scalable and efficient task mapping approach to optimize the reliability of the multi-core system.
- Third, compared to existing works, the new task mapping approach is the first one to explore the workload-dependent power hot spots and its advantages over the existing Linux task scheduling method and temperature-based method, and has been demonstrated, validated on *real commercial multi-core processors*. Experiments on a real Intel Core i7 quad-core processor executing PARSEC-3.0 and SPLASH-2 benchmarks show that, compared to the Linux baseline, the core and hot spot temperature can be reduced by 1.15∼1.31°C. In addition, *Hot-Trim* can improve the chip's EM, NBTI and HCI related reliability by 30.2%, 7.0% and 31.1% respectively compared to Linux baseline without any performance degradation. Furthermore, it improves EM and HCI related reliability by 29.6% and 19.6% while further reduces the

temperature by half a degree compared to the conventional temperature-based mapping technique.

This paper is organized as follows: Section 5.1 reviews some related works. Section 5.2 discusses three major reliability effects and their models used in this work. Section 5.3 presents the thermal imaging system setup and a motivation example for this work. Section 5.4 introduces the proposed hot-spot-aware task migration method. Section 5.5 presents the results and comparisons on a real Intel i7 quad-core processor. Section 5.6 summarizes the work of this chapter.

1.5 Organization of This Thesis

The rest of this thesis is organized as follows. Chapters 2 and chapter 3 describes the post-silicon full-chip power modeling approach for commercial multi-core processors and commercial TPU chips, respectively. Chapter 4 introduces the full-chip thermal modeling method while the multi-core processor is obscured by a heat sink. Chapter 5 elaborates a dynamic thermal and lifetime reliability management method for commercial multi-core processors. Each chapter begins with related work review or motivation, followed by details of the work and experimental results, and ends with a chapter-specific summary, and the corresponding published paper is indicated in the summary. Finally, chapter 6 summarizes the overall thesis.

Chapter 2

Post-Silicon Full-Chip Power Map Modeling

2.1 Related Work and Motivation

Post-silicon power modeling is concerned with finding the powers of functional blocks or power density maps of a whole chip under various workloads. A few existing works have proposed to estimate the component power and the total power of a *real* microprocessor [JM01, IM03, WJY$^+$07, DNR13]. One idea is to tune each component unit power until the summation matches with the total power that is measured experimentally [JM01, IM03]. The main difficulty of those approaches, however, is that searching for component unit power values still remains an ad-hoc approach, which almost always involves manual tuning. Wu *et al.* [WJY$^+$07] tried to mitigate this problem by performing linear regression with K-means method to identify the unique power track patterns from the running programs. Dev *et al.* [DNR13] frames the problem as constrained

optimization problem once the thermal models are obtained from finite element simulation and measurement. Recently an recurrent-neural-network (RNN) based approach has been proposed to quickly estimate the thermal and power hotspots based on the system performance metrics such as Intel's performance counter monitor (IPCM) [SZA+19].

At the same time, many post-silicon full-chip power map estimation works have been proposed [WFMS09, CNR10, PSSK13, NWR13, BBVB16, RDB18b]. Most of those proposed methods tried to frame the *inverse thermal to power* problem as the nonlinear optimization problems as follows:

$$\min ||M \cdot \mathbf{p} - \mathbf{t}||^{\mathbf{2}} \tag{2.1}$$

where M represents the steady-state power to temperature map matrix, which is dependent on the specific thermal models used. $\mathbf{p}$ is a vector that gives the power density at a set of discrete die locations. $\mathbf{t}$ is a vector of the measured or calculated temperatures at the same locations of corresponding power signals. M can be directly measured from the FPGAs [CNR10, PSSK13, NWR13, RDB18b] or by using some approximation methods such as the power blurring method [WFMS09], or by using pre-defined parameterized analytic forms for a special 3D IC chip along with a parameter regression method [BBVB16]. Reda *et al.* [RDB18b] shows the power estimation for commercial multi-core processors. However, it can only deliver core-wise power information based on total power and core-wise thermal sensor measurement.

Paek *et al.* [PSSK13] added some statistical spins into this problem by computing the maximum likelihood of power $\mathbf{p}$ given a condition of the thermal map $\mathbf{t}$. But it requires an accurate thermal model, i.e. using HotSpot [HGV+06] for simulation results to start with for the required accuracy. The author indeed tested the method on a real FPGA chip, but they only achieve 90.7%

accuracy on average. This shows the difficulty in building accurate thermal models for real silicon chips.

In summary, first of all, the existing methods do not work well for off-the shelf commercial multi-core processors as many of them only work for specialized silicon such as FPGAs [CNR10, PSSK13, NWR13, RDB18b] or special 3D chips [BBVB16]. Second, they suffer from large computing costs as they try to solve nonlinear optimization problems shown in (2.1). Some of those methods also require special regulation items [WFMS09] or scaling or permutations for matrix M [RDB18b] during the optimization to enforce some physics laws, which will lead to more computational costs.

On the other hand, recent study show that the relative power density map can be easily obtained by 2D spatial Laplace transformation of measured or calculated temperature maps based on the first principle of heat conduction [SZA$^+$19, SZZ$^+$20b]. However, there exist several major differences between this work and the published work. Firstly, this work targets a different set of problems: finding the true 2D power density maps of multi-core processors, validating the results via thermal measurements and applying the power maps for thermal map estimation with different cooling configurations and different workloads. The proposed techniques will bring much more useful applications.

Second, the previous two works simply applied Laplace method to obtain the power maps from the thermal maps. However, such power maps, called *raw power maps*, are not physical power density maps since the raw power maps have negative values as shown in Fig. 2.3. Furthermore, the prior works mainly identify a few major heat source locations by locating the local maxima from the raw power maps, whereas this work tries to estimate power density values ($\mathrm{W}\cdot\mathrm{mm}^{-2}$) quantitatively

across the full chip. With the full-chip power map, one can further perform the full-chip thermal map estimation for different heat sink cooling configurations. To the best of the authors' knowledge, this is a novel thermal modeling capability achieved for the first time.

2.2 The Power Map Modeling Setup

In this section we briefly outline the framework of the proposed approach, thermal imaging system, and necessary data collection from the commercial multi-core processor.

2.2.1 The Power Density Modeling Framework

Power map (surface power density distribution) has tight relationship with the temperature distribution, the Laplace transform of temperature and the thermal conductivity. Our proposed approach involves two kinds of data. The first dataset is the thermal maps of CPU measured through a high-precision thermal camera, which senses the infrared emissions from CPU surface and transforms them into images of temperature distribution. The second is the total CPU power consumption over time, which can be obtained through the processor's Performance Counter Monitor (PCM).

For real processors, as we do not know the exact power density distribution, to verify the estimated power density maps, we compare their corresponding thermal maps. The idea is to build a thermal simulation framework, which mimics the real experimental setup of the chip in the thermal imaging system with similar thermal boundary conditions and thermal structures. Hence, the verification flow of the real chip can be summarized as follows:

1. Obtain sufficient number of estimated power maps based on the proposed method. The experimental measurements should include an *idle* status, meaning CPU has extremely low power, which will be used to set boundary conditions.

2. Build an FEM thermal simulation model that mimics the real structure of the processor die in the thermal imaging system.

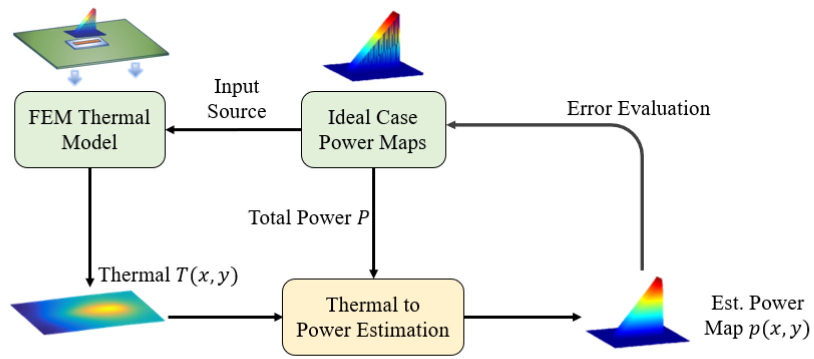
3. Substitute the estimated thermal conductivity κ as well as the estimated power map into thermal simulation model as parameters and inputs.

4. Examine similarities between the computed thermal maps and the experimentally measured thermal maps. Higher degree of identity indicates higher precision of power map estimation, vice versa.

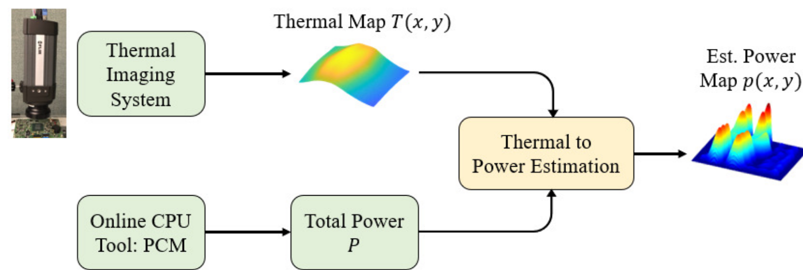
Fig. 2.1(a) illustrates the *power modeling flow* of the proposed power density map estimation model and evaluates the accuracy of the estimations based on the ideal cases. Fig. 2.1(b) shows the *power inference flow* from data resources to the estimated results for real processors during the run-time.

2.2.2 Thermal Imaging System

High precision and resolution of thermal map measurements are critical to the estimation results. One thermal imaging method proposed in [AH15] maximized the explicitness of thermal maps by directly exposing the top surface of CPU die to the camera, while ensuring the CPU's normal thermal condition by cooling it from the back side of the motherboard. Massive heat generated



(a)



(b)

Figure 2.1: Framework overview: (a) power modeling flow; (b) power inference flow

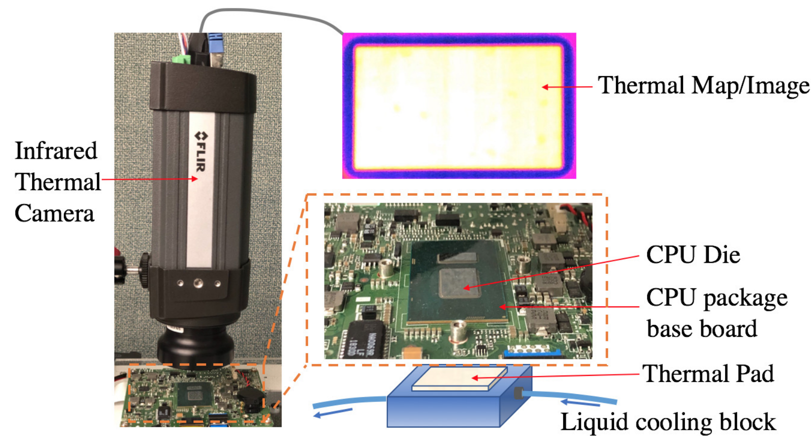


Figure 2.2: Thermal imaging system

from CPU flows downwards through the motherboard into the cooling system, and is dissipated by the quickly circulating coolant.

This work adopts the thermal imaging method proposed in [AH15] where the coolant flow does not contact the chip directly. It uses a Peltier device (electrothermal devices) with soft thermal pads stacked together between the liquid pipe and the back side of motherboard. Further, we take advantage of a high-precision thermal camera installed closely over the CPU die, as shown in Fig. 2.2. The model of our thermal camera is FLIR A325SC (240×320px images with 16 bit precision and 60 Hz capturing rate). Thanks to a close-up lens, the camera makes temperature difference 50mK clearly visible within as small as 50μm/px.

We remark that another thermal imaging system was proposed for power map estimation for commercial processors in [DNR13]. The system uses a transparent silicon window over the surface of the chip and pumps the liquid oil to flow through the window to remove heat from the chip surface directly. After this, FEM method is used to model the setup and generate the power-to-thermal transfer matrix R. Such front-cooling techniques typically require more delicate setup, post

imaging processing and more complicated thermal modeling as the oil based liquid cooling affects the thermal images directly.

2.3 New Power Density Map Estimation Method

In this section, we present our new method to estimate the full-chip power density from the real multi-core processors. We start with a simple example, which leads to an important observation for the proposed power map estimation method. Then we will present the approach to compute the thermal conductivity of the real chip, which is a critical parameter for thermal modeling and validation.

2.3.1 Proposed Power Density Map Based on Laplace Operation

Recently Sadiqbatcha *et al.* [SZZ+20b] proposed an idea of identifying power sources from thermal maps using Laplace transformation. The work starts from the fundamental heat diffusion equation (2.2), which gives the relationship between temperature and heat generation:

$$\rho C_P \frac{\partial T}{\partial t} - \nabla(\kappa \nabla T) = g \tag{2.2}$$

where T is temperature (K), ρ is the mass density of the material ($\mathrm{kg} \cdot \mathrm{m}^{-3}$), C_P is the mass heat capacity ($\mathrm{J} \cdot \mathrm{kg}^{-1} \cdot \mathrm{K}^{-1}$), κ is the thermal conductivity ($\mathrm{W} \cdot \mathrm{m}^{-1} \cdot \mathrm{K}^{-1}$) and g is the spatial heat energy generation ($\mathrm{W} \cdot \mathrm{m}^{-3}$).

When CPU runs into steady state, the transient term can be ignored and equation (2.2) can be simplified as:

$$-\kappa \nabla^2 T = g_T(x, y) \tag{2.3}$$

where ∇^2 is the Laplace operator. From the simplified heat equation (2.3), we can see that the *negative spatial Laplacian* of the temperature distribution across the die is proportional to the spatial heat generation, i.e., the underlining heat-sources $g_T(x, y)$, called the *raw power map*.

This paper distinguishes from the prior work in the way this paper finds the true 2D power density maps ($p(x, y)$, $\mathrm{W} \cdot \mathrm{mm}^{-2}$) of multi-core processors and validates the results via thermal measurements. Specifically, [SZZ+20b] simply applied Laplace method to obtain the power maps from the thermal maps. However, such power maps, called *raw power maps*, are not physical power density maps as they contain negative values. Negative values are clearly shown in Fig. 2.3. They cannot be explained by CPU power distribution since CPU power will never be negative. Furthermore, the prior work mainly identifies a few major heat source locations by locating the local maxima from the raw power maps without solving for the physical parameter of thermal conductivity κ. However, this paper quantitatively estimates the power density values ($\mathrm{W} \cdot \mathrm{mm}^{-2}$) across the full chip as well as the thermal conductivity κ of the chip die. This means we are able to learn the spatially continuous heat sources and their actual power densities.

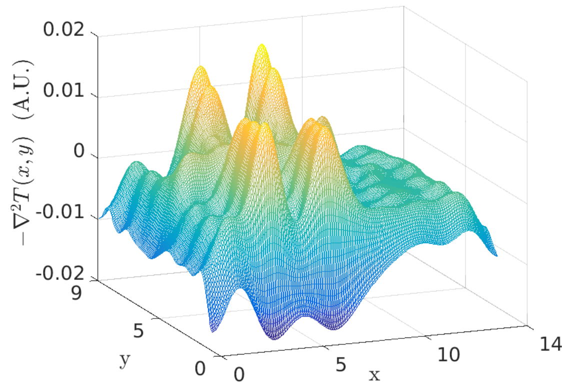


Figure 2.3: A negative-Laplacian map (raw power map) example of experimental thermal measurements in 3D view

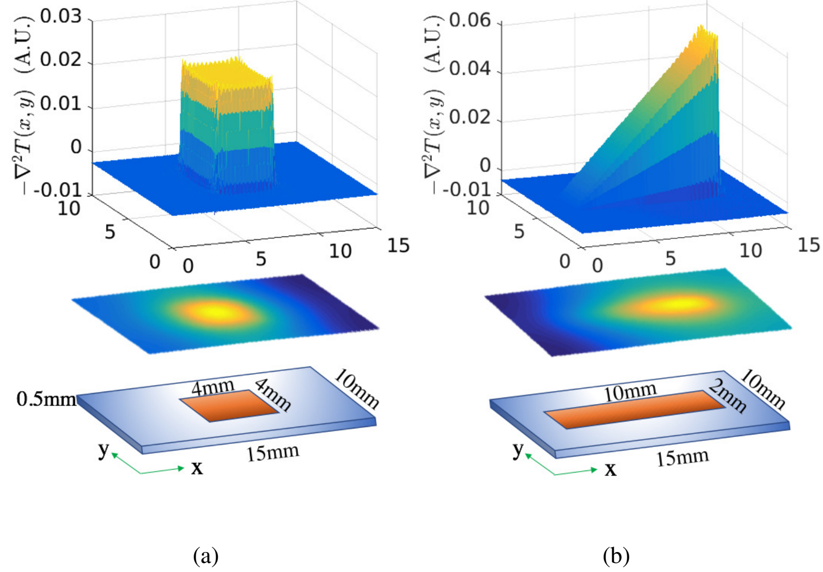


Figure 2.4: Simple ideal cases: (a) homogeneous heat source in orange region with total power 3W; (b) linear heat source in orange region (10×2mm), with areal power density $0.05(x-2)$ W $\cdot$ mm^{-2}, $x \in [2, 12]$ and total is 5W.

In order to closely study the relationship between the Laplace transform of temperature and the CPU power distribution, we build a simple ideal case in COMSOL Multiphysics heat transfer tool [Mul14]. This structure contains a rectangular base whose geometric dimension is $10\times15\times0.5$mm, and a $4\times4\times0.5$mm heat source block embedded in the base, whose total power is set at 3W (0.1875W $\cdot$ mm^{-2}) and homogeneous in space (Fig. 2.4(a)). The geometries can be flexible, we set it to approximately match the general size of CPU die and core. The κ of the material of the structure in this case is 400W $\cdot$ m^{-1} $\cdot$ K^{-1}. For the boundary conditions, a convective heat flux set at 1000W $\cdot$ m^{-2}K^{-1} is applied to the bottom surface. This convective heat flux mimics the heat dissipation through bottom surface. Ambient temperature is set to room temperature 297K.

As shown in Fig. 2.4(a), it is obvious that the high-rising portion of negative-Laplacian map reflects the area of active power density. Furthermore, we observe that *integration of negative-*

Laplacian map over all the area (pixels) is always zero, no matter how the power setting or geometry changes. The reason is that the thermal map we obtained comes from steady state of the CPU with specific thermal boundary conditions. This means that power generation and power dissipation are balanced in such equilibrium state. The negative power density value actually stands for more power dissipation than generation at the specific location due to thermal transfer and convection process at the boundaries. Where positive value means the opposite. For the very positive high-rising portion, which means the heat generation is significantly larger than the dissipation, typically indicates the hotspots of the chip.

In another example, we have an ideal linear heat source, whose power density increases linearly along the x-axis with total power 5W. Fig. 2.4(b) illustrates the location of power source, power setting and its corresponding negative-Laplacian map. We can observe that the negative-Laplacian in such rectangular power region shows an important linear trend as well, while the surrounding region is negative.

Based on the observations from these examples, we can see that the positive part of the negative-Laplacian map are the region where most of the real power densities are located. In this two simple cases, they cover the 100% real power density distribution. As a result, we can just use the *positive part* of the negative-Laplacian map to represent the estimated power map. Though the actual values of power map in those region are yet to be determined, which will be answered in the following section by calculating the accurate thermal conductivity κ.

2.3.2 Estimation of Real Thermal Conductivity

Modern microprocessor die is usually as thin as 0.5mm or below. Thus thermal characteristics along z-axis can be viewed as homogeneous. Power density distribution is only important on the surface x-y plane.

In reality, heat density is a combination of CPU power and heat dissipation by heat sink. Assume the thickness of CPU die is Δz, $p(x, y)$ stands for surface power density (W $\cdot$ mm^{-2}) at location (x, y) and $p_d(x, y)$ denotes heat dissipated locally. Heat density can be expressed as:

$$g_T(x, y) = \frac{p(x, y) - p_d(x, y)}{\Delta z} \tag{2.4}$$

Then for location (x, y), (2.3) can be rewritten as:

$$-\kappa \nabla^2 T(x, y) = \frac{p(x, y) - p_d(x, y)}{\Delta z} \tag{2.5}$$

Considering the entire chip, integrate both sides on the whole die area,

$$-\kappa \int \nabla^2 T(x, y) dxdy = \int \frac{p(x, y) - p_d(x, y)}{\Delta z} dxdy \tag{2.6}$$

Suppose P is total CPU power, and P_d is total heat dissipation (mainly through convective heat flux), (2.6) can be further written as:

$$-\kappa \int \nabla^2 T(x, y) dxdy = \frac{P - P_d}{\Delta z} \tag{2.7}$$

At steady state P should be equal to P_d. This infers the integration on the right hand side of (2.7) would give zero total heat, as CPU power is balanced with heat removal. It also implies the integrated Laplacian should be zero. In fact, this zero result has been observed both in our experiments and aforementioned simulation. Based on the discussion in the previous sub-section,

(2.7) can be approximated as:

$$-\kappa \int_{S_P} \nabla^2 T(x,y)dxdy \approx \frac{P}{\Delta z} \tag{2.8}$$

$$\kappa \approx \frac{P/\Delta z}{-\int_{S_P} \nabla^2 T(x,y)dxdy} \tag{2.9}$$

where S_P indicates area where negative-Laplacian of temperature is positive. Since die thickness Δz is constant, once negative-Laplacian map is obtained from temperature image, the equivalent thermal conductivity κ can be obtained. It basically means that the proportional factor κ can be acquired from dividing total power by thickness and by areal integration of the positive parts of negative-Laplacian. Having this κ, CPU power density map becomes straightforward, which is expressed as:

$$p(x,y) = \begin{cases} \kappa\Delta z[-\nabla^2 T(x,y)], & -\nabla^2 T(x,y) > 0 \\ 0, & -\nabla^2 T(x,y) \leq 0 \end{cases} \tag{2.10}$$

Using the above equations to estimate the power map for the homogeneous heat source example and the linear heat source example, the results are shown in Fig. 2.5. Fig. 2.5(c) and Fig. 2.5(d) are the estimated power densities for the two cases, while Fig. 2.5(a) and Fig. 2.5(b) are the corresponding original power density maps. As we can see, some spikes exist at corners in the estimation results due to numerical noise.

To compare the similarity of the two power maps, we introduce *2D correlation coefficient*, or simply *correlation* to evaluate the similarity between the real power map and the estimated power map, which is defined as

$$r = \frac{\sum_m \sum_n \left(A_{mn} - \overline{A}\right)\left(B_{mn} - \overline{B}\right)}{\sqrt{\left(\sum_m \sum_n \left(A_{mn} - \overline{A}\right)^2\right)\left(\sum_m \sum_n \left(B_{mn} - \overline{B}\right)^2\right)}} \tag{2.11}$$

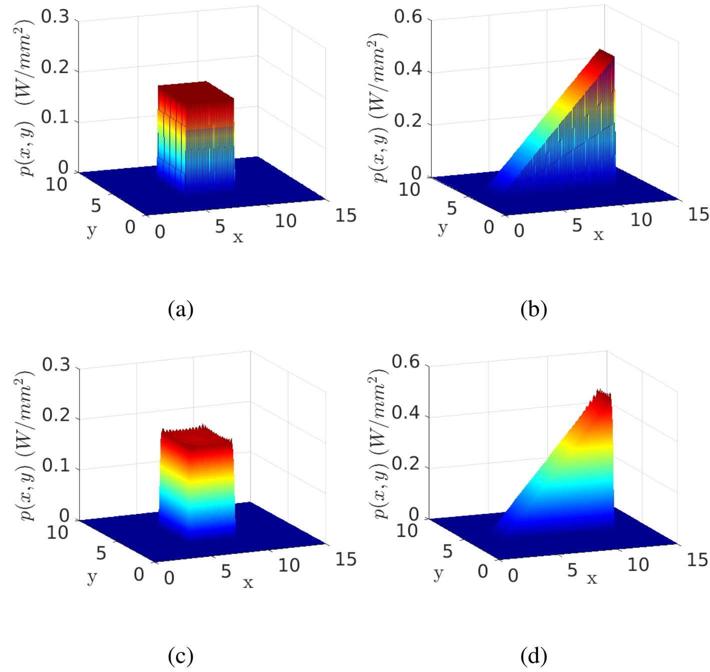


Figure 2.5: Comparison between estimated power density maps and exact ones for simple ideal examples. (a) and (b) are original power density maps for homogeneous heat source and linear heat source, respectively; (c) and (d) are the corresponding estimated power density maps of (a) and (b).

where $\overline{A}$ and $\overline{B}$ are mean of all entries in A and B, respectively. r is a scalar between 0 and 1, the more it approaches 1 the more they look alike. For the above two examples, the correlations of the first and second example are 0.977 and 0.973 respectively. In addition, RMSE of estimated power map on the active powered region is $0.005\text{W} \cdot \text{mm}^{-2}$ and $0.015\text{W} \cdot \text{mm}^{-2}$ respectively for the two cases as well. As a result, we can see that the proposed power map estimation method is quite accurate.

The thermal conductivity κ of silicon is about $130\text{W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}$, copper is about $400\text{W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}$. Due to the mixture of silicon, copper and some other materials in real die, the overall κ could be somewhere around $130 \sim 400\text{W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}$. The material in the motivation examples in

simulation has κ of $400\text{W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}$. In our case, the estimated κ by the proposed method is about $417\text{W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}$, about 5% error for the estimation.

One may wonder why the estimation error is 5% for κ based on those ideal simulation cases. There are several reasons. The first is the non-zero thickness of the chip hence the thermal distribution is not perfectly homogeneous along the vertical z-axis of chip die. The second reason is the nature of finite-element method, meaning elements are not infinitely-fine on the object. We can see some glitches on the power map surface. The side surfaces are not absolutely vertical but with a small angle from the vertical plane. Ideally, the sides should be absolutely vertical. Thirdly, a very small part of the low powered area that only has slight power density (e.g., smaller than $3 \sim 5\%$ of the average power) are computed as zero power because of ignoring the negative values in the raw power map. In reality, the above mentioned extremely low powered area may still show negative Laplacian values due to the heat diffusion behavior, such as vertical diffusion within the chip. To look at the error more closely, we repeat the trials with reduced thickness and the finest mesh structure in FEM in COMSOL for several ideal cases. The κ estimation error could be improved to within $\pm 1\%$ for ideal homogeneous heat sources (square or rectangle shaped, etc.), and 0$\sim$3% for the ideal linear heat source case. Further from this observation, the error is workload dependent due to the aforementioned third reason. As we see, the error is different for ideal cases. One consideration is the linear heat source case has more errors happening at the border of heat source, where there is a small part of places at the border with positive power are still shown as negative values in the simulation setup and hence are zeroed out. In another words, the proposed κ approximation method is not perfect but is a reasonable estimation.

Therefore, we have verified the approach of estimating the power map in simulations. Moreover, the estimated power density maps sufficiently match the original power setting. We further note that the estimation error for κ can be smaller or equal to 5% from the simple ideal cases. However, it is difficult to know the true accuracy since we do not know the actual κ for the commercial chips.

2.3.3 Thermal Conductivity Estimation for Real Chip Die Area

Another important parameter for the thermal model is the thermal conductivity of the chip. Based on the power map model derived in Sec. 2.3.2, we show in this section how to estimate the equivalent thermal conductivity κ of the die from the measurement of thermal maps.

For our work, the total power of CPU is also needed. Intel Performance Counter Monitor (PCM) provides users a software interface that estimates the internal resource utilization of the latest Intel core processors. One metric of the PCM dataset is CPU energy consumption between two accesses. To ensure precision, power data has to be synchronized with the thermal maps. As mentioned in the system setup, if the capturing frequency of infrared camera is f, PCM data should be recorded in this same frequency. Suppose the CPU energy along discretized time points is series E, then total power $P = E/\Delta t$ is also a time series, and $\Delta t = 1/f$.

One thermal map is related to one raw power map, and it will result in one κ value. Different thermal maps may result in different κ values. A reasonable κ should be a constant despite different workloads and time. In this work, estimating κ accurately is important to the thermal characterization for that the FEM thermal model depends on the κ parameter. It is also important to power map estimations due to the proportional factor, as seen in equation (3.1). And the accuracy

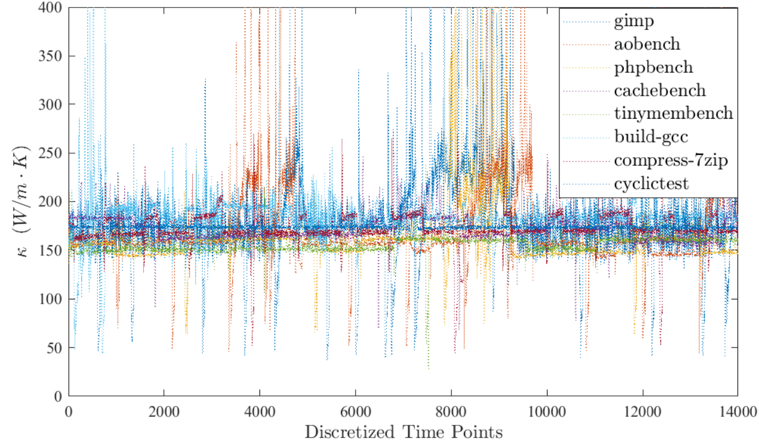


Figure 2.6: Estimated thermal conductivity of die area with respect to multiple workloads in time domain

of κ depends on the accuracy of thermal imaging measurements. Therefore, we have analyzed a sufficient amount of imaging samples regarding various workloads to obtain an optimal κ expectation. In this work, we execute eight workloads of different kinds and capture over 14000 thermal maps for each workload. The workloads are *gimp, aobench, phpbench, cachebench, tinymembench, build-gcc, compress-7zip* and *cyclictest*, respectively. The resulted κ with respect to different workloads along time line are plotted in Fig. 2.6. See from the traces, κ comes out quite constant and for half of the workloads and there is an obvious overlap, which is expected. On the other hand, it is observed that the mean κ of workloads *tinymembench*, *phpbench*, *aobench* and *build-gcc* are measured 160, 164, 165 and $180\text{W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}$, which have slightly large deviation compared to other workloads. In our experiment, the workloads are selected such that: (a) they contain steady states during the execution since this work focus on the steady state power and thermal estimations; (b) they invoke enough power to generate measurable heat and infrared emissions so that the infrared camera captures the chip's spatial temperature efficiently. As seen in the results, the κ seems workload dependent. Possible reasons are that the top surface temperature does not perfectly follow

the temperature inside the chip as the material is not perfectly homogeneous, as a result, the error is dependent on workloads and the thermal measurement error. Some glitches exist in κ due to CPU changing power levels thus not running in steady state. During those times, temperature transient term ($\rho C_P \frac{\partial T}{\partial t}$) cannot be ignored. Furthermore, the global arithmetic mean of κ is $174\text{W}\cdot\text{m}^{-1}\cdot\text{K}^{-1}$. We define this κ as the estimated equivalent thermal conductivity of CPU die.

In theory, κ is related to the whole system (die + FR4 (Circuit Board) base + motherboard + cooling pads). However, the die has much greater thermal conductivity than the FR4 base below it, which allows us to separate the die out of the system when computing κ. For instance, if the FR4 base and motherboard were metal or silicon with the same κ as the die, then we can't treat the chip as a thin piece anymore. The chip and boards should be treated as a homogenous object. Thanks to the fact that die has far greater κ than its base, we can approximate the κ=174W $\cdot$ m^{-1} $\cdot$ K^{-1} as the κ of silicon die. However, the resulting κ is higher than that of the pure silicon. This could come from the fact the metal layers inside of the die and the metal protection covering the surface of the die increases the overall κ.

2.3.4 FEM Model Architecture to Imitate Real Experiment Setup

In this subsection, we present the FEM thermal model that computes (reconstructs) the thermal map for the test setup with bare chip and back cooling of the processor and a novel approach to validate the results.

Once we obtain the estimated power density maps and equivalent thermal conductivity κ of the CPU die, we then start to build a heat transfer structure that mimics the real experiment setup. Fig. 2.7 illustrates the structure created using COMSOL Multiphysics which matches the

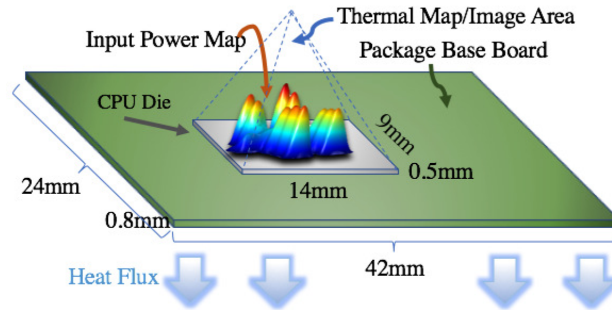


Figure 2.7: Thermal structure created to imitate the real experiment setup.

real CPU package geometry. Geometries of Intel i7-8650U are acquired from the open resource from WikiChip organization [Wik]. The CPU package dimension is 42×24×1.3mm, thickness of base circuit board is 0.8mm. There are two pieces of dies soldered on the base board, the CPU die, which is the object we will study, has dimension 14×9×0.5mm. In our model, material of the CPU die and package base board are initialized with silicon and FR4, respectively. However, thermal conductivity of die part (silicon) is set to the computed κ, i.e. $\kappa = 174\text{W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}$ in our case. A convective heat flux is applied to the bottom surface of the package base board, which simulates the heat flow from CPU package through motherboard to the cooling system, as indicated by the arrows at bottom in Fig. 2.7. The convective heat flux rate and thermal conductivity of base board will be determined as boundary conditions in the next.

For the FEM thermal simulation, we also need to know the correct thermal boundary conditions of the die. One idea is to explore the idle status of CPU (its boundary conditions are the same of the CPU under other workloads) as it is easy to extract the power map in this status. Specifically, since the idle status has extremely low power, power map is pre-known as almost zero, except for very few places that have slight power. Majority of CPU appears to be approaching ambient temperature, spatial temperature appears relatively flat. At the beginning, simulated thermal

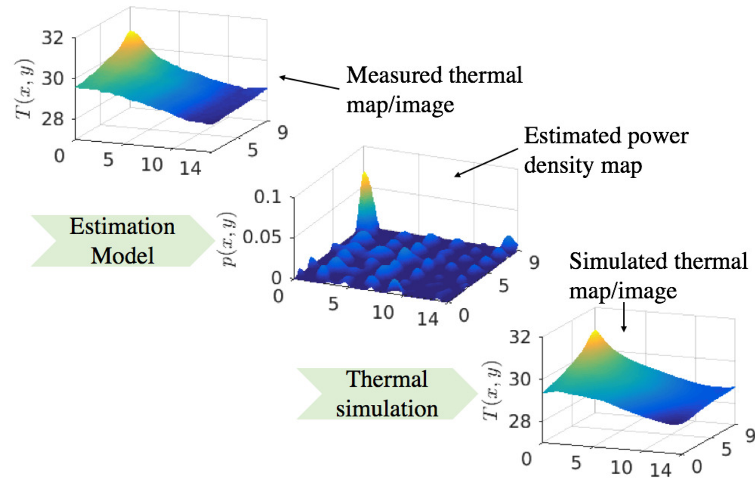


Figure 2.8: Setting proper boundary conditions for thermal simulation model using measurements of CPU's idle status

map according to the estimated idle power map has the same trend with the measured thermal map, whereas the amplitude and the range have a little discrepancy. This will guide us how to adjust the thermal conductivity of package base board and bottom convective heat flux rate such that the simulated thermal map matches the measured thermal map as much as possible for *idle* status, as shown in Fig. 2.8.

The boundary conditions for the FEM model are determined systematically and methodologically. Geometry dimensions are known and thermal conductivity of chip die has been calculated. We first feed the estimated *idle* power map to FEM and manually adjust the temperature of the bottom interface of the base board at somewhere around 30°C, making the FEM simulated thermal map to mirror the measured thermal map, as Fig. 2.8 shows. It needs a couple simple trials, once the simulated thermal map of the idle status mirrors the measured thermal map, temperature of base board bottom interface is then set and fixed. The idle power map has extremely low power across the chip only except for a corner area thus it is the best choice for setting the interface temperature.

Then we prepare an estimated power map of an arbitrary active workload at an arbitrary steady state during runtime. We feed that power map to the FEM model and adjust the thermal conductivity of base board again, making the temperature range of the simulated thermal map equal to that range of measured thermal map. This is because the thermal conductivity has a scaling effect to the temperature range. Finally, adjust the heat flux rate at the interface such that the max temperature equal the measured max temperature. **Note that this is one-time action for all, meaning these parameters are fixed in later tests for various workloads, including their various steady states of runtime.** For the setup with heat sink, the physical parameters are computed in a similar way. The difference is the usage of equation (4.3) to compute the heat transfer coefficient or heat flux from heatsink to ambient. In our simulations we found that these one-time calibrated parameters perform well for all the data samples. We found that the convective heat flux rate is about $600\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{K}^{-1}$ and thermal conductivity of base board is about $6\mathrm{W} \cdot \mathrm{m}^{-1} \cdot \mathrm{K}^{-1}$.

Note that this FEM thermal model is built to validate the accuracy of estimated power maps. As mentioned before, since the exact power map cannot be directly measured, we take an indirect method to to validate it by comparing the measured thermal maps to the FEM computed thermal maps. The power map accuracy results are described in details in Sec. 2.4.1.

2.4 Experimental Results and Discussions

In this section, we will present the results with the experiment of Intel i7-8650U in two stages, which has 4 CPU cores, and an integrated GPU. First, we present the estimated power map results for the 4-core processor and validate the estimated power density maps by finite element based thermal simulation with no heat sink using COMSOL Multiphysics as discussed ear-

lier [Mul14]. Then we compare it against a recently proposed power map estimation method [RDB18a]. In the next chapter, we presented the accurate thermal models for estimating full-chip thermal maps when the processor works in real situation under heat sinks.

2.4.1 Power Map Estimation and Comparison Results

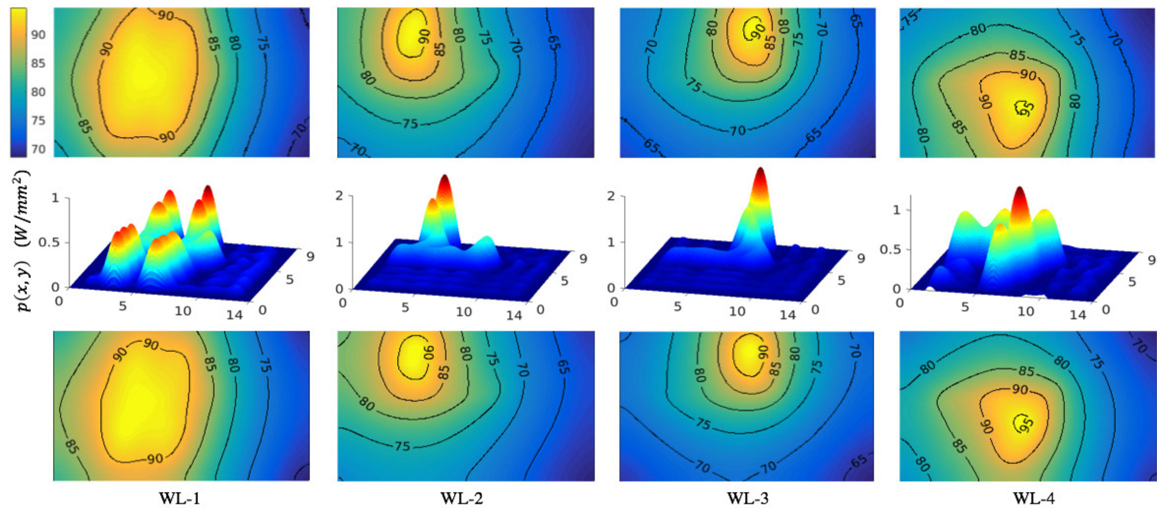


Figure 2.9: The power map estimation results, from first to last row are experiment measured thermal maps, estimated power maps in 3D view, and simulation generated thermal maps, respectively. Each column is related to one workload at one steady state.

We have firstly examined sufficiently large amount of data samples that relate to various steady states when with back side cooling. For most data samples, the estimated power maps are able to reconstruct the thermal maps that are almost identical to the measured thermal maps, with average similarities over 96%. Besides, the average absolute error for thermal maps reconstructed by estimated power maps and FEM model is 1.3°C. Fig. 2.9 lists four typical samples of the power map estimation results. The first row are the experiment measured thermal maps, the second row

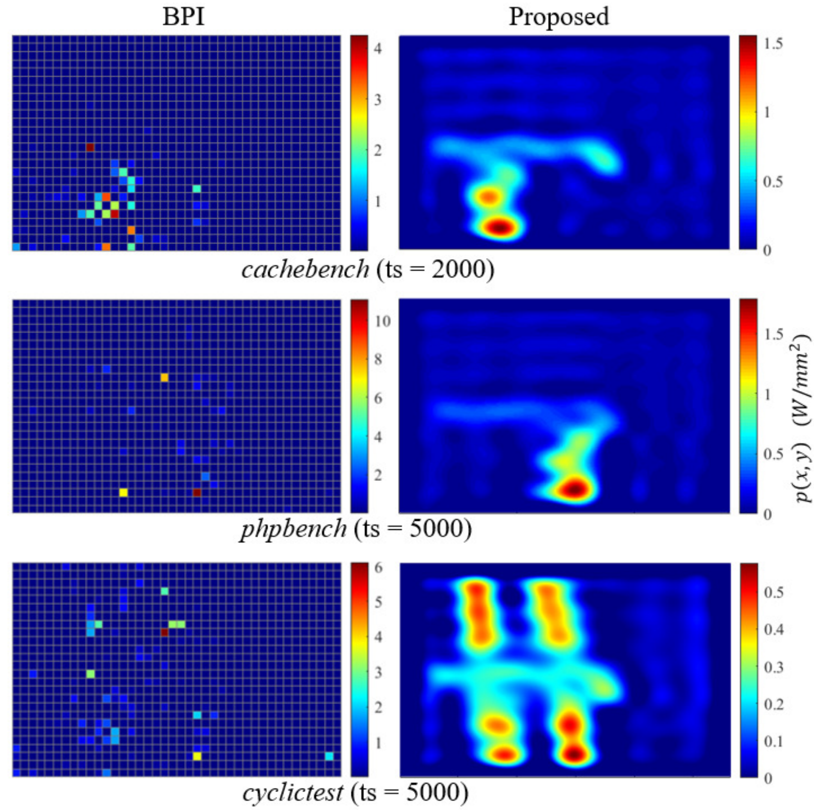


Figure 2.10: Comparison of the estimated power maps between the BPI (27×41 resolution) and the proposed method for three benchmarks (*cachebench*, *phpbench* and *cyclictest*) with respect to a specific time step during their full execution, respectively.

are the estimated power maps in 3D view, and the last row are thermal maps generated by FEM thermal simulation in the back-side cooling scenario.

As mentioned in the related works in Sec. 2.1, Reda *et al.* [RDB18a] proposed a general blind power identification, called *BPI* method for power maps from thermal measurements. For fair comparison, we utilize the open source code of BPI implemented in Matlab program [RB18] with our thermal measurements and compare it with the proposed method, which is also implemented in Matlab. We choose arbitrary steady-state time steps of workloads (*cachebench*, *phpbench* and *cyclictest*) as the target of power map estimation. Fig. 2.10 left-hand side illustrates the power maps

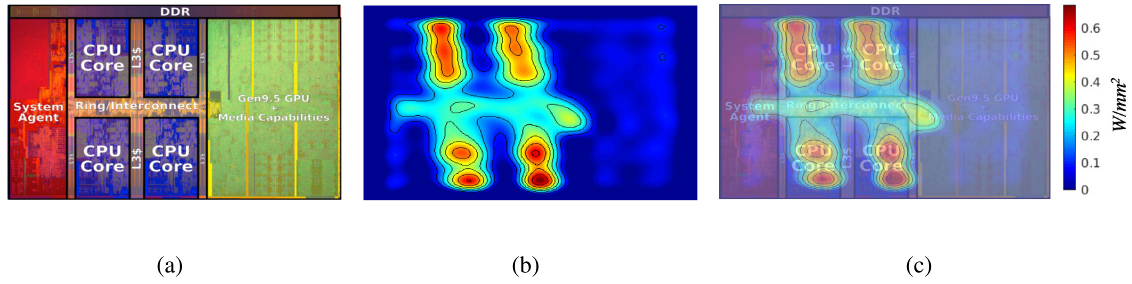
Table 2.1: Computational cost comparison for power map estimation

	BPI			Proposed
Map Resolution (pixel by pixel)	13×20	19×29	27×41	190×290
Time Per Single Power Map (ms)	39	277	740	7.5

under 27×41 resolution estimated by the BPI method, and the right-hand side illustrates the power maps under 190×290 resolution estimated by our method. It is apparent that the power patterns of the proposed power maps actually match the powered blocks in the power maps estimated through BPI. The total integrated power over the power maps equal to each other as well. The powered blocks may have excessive power density because of pixilation. In one word, the proposed method provides a more fine-grained estimation than the BPI method.

Furthermore, the proposed method is more efficient than the BPI method, as shown in Table 2.1. For BPI method to achieve high-resolution power maps, the size of required response matrices will grow exponentially as the resolution increases. Actually achieving resolution beyond 27×41 becomes challenging due to excessive computational costs of the underlying optimization. Note that we used Matlab tool to perform the computation on an Intel i7-8750H 2.2 GHz PC. As we can see, the proposed method is much more efficient even with higher resolution, which is essentially the resolution of the thermal image.

Fig. 2.11 further illustrates the power density distribution (Fig. 2.11(b)) projected on the processor die floor-plan (Fig. 2.11(a)) at an example steady state. Intel i7-8650U processor has a



(a) (b) (c)

Figure 2.11: (a) Intel i7-8650U processor die floor-plan [Wik]; (b) an estimated power density map; (c) projection of power density map onto the processor die floor-plan.

Table 2.2: Estimated power example for processor component (i7-8650U)

Component	Power	Component	Power
System Agent	0.70W	Ring/Interconnect	1.78W
GPU	0.59W	L3 cache	0.53W
Core#1	2.22W	Core#2	2.36W
Core#3	2.27W	Core#4	2.51W

system agent (including an image processing unit, a display engine and an I/O bus), a GPU module on the side and four cores in the middle. As we observe, the four cores consume the major power, whereas the system agent and GPU module consume low power at such steady state (Fig. 2.11(c)). We can see that power pattern aligns with the arrangement of cores and Ring/Interconnect quite well. We obtain the component power by power density integration for the component, which are presented in Table 2.2.

2.5 Summary

In this chapter, we address the problem of accurate full-chip power and thermal map estimation for commercial off-the-shelf multi-core processors. Processors operating with heat sink cooling remains a challenging problem due to the difficulty in direct measurement. We first propose an accurate full-chip steady-state power density map estimation method for commercial multi-core microprocessors. The new method consists of a few steps.

First, 2D spatial Laplace operation is performed on the measured thermal maps (images) without heat sink to obtain the so-called *raw power maps*. Then, a novel scheme is developed to generate the true power density maps from the raw power density maps. The new approach is based on thermal measurements of the processor with back-side cooling using an advanced infrared (IR) thermal imaging system. FEM thermal model constructed in COMSOL Multiphysics is used to validate the estimated power density maps and thermal conductivity.

Experiments on an Intel i7-8650U 4-core processor with back cooling shows 96% similarity (2D correlation) between the measured thermal maps and the thermal maps reconstructed from the estimated power maps, with 1.3°C average absolute error. Furthermore, the proposed power map

estimation method achieves higher resolution and at least $100\times$ speedup than a recently proposed state-of-art Blind Power Identification method. This work is published in [ZSO$^+$21].

Chapter 3

Power Estimation for Commercial TPUs

3.1 Power Map Estimation Framework

A brief overview of the proposed approach will be presented in this section, along with a description of the thermal setup used for collecting the necessary data from a commercial off-the-shelf TPU chip while it is under the workload.

3.1.1 Estimation Flow Overview

The proposed approach involves three engineering phases. First, we obtain full-chip power map measurements across the TPU with both high accuracy and resolution by implementing a state-of-the-art thermal-to-power technique. Second, we propose to take advantages of the hyper-parameters of the NN workloads that inferences on the TPU as the model's input features, and the outputs are power maps across the TPU immediately. Last but not least, the new model employs a special Generative Adversarial Network architecture called Conditional GAN or CGAN to train the online power characterization model.

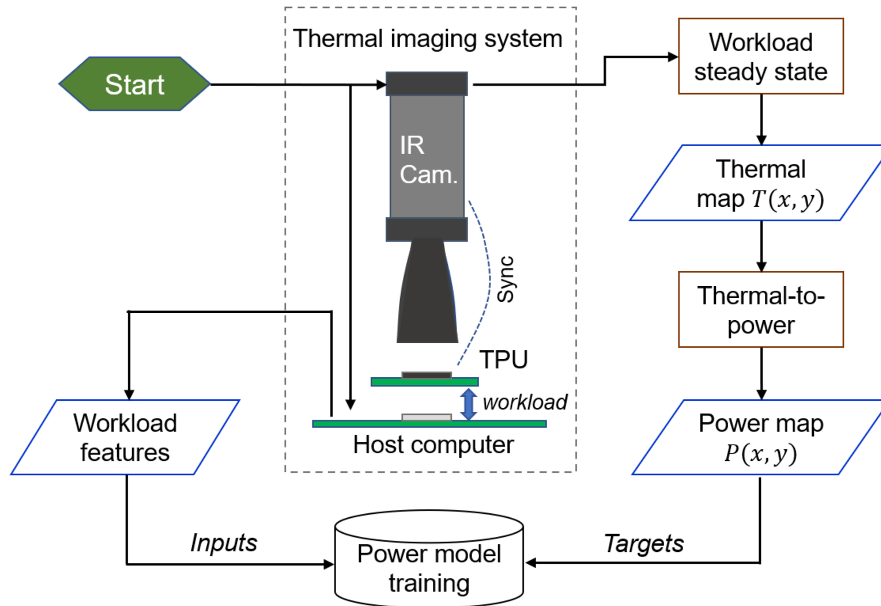
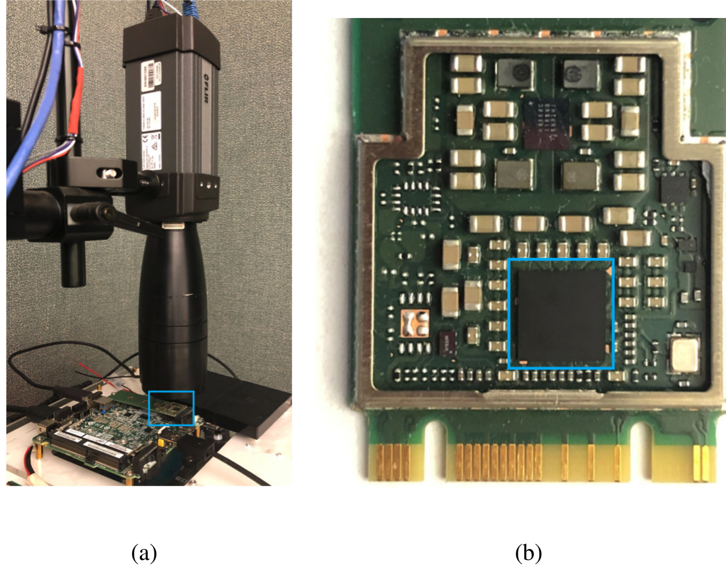


Figure 3.1: Framework and data acquisition flow

The CGAN-based power model requires two chunks of data for the training procedure, one is the off-line measured power maps when TPU is under load, which are used as targets when training the model. The other is a set of hyperparameters extracted from the NN workloads to be executed on TPU. It should be noted that those hyperparameters can be extracted either online or prior to the workload execution. Once the model is trained, we can use it for online TPU power inferencing. Fig. 3.1 illustrates the framework and data acquisition flow of the proposed approach. The first and the second phase, including every step shown in Fig. 3.1 will be described in detail in the next section. The third phase will be explained in Section 3.3.



(a) (b)

Figure 3.2: (a) Thermal Imaging system setup (b) TPU chip under-test, Coral M.2 TPU module. TPU module is shown in the blue box.

3.1.2 Thermal IR Imaging System

The proposed machine-learning based approach relies on proper data acquisition of the chip-level spatial power information from the TPU under workloads for the model's training and testing procedure. Directly measuring the power maps of TPU chips is not achievable. To address this issue, we indirectly measure the full-chip power map through a thermal-to-power approach proposed in [ZSJT20]. This thermal-to-power approach has both high precision and resolution. It calculates the full-chip spatial power density maps from the spatial temperature maps measured when the processor is under a thermal steady state. The approach basically takes advantage of the high correlation between heat source distribution and the 2D Laplacian transformation of the temperature of the chip surface.

The thermal-to-power approach requires accurate measurements of chip surface temperature maps. Hence, we have a built-in advanced infrared (IR) thermal imaging system, as shown

in Fig. 3.2. In order to expose the TPU chip to the IR camera, we have removed the stainless steel cover on top of the TPU. It should be noted that in our test case, the TPU module requires no external cooling kits, such as heat sinks, to ensure proper thermal conditions for the TPU to work. However, for some other TPU modules that come with heat sinks, after removing the heat sinks, the TPU module can be cooled with a widely used back-side liquid cooling technique [AH15] to ensure proper thermal operating conditions. The back-side liquid cooling approach features a thermoelectric (Peltier) device mounted on the PCB directly beneath the processor module allowing it to be cooled from underneath. This leaves the front side of the processor fully exposed to the IR camera without any interference layer in between.

Product information and specs of our IR thermal imaging system are described as follows. The IR camera used in this setup is a FLIR A325sc which supports a maximum imaging resolution of 320×240 pixels (px) with 16-bits of precision per px, and a maximum capturing frequency of $60\mathrm{Hz}$. The IR sensor is factory calibrated for accuracy across the temperature range of $-20°\mathrm{C}$ to $120°\mathrm{C}$, and resolves the IR spectral range of $7.5\mu m$ to $13\mu m$. A high-resolution microscope lens is used to achieve the spatial resolution of $25\mu m$ per px.

3.2 Data Preparation and Feature Selection

In this work, we model the spatial power from the workload features that are available in real time. Like any other regression model, the machine-learning model architecture we deploy is a supervised learning model, for which the proper data set is ultimately important. As previously mentioned, the data required for training the learning-based model involves measuring the offline power maps across the TPU full-chip and collecting the hyperparameters of NN workloads running on the

TPU module. It should be marked that each individual workload has a unique hyperparameters-powermap data pair, which serves as a unique training data point. Google Coral Edge TPU has 3 different frequencies, $500\mathrm{MHz}$, $250\mathrm{MHz}$, and $125\mathrm{MHz}$. Our workload mainly runs in $500\mathrm{MHz}$. In this section, the detailed process of acquiring the necessary data is presented.

3.2.1 Offline Power Map Acquisition

There have been various post-silicon approaches transforming thermal distribution to power distribution [WFMS09, CNR10, PSSK13, NWR13, BBVB16, RDB18b, ZSJT20]. Among those [SZZ+20b, ZSJT20] suits for our study case best, giving it calculates spatially continuous and relatively precise power maps from thermal maps with high efficiency, which is suitable for real-time inferences.

Considering the steady state 2D spatial thermal distribution of the processor as $T(x, y)$, where (x, y) is the coordinates of the thermal map. Power map can be approximated as [ZSJT20]:

$$p(x,y) \approx \begin{cases} k[-\nabla^2 T(x,y)], & -\nabla^2 T(x,y) > 0 \\ 0, & -\nabla^2 T(x,y) \leq 0 \end{cases} \tag{3.1}$$

with

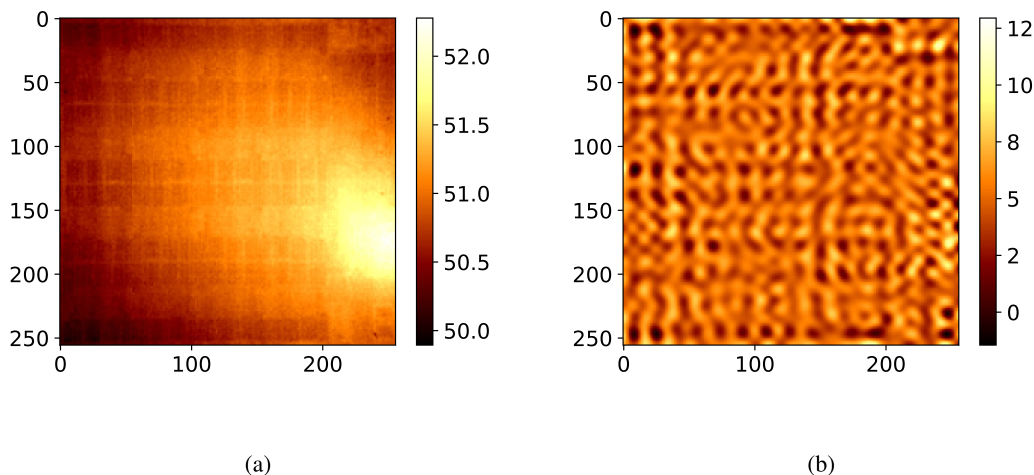
$$k = \kappa \Delta z \tag{3.2}$$

where $p(x, y)$ stands for the spatial power map (density, Watt/area), κ and Δz for thermal conductivity and chip thickness, which are constants. And $\nabla^2 T(x, y)$ is the 2D Laplacian of temperature. The coefficient k is expressed by:

$$k = \kappa \Delta z \approx \frac{P}{-\int_{S_P} \nabla^2 T(x,y) dxdy} \tag{3.3}$$

where S_P indicates the area where the negative-Laplacian term of temperature $[-\nabla^2 T(x,y)]$ is positive. The negative-Laplacian term reflects the pattern of spatial power distribution. In this work, we call k the thermal-to-power coefficient. It can be calculated by the thermal measurement of idle status $T_{idle}(x,y)$ combined with standby total power consumption P_{idle} provided by the official specification. Once we have $T_{idle}(x,y)$ and P_{idle}, we can substitute them for equation (3.3) to obtain k. After k is obtained, power maps under any workloads can be acquired straightforwardly through equation (3.1) from thermal measurements.

As we see, one special requirement of this method is that it needs the processor to be under thermal steady-states when calculating its power maps from thermal measurements. Hence, to satisfy this requirement, we have the TPU module run each workload for sufficiently long (e.g. 2 minutes per workload) to stabilize its temperature. Multiple thermal images are captured after the TPU reaches a steady state corresponding to that workload. Once the steady-state thermal images are obtained, they will be processed to calculate an averaged power map for that workload as the estimation target of that data point. It should be noted that the proposed learning-based power model does not require any waiting during deployment since the proposed model needs no thermal measurements for power estimations. As an example, Fig. 3.3(a) shows an averaged steady-state thermal image for MobileNet-V2-224-1.0 network, which is a widely used image-classification model available on TensorFlow [A$^+$15]. Fig. 3.3(b) further illustrates the resulting TPU power map. In order to automate the measuring procedure, we arranged a sequence of workloads for the TPU module, and in the meantime, the IR camera is synchronized with the TPU for each workload's running period.



(a)

(b)

Figure 3.3: (a) Thermal image and (b) power map of TPU for MobileNet-V2-224-1.0 network.

The size of the TPU module is about $5.06 \times 4.94\mathrm{mm}$. And it only occupies partial camera's $320 \times 240\mathrm{px}$ field of view. We crop the chip area out of the photo and then calculate the power map. Thermal noise is a big problem when we need to calculate the Laplacian. Although the noise is small relative to the temperature, its Laplacian can be locally larger than the Laplacian of temperature, overshadowing useful information. An effective method to extract information is the discrete cosine transform (DCT) [ANR74]. The majority of the information is contained in low-frequency coefficients of DCT. Therefore, we transform the heatmaps into the spatial frequency domain by 2D DCT, keeping the low-frequency coefficients, and then transform them back. This reduces some resolution but allows us to analyze the spatial distribution of power.

3.2.2 Feature Selection Considering TPU Workloads

For neural networks such as TPU's workloads, power distribution is an immediate reflection of hardware resource utility invoked by the neural networks executing on the TPU. TPU

hardware resources that the network demands are tightly related to the network model architecture, size, operations, etc. Hence, we are able to characterize TPU's power from the workloads' hyperparameters such as operation type, count and workload size, etc.

In this work, we divide the network's hyperparameters as features into two groups, called the *overall features* and *operational statistics*. Neural network models that are coded to run on CPU need to be compiled to a TPU readable version. In our study, EdgeTPU Compiler [Edg] is employed to transform a CPU version network model to a TPU version. On the one hand, the overall features such as model size and memory usage are recorded through the process. On the other hand, we collect statistical information for the type and count of operations indicated by the network in the meantime. We mark that for different TPUs different tools may be involved, however, those network information should always be reachable. Today's world has a vast number of neural networks and hundreds of kinds of operations. To find the most popular operational features of network models, we explored a number of the most popular and widely used open-source deep neural network models from TensorFlow. The selection of models will be explained in more detail in Section 3.4. Table 3.1 shows the 31 features selected for the network workloads.

3.3 CGAN-Based Estimation Model

3.3.1 Review of CGAN

As a machine-learning problem, our purpose is to generate the on-chip power image from the workload features. Generative Adversarial Network (GAN) can be a competitive choice [GPAM+14].

Table 3.1: Selected Workload Features (Coral M.2 TPU, Google Edge)

Overall			
image_shape	pooling_mode	onchip_mem_rem	num_op_tpu
width_multiplier	model_size	offchip_mem_used	num_op_cpu
depth_multiplier	onchip_mem_used	total_op_cnt	infer_time
Operational Statistics			
add	full_connect	pad	reduce_max
avg_pool_2d	l2_norm	quant	relu
concat	max_pool_2d	reshape	strslc
conv2d	mean	sft_max	hard_swish
depconv2d	mul	sub	

GAN has two contrary networks, generator G and discriminator D. G is trying to map an input vector to an output image, while D attempts to tell if an image comes from the real data or G. They will be trained simultaneously and keep trying to optimize themselves to fool/expose others. At last, when the generated image is close enough to the ground truth, the generator should have become a mastered projector, and the discriminator can never tell the difference between a fake and a real image.

Original GAN is used to produce new images within the range of existing image distribution and the generator is fed by noise. As a variant, Conditional GAN (CGAN) also give some labels to the generator, so it can map features to corresponding images [MO14]. Based on CGAN, we no longer use random noise because we expect to give a unique power distribution with one certain feature vector.

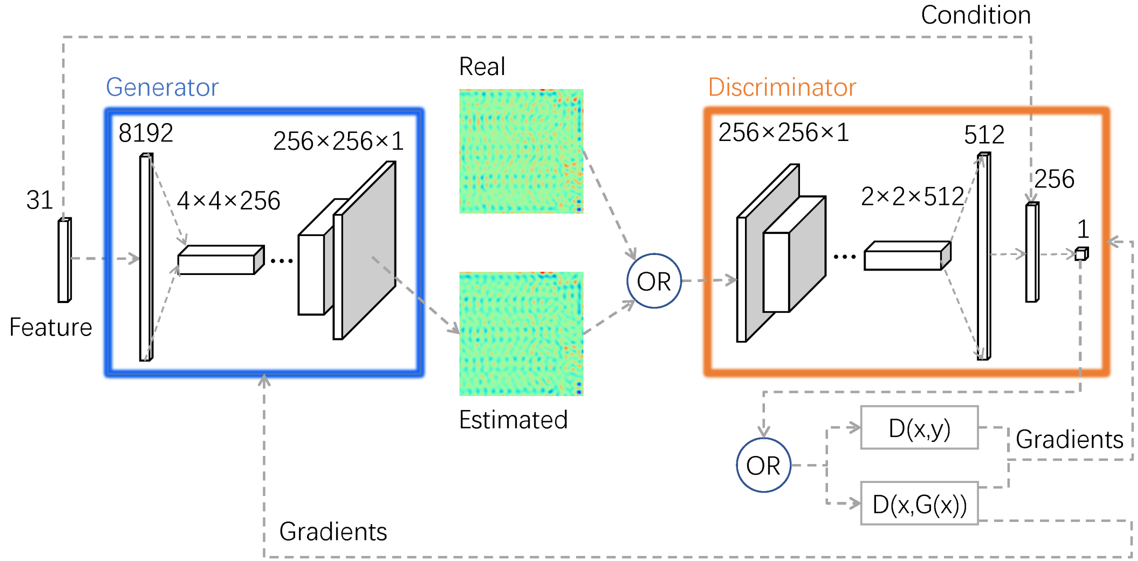


Figure 3.4: Architecture of CGAN model

Sometimes it can be tough to train the GAN model because of the gradient vanishing. We can introduce Wasserstein Distance instead of the conventional JS-Divergence to measure the similarity between the distributions of real and fake images [ACB17]. This modification can stabilize the training process and reduce the frequency of collapses.

3.3.2 Proposed CGAN-Based Power Estimation Framework

The framework of our hyper-parameter to power map model is shown in Fig. 3.4. The input condition x is a 1x31 vector, which will be given to both the generator G and the discriminator D. The generator learns how to map it to the correct power map image y, and produce $G(x)$. Then D accepts y or $G(x)$ alternatively with the condition x, score a degree $D(x, y)$ or $D(x, G(x))$ how confident the given power map y or $G(x)$ is true. Our goal is to maximize $D(x, y)$ and minimize

$D(x, G(x))$ over all (x, y) pairs in the training set. We can write down the objective function to minimize as:

$$\begin{aligned} loss_D =& \mathbb{E}_{(x,y)}[D(x, G(x)) - D(x, y)] + \\ & \lambda_{gp}\mathbb{E}_{\hat{x}}[(||\nabla_{\hat{x}} D(\hat{x}, x)||_2 - 1)^2] \end{aligned} \tag{3.4}$$

Here $\mathbb{E}_{(x,y)}$ is the expectations over the (x,y) pairs in the training set. Also, we introduce an extra gradient penalty term, so that the discriminator has the 1-Lipschitz continuity [ACB17]. $\hat{x}$ is the interpolation between $G(x)$ and y, and λ_{gp} is the weight.

For the generator, we want to maximize $D(x, G(x))$ and minimize the L2 loss $||y - G(x)||^2$. The generator has nothing to do with the real power map y, so there is no $D(x, y)$ term. The loss function is:

$$loss_G = \mathbb{E}_{(x,y)}[-D(x, G(x)) + \lambda_{L2} \cdot ||y - G(x)||^2] \tag{3.5}$$

The architecture and parameters of the generator and discriminator networks are shown in Table 3.2.

First, the generator transforms the input condition vector into an image by a fully connected layer and a reshape operation. After that, there are 6 transposed convolutional layers to finally produce a 256×256px power map. The discriminator is a conventional convolutional classifier that has a similar but reversed structure with the generator, and goes from a 256×256px image to only one real number as the output.

Table 3.2: Architecture and Parameters

Generator			
Layer	Kernel	#Output	Activation
FC	-	8192	Leaky ReLU
Reshape	-	4x4x512	-
Conv_trans	5x5	8x8x512	Leaky ReLU
Conv_trans	5x5	16x16x512	Leaky ReLU
Conv_trans	5x5	32x32x256	Leaky ReLU
Conv_trans	5x5	64x64x128	Leaky ReLU
Conv_trans	5x5	128x128x64	Leaky ReLU
Conv_trans	5x5	256x256x1	-
Discriminator			
Layer	Kernel	#Output	Activation
Conv	5x5	128x128x64	Leaky ReLU
Conv	5x5	64x64x128	Leaky ReLU
Conv	5x5	32x32x256	Leaky ReLU
Conv	5x5	16x16x512	Leaky ReLU
Conv	5x5	8x8x512	Leaky ReLU
Conv	5x5	4x4x512	Leaky ReLU
Conv	5x5	2x2x512	Leaky ReLU
FC	-	512	Leaky ReLU
(+Cond) FC	-	256	Leaky ReLU
FC	-	1	-

3.4 Experimental Results and Discussions

In this section, we demonstrate the experimental results of the proposed approach in two folds. On the one hand, we convince that the power maps obtained through the thermal-to-power way are sufficiently reliable. On the other hand, we show that the online inferencing of power maps by the proposed CGAN-based model is computationally efficient and technically sound.

3.4.1 Validation of the Total Power Consumption

As discussed in Section 3.2.1, directly measuring the spatial power distribution of the TPU is not realistic, we have implemented one of the state-of-the-art methods that compute spatial power maps from the thermal measurements. The question is that whether those inter-mediately obtained power maps are sufficiently reliable in our test case. Fortunately, Coral has open-sourced a few but limited data for total power measurements. Hence, we are able to compare our estimated total power with the manufacturer's provided total power measurements, to see how well they match. The more they match, the more convincing they are. To our knowledge, this indirect way is the best way for validating the power maps in our case.

The estimated total power is simply an integration from the estimated spatial power maps. Coral M.2 TPU module's official specification has released 7 power measurement data points, pertaining to two different workloads under three different operation frequencies, respectively, plus an idle power measurement. Power under idle status is officially indicated by a range between 0.375 and 0.400W. In our work, we take the central value 0.3875W as its golden idle power. Then combine that with the thermal maps captured under idle status to calculate the thermal-to-power

Table 3.3: Total Power Comparison

Workloads	Total Power	500 MHz	250 MHz	125 MHz
MobileNet V2	Real	1.4 W	0.9 W	0.6 W
	Est.	1.42 W	0.92 W	0.60 W
Inception V3	Real	0.7 W	0.6 W	0.5 W
	Est.	0.69 W	0.58 W	0.50 W

coefficient k. Then we use it to calculate the power maps for the same two workloads at those three different frequencies, and further their total power.

By combining two models and three operating frequencies, Table 3.3 lists the six golden total power data points, which are provided from the official specification, with our estimated total power consumption. As we can see, all of the estimated total power data points mirror the real power measurements remarkably well. Given that one decimal point precision is available in official power data, the root-mean-squared-error of total power estimation is only $0.0147\mathrm{W}$, and the percentage error is within 2%.

3.4.2 Power Map Estimation Accuracy

The dataset consists of a number of well-known neural network models for image recognition, such as EfficientNet, InceptionResNet, MobileNet, etc. By varying their architectural hyperparameters, many of their variants were generated and added to the dataset. The final dataset has 7066 data points (networks) in total, where 6359 points are randomly selected for training and 707 points for testing. All networks are executed with the TPU at the nominal frequency $500\mathrm{MHz}$.

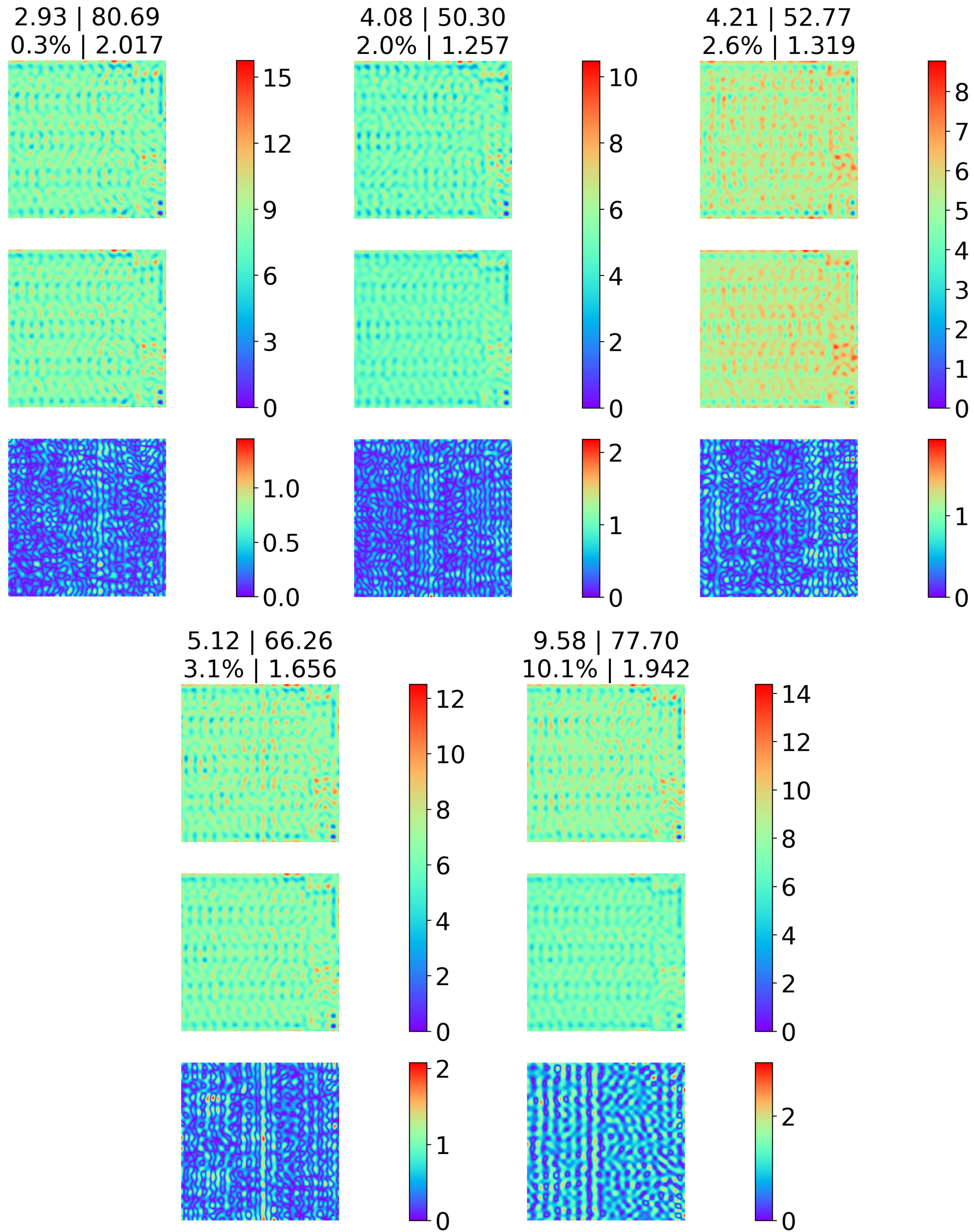


Figure 3.5: Measured power maps (row #1), estimated power maps (row #2), and error maps (row #3). The numbers in the first row indicate the *Power Density RMSE* | *Average Power Density* (unit: $\mathrm{mW/mm^2}$). And numbers in the second row indicate the *Total Power Percentage Error* | *Total Power* (unit: W).

After the training process, the generator of the CGAN model is able to estimate the power map with the input hyperparameters. To characterize its accuracy, we calculate the root-mean-squared error (RMSE) over each pixel between the generated power map and the measured power map.

In our dataset, the power density ranges from 0 to $189.34\mathrm{mW/mm^2}$. The averaged RMSE of the power map estimation on the test set is $4.98\mathrm{mW/mm^2}$ with a standard deviation of only $2.53\mathrm{mW/mm^2}$. The results are quite accurate considering the data range. Fig. 3.5 compares the estimated and the measured power maps with some examples from the test set. It should be noted that the right-most column shows the worst estimation on the test set, which is about 10% percentage error on the total power. It can be seen that the CGAN-based model has learned the contour of real power remarkably well.

The power map can also be used to calculate the total power by simply integrating the power density over the power map. The mean error of the total power is $0.0968\mathrm{W}$. Considering that the average total power is $1.375\mathrm{W}$ on the test set, these estimations of total power are sufficiently accurate as well.

3.4.3 Computational Efficiency

Training procedure normally takes a few to a dozen hours to complete. Once the generator is well-trained, it can be deployed for real-time power prediction. The average inference time we measured in our experiments is $6.9\mathrm{ms}$, with Intel Core i7-10710U as the host board and the Coral Edge TPU. This low latency ensures the effectiveness of real-time power estimation. On the one hand, most of the models on the TPU have a single inference time of well over $6.9\mathrm{ms}$ (they take dozens or even hundreds of milliseconds). On the other hand, TPUs do not switch deployed neural

networks frequently, and those applications themselves that switch the neural networks generally take more time. As a result, the proposed model is sufficiently rapid to keep up with the TPU.

3.5 Summary

In this chapter, we propose a novel approach for the real-time estimation of chip-level spatial power maps for commercial Google Coral M.2 TPU chips based on a machine-learning technique for the first time. The new method can enable the development of more robust runtime power and thermal control schemes to take advantage of spatial power information such as hot spots that are otherwise not available. Different from the existing commercial multi-core processors in which real-time performance-related utilization information is available, the TPU from Google does not have such information.

To mitigate this problem, we propose to use features that are related to the workloads of running different deep neural networks (DNN) such as the hyperparameters of DNN and TPU resource information generated by the TPU compiler. The new approach involves the offline acquisition of accurate spatial and temporal temperature maps captured from an external infrared thermal imaging camera under nominal working conditions of a chip. To build the dynamic power density map model, we apply generative adversarial networks (GAN) based on the workload-related features.

Our study shows that the estimated total powers match the manufacturer's total power measurements extremely well. Experimental results further show that the predictions of power maps are quite accurate, with the RMSE of only $4.98\text{mW}/\text{mm}^2$, or 2.6% of the full-scale error.

The speed of deploying the proposed approach on an Intel Core i7-10710U is as fast as $6.9\mathrm{ms}$, which is suitable for real-time estimation. This work is published in [LZJ$^+$23].

Chapter 4

Full-Chip Thermal Characterization With Heat Sink Cooling

4.1 Identify Exact Locations of On-Chip Thermal Sensors

Typically, each core of the commercial multi-core processor contains at least one thermal sensor. However, the exact locations of these sensors are not disclosed by the chip vendors or developers publicly.

The exact location of those thermal sensors are needed to be known in order to model the full-chip thermal map which is obscured by the heat sink. In this section, we present our novel method to identify the exact locations of those thermal sensors, which serves as a basis for the following sections.

Thermal sensor values are accessible in the runtime through online CPU tool PCM. In order to locate the embedded thermal sensors, one way is to measure the temperature image of the

chip without heat sink and find the pixel location that matches the sensor value. However, there are two obstacles. The chip's top surface temperature measured by the thermal imaging system is lower than the internal temperature measured by the sensors. Besides, the difference between externally and internally measured temperatures is unknown.

To address this issues, we propose a correlation-based method to identify the exact sensor locations. First, we capture a series of temperature images of the chip without heat sink while it is running under workloads. Workloads activate different cores and heat up different places of the chip earlier or later, which enable all the sensors to have different temperature records along the timeline. To be more specific, if the recording time is sufficiently long, each different location of the chip will show distinct time curve of temperature measurements. Then we divide each of the measured temperature images into 5$\times$5 pixel square blocks, let the average temperature of this square be the temperature of this location. One pixel is 50μm wide thus the resolution is 250 $\times$ 250μm block. This resolution is fine enough considering the chip size. We mark the location of each block and trace their temperature along the time from all the measured temperature images. After collecting the temperature series of each pixel block, each series is compared against the sensor values. Although externally measured temperatures are lower than sensor values, tendency of both variations will be the same, meaning the external temperature rises or falls as the internal temperature does. Therefore, the temperature series on the exact sensor location will have the highest correlation with the sensor values. Linear correlation is applied, which is defined as

$$corr(T_m, T_s) = \frac{\mathrm{E}\left[(T_m - \overline{T_m})(T_s - \overline{T_s})\right]}{\sigma(T_m)\sigma(T_s)} \tag{4.1}$$

where T_m and T_s are the temperatures measured by thermal system and sensors, respectively. $\overline{T_m}$ and $\overline{T_s}$ denote the mean and $\sigma(T_m)$ and $\sigma(T_m)$ denote the standard deviations.

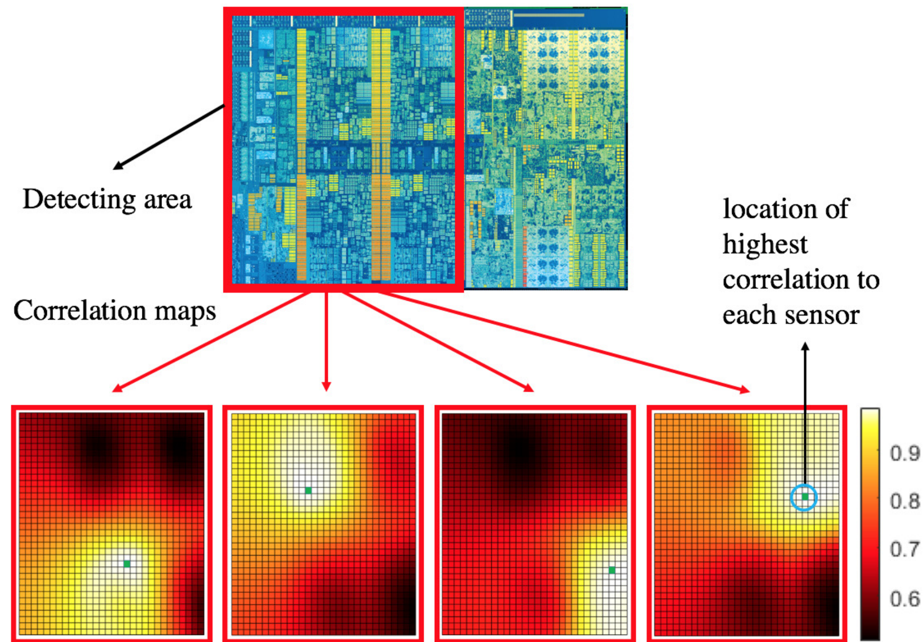


Figure 4.1: The correlation between pixel blocks and sensor values of each core. In each correlation map the pixel block that has the highest correlation to the core sensor measurement is marked in green dots, which infers to the on-chip sensor location.

Correlation maps in Fig. 4.1 illustrate the correlation between the temperatures of pixel blocks and sensor values of each core. Pixel block having the highest correlation in each correlation map, marked as a green dot, indicates where the on-chip sensor location is identified.

Furthermore, the identified sensor locations for all cores are illustrated in Fig. 4.2 as small colored squares, where the black rectangles outline the core regions. The chip layout is sourced from the open source Wikichip Organization [Wik]. When overlapping and aligning the cores we can see that the sensor locations identified in all the cores have quite good consistency.

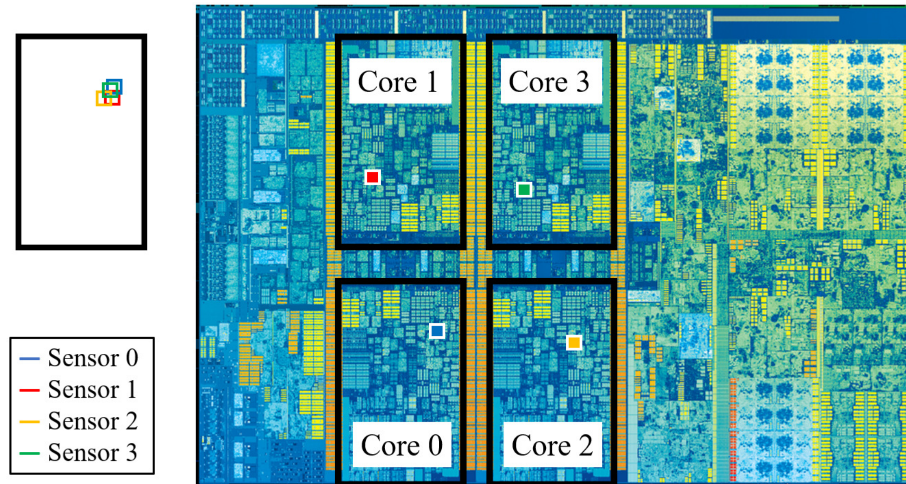


Figure 4.2: The sensor locations of CPU cores are identified in colored squares on an Intel i7-8650U quad-core processor [Wik].

4.2 FEM Thermal Modeling With Heat Sink

We have elaborated on the FEM architecture that imitates the bare chip and back cooling situation in Sec. 2.3.4. As we know, under real working situation, the back side liquid cooling is replaced by the top side heat sink cooling (either passive or active heat sink). The processor on device setup without and with heat sink is shown in Fig. 4.3. Fig. 4.4(a) illustrates the FEM structure of the processor setup and Fig. 4.4(b) illustrates the FEM structure of processor area in a 3D transparent view when it is covered by a fin-shaped heat sink, where the dimensions of the heat sink strictly follow the object in the experiment. Compared to the previous one, this FEM model replaces the back side heat transfer with heat transfer through the fin-shaped heat sink.

We know that the computed thermal map cannot be compared against the exact thermal map since the chip is obscured by heat sink and not measurable through thermal imaging system. However, the results can still be validated in a sense if the computed temperatures on thermal sensor

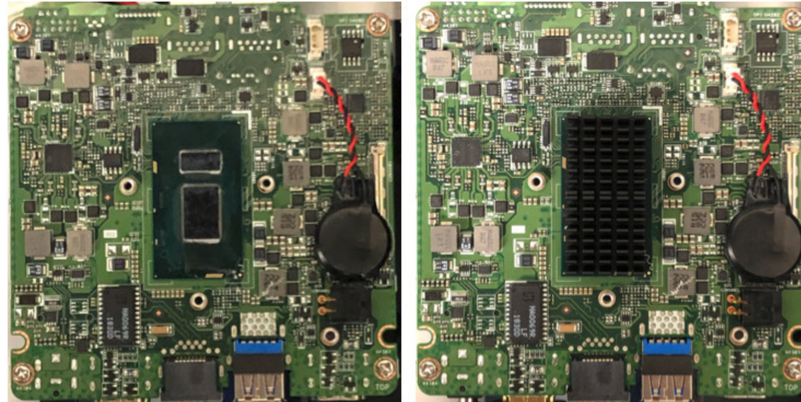
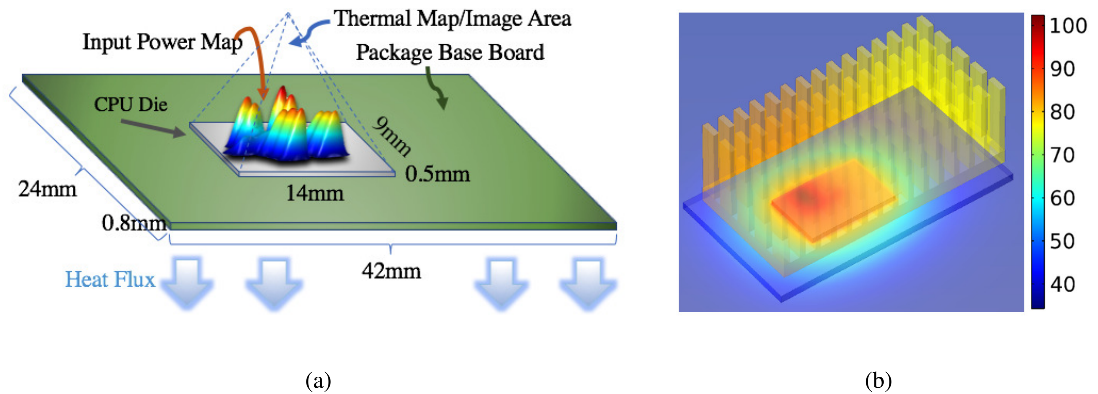


Figure 4.3: Processor without heat sink (left) and with heat sink (right)



(a)

(b)

Figure 4.4: (a) Thermal structure created to imitate the real experiment setup (without heat sink); (b) transparent view of processor area when with heat sink mounted on.

locations match the real on-chip sensor measurements. We compute the thermal maps by FEM based thermal simulation using COMSOL Multiphysics. The FEM thermal model architecture basically consists of four major components to simulate the real device setup - motherboard, processor base board, processor die itself and the heat sink. We remark that the model can be customized to any setup such as adding heat spreaders or using more sophisticated heat sinks.

Ambient temperature and the convective heat transfer rate of the heat sink to ambient are critical environment information for the FEM thermal model. To find out these parameters, we again include the temperature measurements of processor's idle status with heat sink in our analysis. The heat transfer per unit surface through convection can be expressed as:

$$q = h_C A \mathrm{d}T \tag{4.2}$$

where q is the heat transferred per unit time (W), which can be approximate as the processor power. A denotes the surface area of heat sink (m^2), h_C denotes the convective heat transfer coefficient ($\mathrm{W} \cdot \mathrm{m}^2 \cdot^\circ \mathrm{C}^{-1}$) and $\mathrm{d}T$ is the temperature difference between the heat sink and ambient. We use the processor power and average sensor temperature when the processor is in idle status and under workloads to compute h_C according to:

$$\begin{aligned} p_{idle} &= h_C A \mathrm{d}T_{idle} \\ p_{wkld} &= h_C A \mathrm{d}T_{wkld} \\ p_{wkld} - p_{idle} &= h_C A (T_{wkld} - T_{idle}) \end{aligned} \tag{4.3}$$

The average processor power when all cores are under workload p_{wkld} and in idle status p_{idle} are about 10.5W and 0.6W, and the corresponding average sensor temperature are T_{wkld} = 100°C and T_{wkld} = 44°C, respectively. The surface of heat sink is measured as $60cm^2$. Hence the

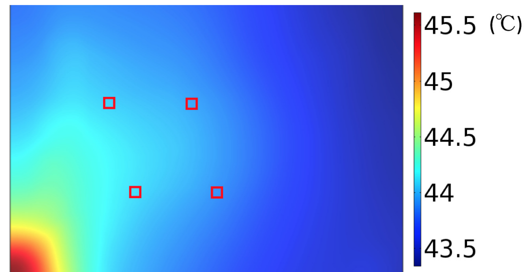


Figure 4.5: The computed temperature of processor with heat sink under idle status. The small red boxes mark the on-chip thermal sensor locations.

convective heat transfer coefficient h_C is acquired as $29.5\mathrm{W} \cdot \mathrm{m}^2 \cdot ^\circ \mathrm{C}^{-1}$. As before, the ambient T_A around the heat sink can be obtained using idle status by adjusting the ambient in FEM simulation such that the simulated sensor location values match the real sensor values for idle status. Fig. 4.5 shows the computed thermal map of idle status under heat sink, in which the computed sensor location temperatures match the real sensor measurements well when ambient is 33°C (306K). Therefore, the FEM thermal model for the processor setup with heat sink mounted is properly built.

4.3 Ensuring the Same Power Density Maps for Both Cooling Conditions

One important aspect of our thermal modeling methodology is that we need to ensure that the power density maps obtained with back cooling and the one with heat sink cooling should be keep as the same as possible. We remark that this is required only for building the thermal models for the chip with heat sinks. Once the model is built, it can be used for different workloads with total different power maps.

This requires the processor to run a series of workloads without heat sink first, and then run the same workloads with heat sink on and with the same CPU core scheduling. Under this

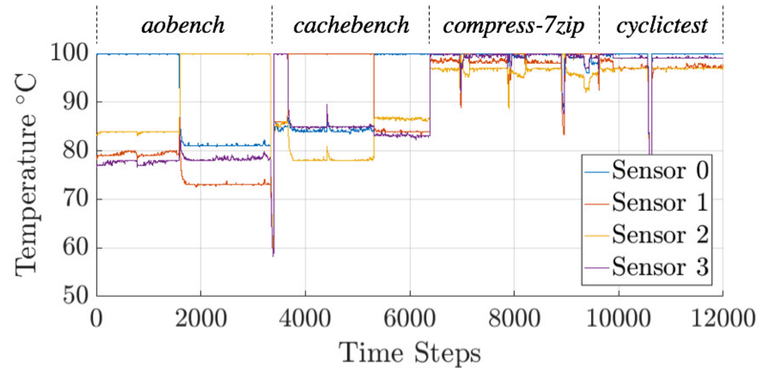


Figure 4.6: Thermal sensor values when processor uses the back-side liquid cooling with respect to workloads

condition, we can claim the power distributions will remain the same for both with and without heat sink, even though the thermal sensor data varies.

In our study, single-threaded and multi-threaded workloads from Phoronix Test Suite are used. For the setup with and without heat sink, single-threaded workloads such as *aobench* and *cachebench* are forced to run with the same core mapping by setting the workload's CPU affinity. Multi-threaded workloads such as *compress-7zip* and *cyclictest* do not need forced core mapping since all cores are utilized and same scheduling. Thermal sensor values of some time-segments of workloads are shown in Fig. 4.6 and Fig. 4.7 for back-side cooling and heat sink cooling, respectively. The relatively high (100°C) or low (80-90°C) temperature indicates the corresponding core is in busy or idle working status. As we can see, for *aobench* and *cachebench* on both cooling setups the sensor of busy core reaches 100°C, which is the thermal spec temperature, whereas the other three cores are about 10°C cooler. And for *compress-7zip* and *cyclictest*, all sensor temperatures are near the maximum values.

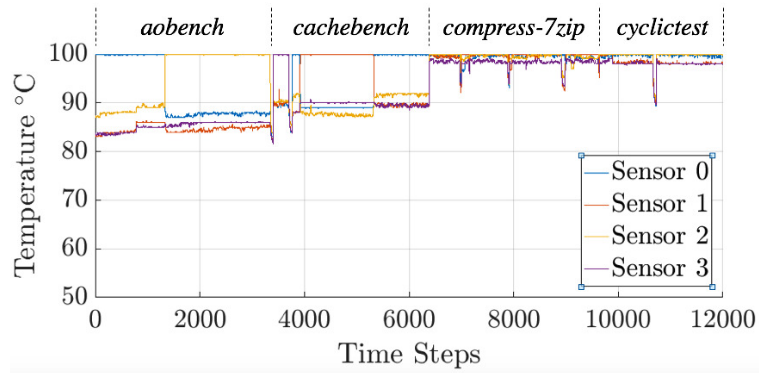


Figure 4.7: Thermal sensor values when processor is mounted with heat sink with respect to workloads

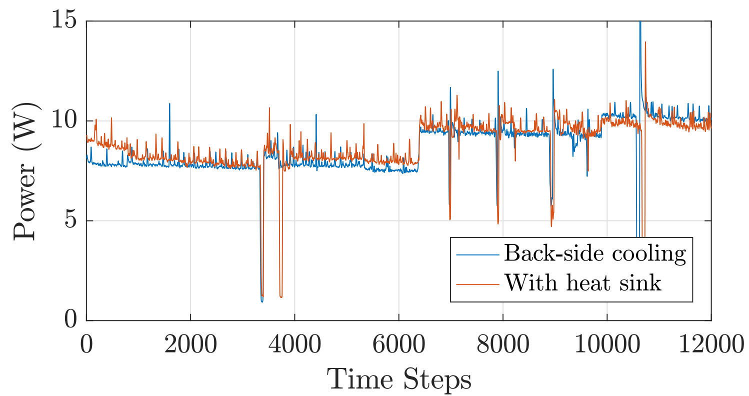


Figure 4.8: Total processor power with back-side liquid cooling and with heat sink during the time line

Furthermore, we ensure both of the cooling techniques - with back-side cooling or with top-side heat sink have similar or close cooling capability, so that the total power is the same for both cooling solutions and CPU can work under similar thermal conditions in both scenarios. As a result, the amount of leakage current would also be similar or close in both scenarios. Our study shows that the leakage distribution difference will not significantly affect the consistency of power maps in the two cooling scenarios under this condition. DVFS conditions are also kept in nominal status, meaning no unexpected frequency kick-down by over heat (actually we only require the DVFS are same in both cooling setups). The corresponding total power consumptions of both cooling scenarios during the time line are plotted in Fig. 4.8, where we can see two power traces follow almost identical trend, despite very slight variations.

4.4 Application for Different Workloads

The proposed thermal model with heat sink can be applied to different workloads once it has been built as the thermal model is workload-independent in theory. The only thing is that one has to obtain the accurate power maps for the workloads first. For different workloads, one way is to go through the same power map characterization process using thermal imaging system as we discussed in this paper and ensure that the back cooling has the same cooling capability as the heat sink cooking in the sense of total power. However, this is laborious as we need to ensure the running settings (task-to-core mapping, DVFS) are same. Estimating accurate power density map or map series for commercial multi-core processors for different workloads is a difficult problem. The proposed thermal map to power map based power map characterization can still be used as the tool to collect the data to train the machine-learning based model based on the real-time utilization

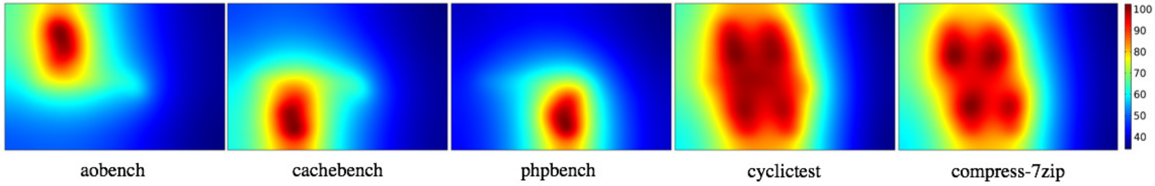


Figure 4.9: Full-chip thermal maps of processor under real working conditions with heat sink mounted on with respect to various workloads

metrics such as on chip Intel's Performance Counting Monitor [Int], DVFS setting and task-to-core assignment and scheduling. We believe this will bring huge online real-time thermal modeling and monitoring capability which is not available for today's commercial multi-core processors. Recent studies show that one can built very accurate data-driven deep neural network model to map from IPCM to full-chip thermal maps in real-time [SZZ+20a] which is assisted by back cooling and infrared imaging system. As a result, we can extend that technique for full-chip power map estimation so that there is no need to go through the same power map characterization process tediously. Instead, we can perform real-time thermal map estimation for commercial multi-core processors under heat sinks. But this can be our future work and is not the focus of this submission.

4.5 Thermal Map Results Under Real Working Conditions

In this subsection, we will firstly present the reconstructed thermal map results from our FEM model created with COMSOL Multiphysics for this commercial multi-core processor underneath the heat sink cooling. We have examined various steady states for both single-threaded and multi-threaded workloads to compare the computed temperature at sensor locations under the heat sink against the real sensor measurements. Full-chip processor heat maps for certain typical power scenarios of workloads are illustrated in Fig. 4.9. Temperatures extracted from those computed heat

Table 4.1: FEM computed temperature at sensor locations VS. real sensor values with heat sink

Workloads		**Sensor0 location**	**Sensor1 location**	**Sensor2 location**	**Sensor3 location**
idle	Real	43.8	44.3	44.9	42.2
	Computed	43.9	44.0	43.9	43.5
cachebench #1	Real	100.0	89.5	91.5	89.0
	Computed	100.6	89.3	90.2	88.5
cachebench #2	Real	89.0	100.0	89.0	90.3
	Computed	88.8	100.5	87.1	90.4
aobench	Real	88.2	86.0	100.0	86.0
	Computed	89.8	86.1	100.2	87.6
phpbench	Real	87.0	84.0	100.0	85.2
	Computed	90.7	86.7	103.6	88.4
cyclictest	Real	99.9	98.2	99.8	98.0
	Computed	101.6	101.5	100.8	101.6
compress-7zip	Real	99.9	99.5	98.3	100.0
	Computed	99.4	99.6	98.8	99.1

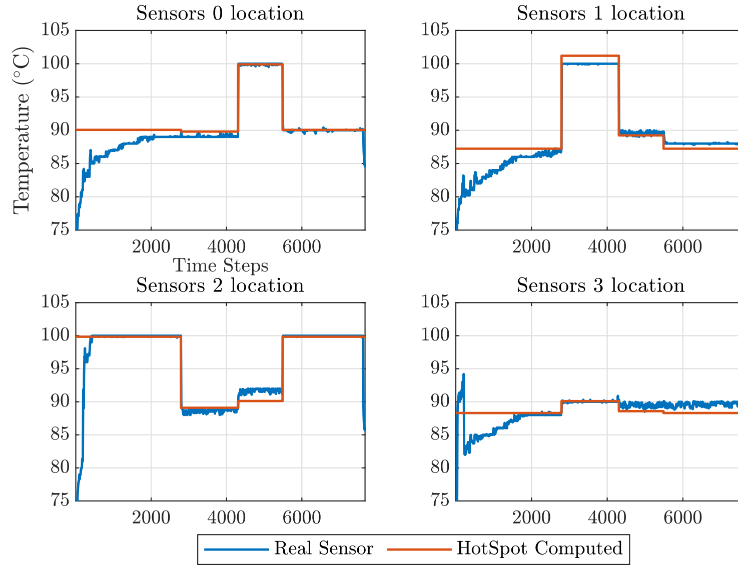
maps are compared against real sensor measurements and listed in Table 4.1. In the test, the estimation error varies with respect to the workloads. From the results listed in Table 4.1 we observed that *compress-7zip* has the best accuracy (0.5°C average error), whereas *phpbench* has the largest error which is shifted up by about (3.3°C) in such thermal steady state. In the time axis, the maximum absolute error for *phpbench* and *cyclictest* in the worst scenario is between 3∼4°C.

To analyze the estimated temperature in the time axis, we utilize HotSpot [HGV+06] to compute the temperature at sensor locations. Power maps can be obtained from the thermal

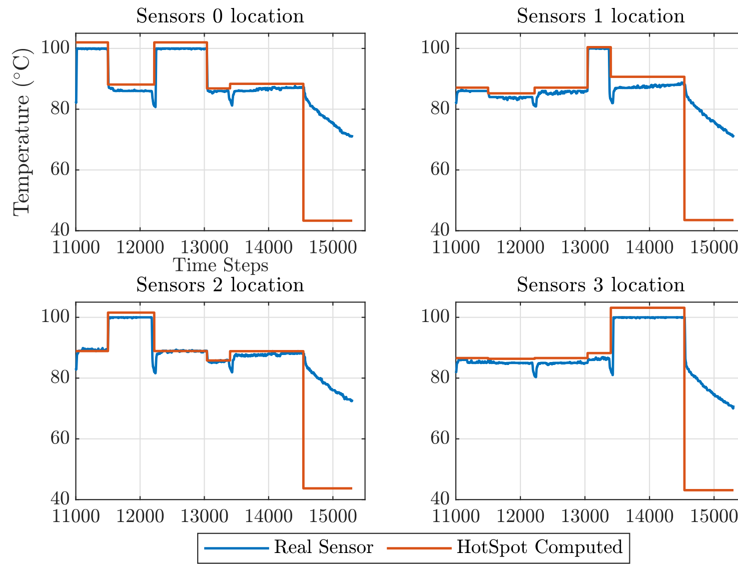
images in the time axis. We tried to perform the true transient analysis in HotSpot based on the power map stream we obtained. However, we found that the true transient analysis in HotSpot takes a prohibitive amount of time in our test cases and hence is not feasible. Reasons are the large map dimension and relatively long sampling interval. Specifically, HotSpot's typical sampling interval is 3.3μs, whereas in this work has 0.016 ms (60Hz). It takes about two minutes to compute one single time step even when shrinking the map area by 50 times. Changing the sampling time (3.3μs) in HotSpot to match the real sampling time (0.016 ms) does not help the simulation because of the underlying mechanism of HotSpot. However, the steady state computation by HotSpot is much faster. As a result, we perform the *pseudo-transient* analysis in which the temperature series of consecutive steady states in time are computed instead. So in this work, we only present and compare the pseudo-transient analysis results.

For each steady state, power maps corresponding to that steady state are averaged to one map sample and fed into HotSpot. For instance, in Fig. 4.10(a) the time steps from 2792 to 4308 is one steady state and time steps from 4358 to 5488 is another steady state, and one averaged power map for each steady state is fed to HotSpot. Hence, we can form the temperature series of consecutive steady states in the time axis. One detail is that we implement the 64 by 64 grid model in HotSpot to compute the thermal maps first and later interpolate them back to the original image dimension.

Fig. 4.10 shows the temperature computations at the sensor locations compared to the real sensor readings under heat sink with respect to *cachebench* and *phpbench* in a certain time duration. The *phpbench* is followed by *idle* from the time step 14540. The maximum absolute error for steady states only for *cachebench* is 1.98°C, and for *phpbench* is 3.7°C. The average absolute errors



(a) *cachebench*



(b) *phpbench*

Figure 4.10: Comparison between the HotSpot computed steady state temperature at sensor locations and the real sensor readings for (a) *cachebench* and (b) *phpbench* in a time duration

over the time axis across four sensor locations are 0.78°C and 2.5°C for these two benchmarks, respectively. Note that *phpbench* has the worst error among the workloads we tested. The results from HotSpot and COMSOL are quite similar.

We remark that the error is calculated based on the measured temperature at the sensor locations because this is the only measured information we can have. But we believe the estimated errors based on the sensor locations are good error indicators for the entire thermal map estimation. As we observe, temperatures track the real trend for all the workloads quite well. One explanation of the error may be due to the leakage difference or the κ error as we discussed in the previous section.

Finally, the average absolute estimation error using COMSOL turns out 2.2°C for the steady states across all sensor locations over all the workloads, in which only one averaged sample is counted for each steady state. In this way the error won't be biased by the length of idle status or other steady states. We also noticed that for some of the workloads the hottest spot is located away from the sensor locations and can be 3°C higher than the nearest sensor measurement even though only 2mm away. Such difference is heat-sink-related. Our measurements also show that when the device runs with back-side cooling and with the processor exposed to air, the underestimated hot spot temperature can be 6-7°C higher than the nearest sensor. The thermal spec power of the processor in our study is 15W, however, the difference between the hottest spot and sensor may reach a higher value as processor power goes higher. This observation raises the importance of complete heat maps rather than thermal sensor values of limited locations.

4.6 Summary

In this chapter, we present our accurate thermal model for real multi-core processors with heat sink and the validation method. We first show how to identify the locations of physical thermal sensors of commercial off-the-shelf processors, whose sensor locations are usually not available publicly.

Then in this work we creates a high-fidelity FEM thermal model with heat sink and reconstructs the full-chip thermal maps while the heat sink is on. Third, we ensuring that power maps are similar under back cooling and heat sink cooling settings. Lastly, thermal maps are reconstructed by the FEM thermal model using those power maps, and the reconstructed thermal maps are verified by the matching between the on-chip thermal sensor readings and the corresponding elements of thermal maps.

Under heat sink cooling, the average absolute error is 2.2°C over a 56°C temperature range and about 3.9% error between the computed and the real thermal maps at the sensor locations. This work is published in [ZSO+21].

Chapter 5

Thermal and Reliability Management Considering Hot Spots

5.1 Related Work

Khdr *et al.* [KES$^+$14] introduces a multi-objective DTM method that aims to efficiently avoid thermal threshold violation and at the same time keeps the temperature balanced between cores based on the core temperature. It Derives a regression-based distributed temperature prediction model and a centralized task allocation model, it stops tasks that potentially cause overheating or imbalance of the cores, and resumes the tasks once there are available cores. Das *et al.* [DSM$^+$14] develops a DTM technique that takes advantage of both the thermal profile within (intra) and across (inter) applications based on Q-learning, which learns the relationship between the task allocation, dynamic voltage/frequency scaling (DVFS) and device aging / mean-time-to-failure (MTTF). Lu *et al.* [LTB15] presents a task allocation method based on the core and router temperatures and pre-

dicts near-future temperature that assists the DTM. Their algorithm updates the prediction models after each allocation based on Q-learning. Q-learning-based control techniques are often subject to fast rising learning spaces as the states and actions of systems expand. Iranfar *et al.* [ISKAK15] proposed a machine learning or ML-based power/thermal management approach that uses a heuristic to limit the learning space by assigning a specific set of available actions to each existing state. A recent state-of-the-art DVFS technique enables scaling down of the management cycle to microsecond time scale and achieves fast per-core DVFS [ZGL+20], which significantly reduces the power consumption across cores. Recently [ZSG+20] proposes a deep reinforcement learning based method to allocate the tasks based on the hot spot power rather than temperature information, which infers the power information has great potential to be used to improve the system and thermal performance of the chip.

5.2 Reliability Models

In this section, we briefly review the three major VLSI reliability effects: the electromigration (EM) for interconnects, the negative biased temperature instability (NBTI) and hot carrier injection (HCI) for MOSFET devices and their calculation models. We note that the proposed method can consider other failure effects as well. The three failure effects are the dominant aging effects in the VLSI systems as EM will cause the power grid network to be time-varying and changes the voltage drop over time. NBTI and HCI can lead to the threshold voltage shift such may cause failure to signal transition and timing. In addition, calculations for the aging and lifetime due to EM and NBTI are implemented through an open source tool – LifeSim [RRC+18], which we will explain in detail in Section 5.5.

5.2.1 EM Model

The currently employed method of predicting the time to failure regarding the EM effect is based on a physics-based EM analysis method [HYST14, TTK$^+$19]. It comprehensively models the EM effect considering the void nucleation phase and growth phase during which the wire resistance starts to change. Specifically, the void nucleation time can be expressed as:

$$t_{nuc} \approx \tau^* e^{\frac{EV}{kT}} e^{\frac{f_v \Omega}{kT}} (\sigma_{res} + \frac{eZ\rho l}{4\Omega} j) In\{\frac{\frac{eZ\rho l}{4\Omega} j}{\sigma_{res} + \frac{eZ\rho l}{4\Omega} j - \sigma_{crit}}\} \quad (5.1)$$

with $\tau^* = \frac{l^2}{D_0} e^{\frac{E_D}{kT}} \frac{kT}{\Omega B}$. Here, E_V and E_D are the activation energy of vacancy formation and diffusion, f_v is the ratio of volumes occupied by vacancy and lattice atom, σ_{res} and σ_{crit} are the residual stress and critical stress. Ω is the atomic volume, l is the wire segment length, eZ is effective charge of the migrating atoms, j is current density, T is temperature, and ρ is the wire electrical resistivity.

At the system level, to model the current density, we follow the similar formula used in the RAMP [SABR04] and the work in [DK96], which can be related to the switching probability of the line, α, as

$$j = \frac{CV_{dd}}{WH} \times f \times \alpha \quad (5.2)$$

where C, W and H are the capacitance, width, and thickness, respectively of the line and f is the clock frequency.

Once the void is formed in the wire it starts to grow and the wire resistance increases over the time. The drift velocity of the void edge is expressed as:

$$\theta = \frac{D}{kT} eZ\rho j \quad (5.3)$$

Further, kinetics of the wire resistance change with respect to the growth time is approximated as [HYST14]:

$$\Delta r(t_{grow}) = \theta t_{grow} \left[\frac{\rho_{Ta}}{h_{Ta}(2H+W)} - \frac{\rho_{Cu}}{HW} \right] \tag{5.4}$$

where ρ_{Ta} and ρ_{Cu} are the resistivity of the barrier material and copper, W is the line width, H is the copper thickness and h_{Ta} is the barrier layer thickness. The growth time is calculated for a given resistance percentage change threshold (such as 10%). The final time-to-failure due to EM effects is determined by adding the nucleation time and the void growth time together.

5.2.2 NBTI Model

Negative biased temperature instability (NBTI) occurs when negative biased voltage is applied to the gate of a PMOS transistor, the presence of holes in the channel causes Si-H bonds to break at the interface between the gate oxide and the channel, causing positive traps in the interface, which increase V_{th} [KYM$^+$99]. The reaction rate mainly depends on the temperature T and the supply voltage V_{dd}. The model of lifetime reliability due to NBTI we use is based on the work by Srinivasan *et al.* [SABR05]. MTTF due to NBTI at a temperature T, is given by:

$$MTTF \propto [(ln(\frac{A}{1+2e^{\frac{B}{kt}}}) - ln(\frac{A}{1+2e^{\frac{B}{kt}}} - C)) \times \frac{T}{e^{\frac{-D}{kt}}}]^{\frac{1}{\beta}} \tag{5.5}$$

where A, B, C, D, and β are fitting parameters using the published NBTI failure data [ZLS$^+$04], and k is the Boltzmann constant. Based on the model in [SABR05], the values we use are $A = 1.6328$, $B = 0.07377$, $C = 0.01$, $D = -0.06852$, and $\beta = 0.3$.

5.2.3 HCI Model

Hot carrier injection (HCI) refer to the high energetic carriers, which is the result of high electric fields in the drain region of a transistor, are injected into the gate oxide. These carriers form interface states and eventually result in performance degradation (increase of V_{th}) in the transistor under stress [TYMH95]. The equation below evaluates the HCI-induced threshold voltage increase [OT12].

$$\Delta V_{th}(\alpha, T, V_{dd}, t) = A_{hci} \cdot u(V_{dd}) \cdot v(T) \cdot \sqrt{\alpha \cdot f \cdot t} \tag{5.6}$$

with

$$u(V_{dd}) = e^{(\frac{V_{dd}-V_{th}}{E_1})},\ v(T) = e^{(-\frac{E_a}{kT})} \tag{5.7}$$

where t stands for operation time, α is activity factor and f is core frequency. In addition, t_{ox} is the oxide thickness, and E_1 depends on the device specifications, temperature, and V_{dd}. Further, A_{hci} is a technology-dependent constant and activation energy E_a is considered a positive constant.

5.2.4 Summary of Reliability Models

In summary, EM causes the power grid network to be time-varying and changes the voltage drop over time. NBTI and HCI lead to the threshold voltage shift such that may cause failure to signal transition and timing. In this work, we set the failure criterion to be 10%, i.e. 10% wire resistance change due to EM and 10% change of threshold voltage due to NBTI and HCI are considered end of lifetime.

5.3 Observation and Motivation

The analysis, measurements and implementations of this work are all based on real systems. The reason is measuring from a real processor when it is executing workloads is more precise and has more realistic meaning than from computer simulators. Secondly, open-source computer simulators hardly include the ready-to-use architecture resources for the latest off-the-shelf processors.

5.3.1 Thermography System Setup

In order to acquire precise thermal and power information within the core, a proper measurement system for spatial temperature is critical. To this end, we have adopted the thermography measuring system proposed in [AH15]. This setup features a thermoelectric device mounted on the other side of the motherboard right beneath the processor allowing it to be cooled from underneath, as opposed to heat sinks drawing heat upwards. This setup leaves the front side of the processor fully exposed to the infrared camera without any interference layer in-between, as shown in Fig. 5.1.

An adjustable DC power supply is used to control the heat flow through the thermoelectric device so that the operating conditions can be matched to the baseline cooling unit (stock heat-sink) using the calibration method discussed in [AH15]. Unlike the traditional flowing-oil-based front-cooling methods [DNR13], no decoupling procedures are required in this setup. The thermal image capturing rate can reach as high as 60 frames per second.

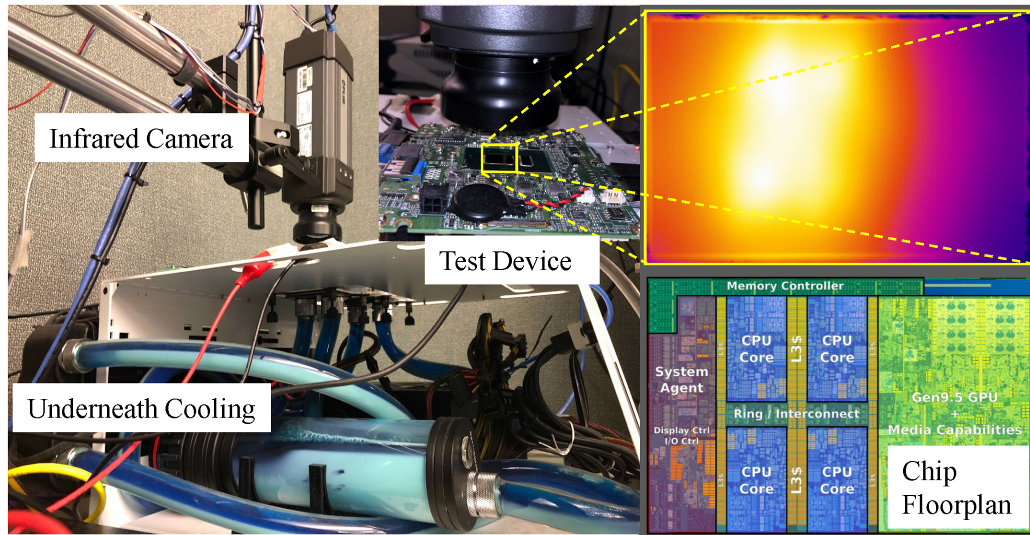


Figure 5.1: Infrared thermography system

5.3.2 A Glance of Hot Spots

As thermal sensor measurements alone cannot provide the information how cores are stressed spatially, there is a remarkable room for improvement compared with existing state-of-art DTM/DRM approaches. We first illustrate how the cores can be stressed in various ways that the sensors cannot tell. Then the idea of optimization over the existing techniques will be described at a high level in this section.

Fig. 5.2 shows the measured spatial temperature (Intel Core-i7 quad-core) when it is under a workload (Splash-2 benchmark *radiosity*). It reveals that the temperature between a true hot spot and the nearest sensor can be quite different. When there are many cores under workloads, temperature sensors are likely to measure the same or similar temperature even though cores are under different workloads being stressed in different patterns. We measured the temperatures in the time axis by the embedded sensors, shown as Fig. 5.3(a). It is obvious that temperatures across

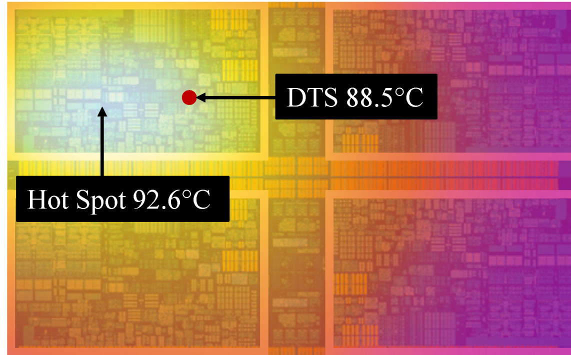
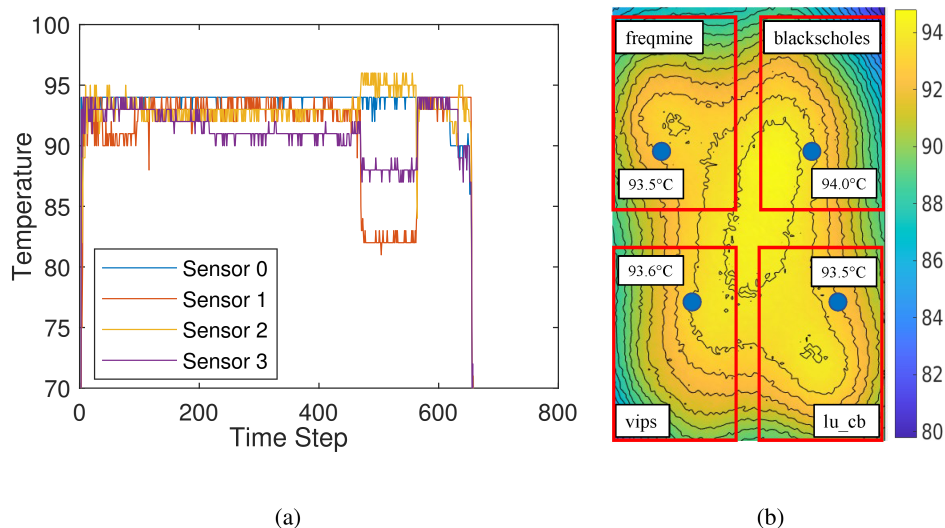


Figure 5.2: Measured temperature of a hot spot versus the nearest sensor reading

all sensors, at least two or three, are often very close during the runtime. Note that the precision of sensors is only integer. Moreover, as shown in Fig. 5.3(b), when four workloads (*lu_cb*, *vips*, *blackscholes* and *freqmine*) are running on the four cores respectively, the temperatures at sensor locations measured by the imaging system are 93.5, 93.6, 94.0 and 93.5°C, where the difference is quite small.

We remark that the thermal hot spots are always the power density hot spots or the heat-source hot spots. But this is not true the other way around as shown in a recent study [SZZ+20b]. Heat-source hot spots can be viewed as *potential thermal hot spots* or their *spatial distributions*, which can be activated by specific workload. The hot spots from heat sources or power sources can provide more useful, especially critical information about the true thermal hot spot distributions for real commercial multi-core processors, which is the motivation in this work to use power density hot spots. In the sequel, for the sake of simplicity, hot spot simply means power density hot spot and power or power pattern means the power density or power density pattern.



(a) (b)

Figure 5.3: (a) On-chip sensor readings (one sensor per core). (b) Measured temperature at the sensor locations (blue dots).

We calculate the core's power patterns of various PARSEC-3.0 and Splash-2 workloads and some typical patterns are shown in Fig. 5.4 as examples, respectively. There are three primary hot spot locations observed in the core. Some workloads have higher and sharper power peaks than others, while other workloads show more even power distribution. Consequently, utilization of the hardware resources, reflected by the power density at hot spots, indicates the different stresses of the silicon chip. Hence, there is a considerable potential for task migration operations to optimize the thermal and reliability performance by utilizing the hot spot power information.

For illustration, the typical power density measured at the hot spots with respect to workloads are listed in Table 5.1, where the three primary hot spots are named as *HS1*, *HS2* and *HS3* are listed. It should be noted that the applications may contain both serial and parallel threads, and the power density values listed in Table 5.1 are averaged values through the thermal-to-power calculation when the applications run into a thermal steady state, hence the parallel phase (also dominant

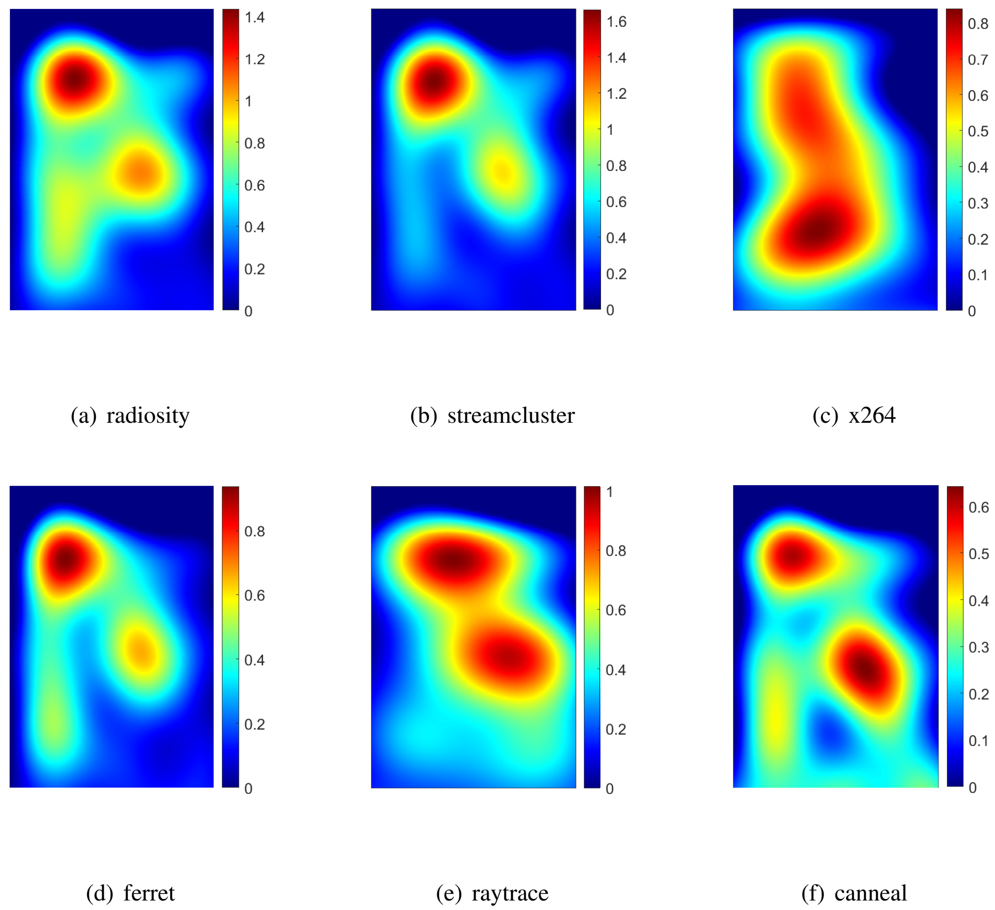


Figure 5.4: Power patterns of PARSEC-3.0 and Splash-2 benchmark workloads on a real Intel Core-i7 processor at the core scale (within the core).

Table 5.1: Average Power Density (W/mm^2) at Hot Spots for Various Workloads

Workload	HS1 Power	HS2 Power	HS3 Power
blackscholes	1.04	1.56	1.82
bodytrack	0.92	1.40	1.82
fluidanimate	0.75	1.3	1.8
streamcluster	0.52	1.06	1.57
dedup	0.75	1.0	1.56
facesim	0.75	1.0	1.56
swaptions	0.52	1.0	1.53
lu_cb	0.52	1.0	1.52
freqmine	0.52	0.91	1.38
radiosity	0.72	1.1	1.37
vips	0.26	1.0	1.3
radix	0.52	1.0	1.2
ferret	0.52	0.65	0.9
canneal	0.39	0.63	0.6
raytrace	0.56	1.08	1.15
x264	0.82	0.26	0.75
fft	0.5	0.9	1.3
ocean_cp	0.26	1.0	1.43
volrend	0.78	1.0	1.3

phase) of the application is considered in this table. The measuring workflow can be implemented on other chips as well.

5.4 Proposed Hotspot-Aware Task Allocation Framework

In this section, we will describe the overall workflow for the proposed task allocation algorithm. The framework consists of two major components – (1) a detector model detecting the power density of the primary hot spots and (2) a management controller that collects the power information of those hot spots of all the cores and controls the allocation of threads. For the sake of comparison, we will not interfere with the DVFS policy of the system.

5.4.1 Learning-Based Hot Spot Modeling and Detection

One important aspect of the proposed method is to know which hot spot locations are active or invoked by the workload in a core in real-time. This can be achieved by using deep neural networks. We estimate the power density at the hot spots of the off-the-shelf multi-core processors during real-time from the online utilization metrics. Specifically, we implement a deep neural network (DNN) as a supervised learning model which can estimate the power densities at hot spots in cores from the underlying real-time resource utilization information.

In our implementation, we take advantage of a multi-layer perception (MLP) network with two fully connected layers and a dropout layer for hot spot power density detection (Fig. 5.5). The input data for the network's training and inference is obtained from Intel's Performance Counter Monitor (IPCM) [Int], IPCM provides the system-level utilization metrics that we will be utilizing in this work. For non-Intel chips, the equivalent performance monitors can be used (i.e. AMD

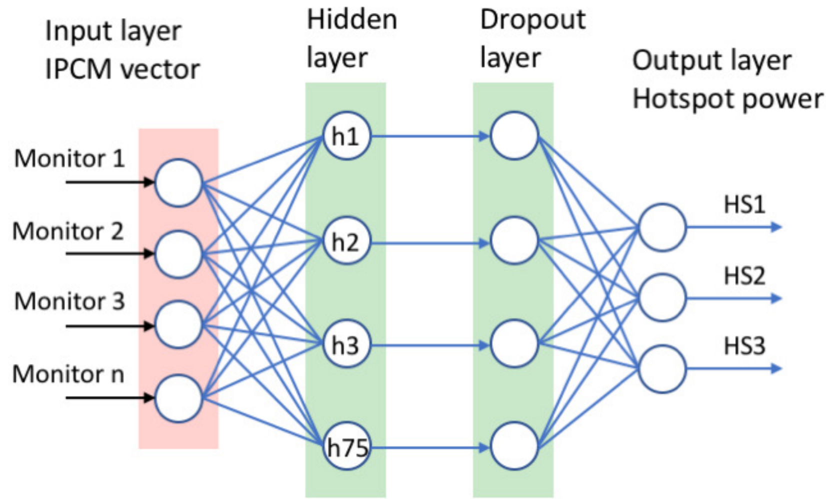


Figure 5.5: Power detector network architecture

uProf [AMD]). IPCM provides the real-time processor package and core-wise performance metrics such as frequency, energy, instruction per cycle, cache hit, read/write rate, etc., as well as the sensed temperature from the embedded sensors. The Intel chip used in this study, i.e. Core i7-8650U, has 4 cores and each core supports 2 threads with Intel's hyperthreading technology. Table 5.2 shows the complete list of IPCM performance metrics from both the package and core-wise (or thread-wise) domains that are used in this work. We note that the IPCM-based full-chip thermal map modeling method has been proposed recently [SZAT21]. There are 30 metrics corresponding to the whole package domain, and 16 metrics for each core thread. Considering that hyperthreads may happen on this chip, when measuring the training data we disabled the hyperthreading option, having one core only execute one thread at a time instead of two. In this way, we make sure the externally captured thermal images are matched for the thread executed in the core. Otherwise, the thermal images and the following calculated power densities would be a contribution of two separate threads

Table 5.2: High-level Performance Metrics (Intel PCM)

Package			Core	
Exec	Read	C1res%	Exec	c0res%
IPC	Write	C2res%	IPC	c1res%
Freq	INST	C3res%	Freq	C3res%
AFreq	ACYC	C6res%	Afreq	C6res%
L3Miss	Time	C7res%	L3Miss	C7res%
L2Miss	PhysIPC	C8res%	L2Miss	T_{sens}
L3Hit	PhysIPC%	C9res%	L3Hit	
L2Hit	INSTnom	C10res%	L2Hit	
L3MPI	INSTnom%	Energy(J)	L3MPI	
L2MPI	C0res%	T_{sens}	L2MPI	

running concurrently on the same core due to the hyperthreading function. Once the NN model is trained it can be used in a thread-wise manner as one core's power is a combination by two threads. In total, the input vector contains 46 IPCM metrics for the core-wise (or thread-wise) hot spot power density detection neural network. In our later experiments, we limit one core to execute only one thread in order to reach easier software implementation of the algorithm in the user space, which will not lose the validity of the algorithm.

Output data of the network are the power densities at the identified primary hot spots of the core in real-time. In our case, the output dimension is three due to three identified hot spots. Note that the name of the workload is not a factor in the power detector network. We obtain the

core's hot spot power densities by deploying a recently proposed thermal-to-power transformation approach [SZZ+20b]. The corresponding thermal imaging measurements are collected at the same time when the processor is under workload. IPCM tool is launched also at the same time when the processor is under workloads, data of the performance counter metrics is sampled at the same frequency and synchronized to the thermal image capturing. Then, spatial power patterns are calculated through thermal measurements and power densities at the primary hot spots are extracted [ZSJT20]. Finally, IPCM metric vectors serve as inputs and power densities of hot spots per core serve as targets for the learning-based network. We measured 7200 thermal images with the highest camera frequency (60 Hz) and the synchronized IPCM metric vectors corresponding to each workload application, where 20 applications from PARSEC-3.0 and SPLASH-2 are measured for the network training and test procedure. In our study, we observed that the power patterns of all the workloads are steady during almost their entire execution time except for a slight instant fluctuation at the beginning. Moreover, the same workload demonstrates the same power pattern across different cores when executing on multiple cores parallelly. This convention actually shrinks the complexity of model learning and makes the network easy to train and use. We will present the inference (detection) accuracy of the online power density detector in the results section (Section 5.5).

5.4.2 Task Management Controller

The task management controller collects power information of hot spots, maintains the core and task status, and allocate incoming or ongoing tasks. We define the following concepts for a clear description.

Task Queue: Incoming tasks/applications are put in a queue following the first-in-first-out (FIFO) order. It is assumed there is no priority order among them since the priority is not related to this study.

Parallelism Count: The number of parallelisms is usually determined by the user space. To generalize the new algorithm for tasks running with multiple parallel threads, each element in the task queue contains the name of task and the number of parallel threads it asks for. In our implementation, the task will be assigned with as many available cores as the user-determined parallelism count by setting the task's CPU affinity, where CPU affinity means a list of cores the task can run on. Note that we only set/update the task's CPU affinity in every management cycle instead of assigning the underlying specific threads to the specific cores. The order of threads is maintained by the task itself and the functionality is guaranteed.

Core Status: Cores have two status, either available or busy.

Waiting Parallelism Queue: For an incoming multi-threaded task that requests multiple cores for parallel execution, the number of available cores may be less than the number it requests for. Then all the available cores are assigned to the task and the excessive number of parallelisms requested is put in the waiting parallelism queue till other cores become available.

Management Cycle: Threads of tasks are migrated among cores every management cycle, δt, e.g. 1$\sim$5 seconds.

Sampling Interval: Every sampling interval, e.g. 100$\sim$1000 milliseconds, the management controller updates the core, task, and hot spot status that it maintains, and allocates the queued task to cores immediately once there are available cores detected.

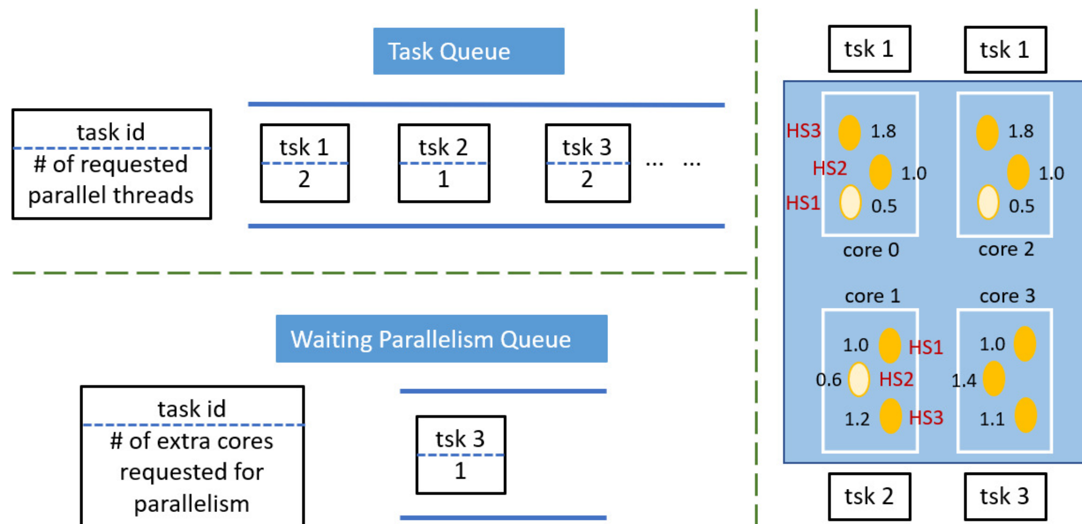


Figure 5.6: Example of task queue, waiting thread queue and the initial instant mapping.

Fig. 5.6 illustrates an example of the task queue, waiting thread queue, instant mapping and the corresponding power pattern on a quad-core processor. In this example, task 1 first occupies two cores, then task 2 occupies one core. Task 3 requesting for two cores is only mapped to one core given only one core left available. Task 3's another thread request is held in the waiting thread queue for the next available core. In our example, the processor layout follows a central symmetric pattern.

It should be noted that to reduce the complexity of interfering with the OS scheduler in this work, the management controller checks the status at each sampling interval from the user space rather than the kernel space of the OS. In the future, once the technique has been built into the OS kernel, the model does not need to check the status using the interval manner anymore, it should know those events immediately instead. We also comment that the power detector model does not need to calculate the hot spot power density all the time. As discussed, power pattern of the same task is quite steady on the time axis. Hence, the hot spot power information can be sampled, stored

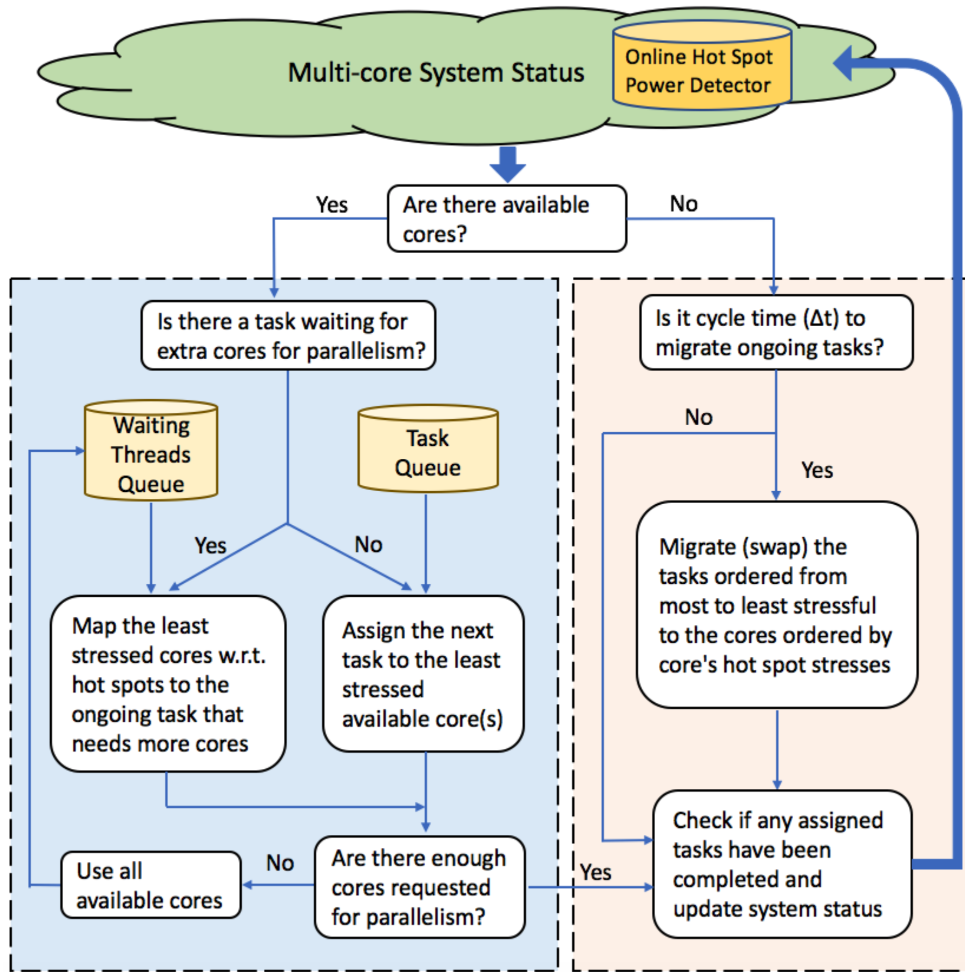


Figure 5.7: Task management workflow.

and reused. If an unknown task comes, the detector model will wake up for a short period of time intermittently and obtain an averaged hot spot power data for that task. In this way, the computation cost by the power detector model is much shrunk.

Fig. 5.7 presents the workflow of the management controller. At the top, the model accesses the multi-core system information it needs, including the core status, queue status and hot spot powers. In each information sampling cycle, it first checks if there are available cores. When

yes, it then checks the waiting thread queue to see if the ongoing task needs more cores. It always allocates the waiting thread before the next task unless the waiting threads queue is empty (i.e. FIFO). When available cores are not enough for an incoming task from the task queue, the task will be mapped to all the available cores and registered to the waiting thread queue for future available cores.

In every management cycle, the controller migrates the ongoing tasks from cores to cores according to the proposed mapping algorithm, which will be discussed in the next subsection. Afterward, the controller updates the system status it maintains.

5.4.3 Proposed Mapping Control Algorithm

As we already observe that the power (density) at the hot spots can vary considerably depending on the specific workloads. The higher power peaking at the hot spot, the more severe threat to the core's reliability. And the longer time the hot spot has been stressed, the lower reliability, too. Therefore, we develop a heuristic mapping algorithm that allocates tasks such that the average power peaking at the hot spots is mitigated. The mapping algorithm deals with two scenarios, one is migrating the ongoing tasks among cores, and the other is mapping the waiting threads or an incoming task to the available cores.

Migrate the Tasks

Suppose the processor has M cores where each core has H primary hot spots. And the current task map corresponding to the ongoing N ($N \leq M$) tasks is noted as $[tsk_{1,1}, .., tsk_{i,m}, .., tsk_{N,M}]$, where $tsk_{i,m}$ means the ith task running on the mth core and can be *None* if no task runs on that core. One task may run on multiple cores. Power at all H hot spots of a core activated by the task

tsk_i is noted as $P_{tsk,i} = [p_1, .., p_j, .., p_H]_i$, where p_j means the power at the jth hot spot activated by tsk_i.

Algorithm 1 Task Migration in A Management Cycle

Input: M cores, N ongoing tasks, H hot spots per core, current task map and performance counter metrics (IPCM)
Output: New task map $newMap$

1: $curMap \leftarrow [tsk_{1,1}, .., \ tsk_{i,m}, .., \ tsk_{N,M}]$

2: **for** i = 1 to N **do**
3: $\quad P_{tsk,i} \leftarrow [p_1, .., \ p_j, .., \ p_H]_i = Net(IPCM(tsk_{i,m}))$
4: $\quad P^{max}_{tsk,i} \leftarrow max(P_{tsk,i})$
5: $\quad h^{wst}_{tsk,i} \leftarrow argmax(P_{tsk,i}), 1 \leq h^{wst}_{tsk,i} \leq H$
6: **end for**
7: $maxPwrs \leftarrow [P^{max}_{tsk,1}, .., \ P^{max}_{tsk,i}, .., \ P^{max}_{tsk,N}]$
8: $wstHSs \leftarrow [h^{wst}_{tsk,1}, .., \ h^{wst}_{tsk,i}, .., \ h^{wst}_{tsk,N}]$
9: $sortedTsk \leftarrow \text{argsort}_{tsk}(maxPwrs, \text{reverse} = true)$
10: Sort $wstHSs$ by the same order to match the tasks in $sortedTsk$

11: Initialize $Cores \leftarrow set\{1, 2, .., M\}$
12: Initialize $newMap \leftarrow [None_1, .., \ None_m, .., \ None_M]^*$
13: **for** i = 1 to N **do**
14: $\quad tsk_i \leftarrow sortedTsk[i]$
15: $\quad h \leftarrow wstHSs[i]$
16: $\quad P_{c,h} \leftarrow [p_{(c,1)}[h], .., \ p_{(c,m)}[h], .., \ p_{(c,M)}[h]]$
17: $\quad prefCoreLst \leftarrow \text{argsort}_c(P_{c,h})$
18: $\quad$ Initialize $mappedCores_{tsk,i} \leftarrow set\{\ \}$
19: $\quad$ **for** $core$ in $prefCoreLst$ **do**
20: $\quad\quad$ **if** $core$ in $Cores$ **then**
21: $\quad\quad\quad$ add $core$ to $mappedCores_{tsk,i}$ for tsk_i
22: $\quad\quad\quad$ remove $core$ from $Cores$
23: $\quad\quad$ **end if**
24: $\quad$ **end for**
25: $\quad$ Update $newMap \leftarrow mappedCores_{tsk,i}$
26: **end for**
27: Use $newMap$ for task migration operation

The proposed task migration algorithm is elaborated in Algorithm 1. We firstly estimate the power at hot spots activated by every running task (line 2-3) through the machine learning-based power detector. And find the maximum power $P^{max}_{tsk,i}$ of hot spots (line 4) and the worst hot

Table 5.3: Exemplary Ordering of Tasks and Cores and Migration

Task Order	Worst Hot Spot	Preferred Cores	Mapped Cores
1) Tsk 1	HS3: 1.8 $\mathrm{W/mm^2}$	1, 3, 0, 2	1, 3
2) Tsk 3	HS2: 1.4 $\mathrm{W/mm^2}$	1, 0, 2, 3	0
3) Tsk 2	HS3: 1.2 $\mathrm{W/mm^2}$	1, 3, 0, 2	2

spot $h^{wst}_{tsk,i}$ (line 5) activated by that task. Then we sort the tasks by how stressful they are by the maximum power of hot spots they activate (line 7-9). The task having a higher maximum power of hot spots is considered more stressful. If two tasks stimulate the same maximum power (not necessarily on the same hot spot), then compare their second highest hot spot power, and so on so forth. For example, according to the data shown in Table 5.1, *blackscholes* should be ordered ahead of *bodytrack*, then *fluidanimate*.

Then, similarly, for each task the cores are ordered from the most preferred to least preferred ($prefCoreLst$) with respect to that task (line 13-17). Here, h indicates the worst hot spot that will be stressed by this task most and $p_{(c,m)}[h]$ denotes the accumulated power (energy) at the hot spot h of the core m. Line 19-24 map the task based on the order of its preferred cores. The task which is more stressful is taken care of earlier as having higher priority to pick the preferred cores than the less stressful tasks. If some preferred cores are already scheduled for other tasks in this management cycle, then these cores will be skipped for this task.

Once the new task map has been obtained, the task management controller migrates the tasks for this cycle. Following the example in Fig. 5.6, Table 5.3 shows the order of the tasks, the

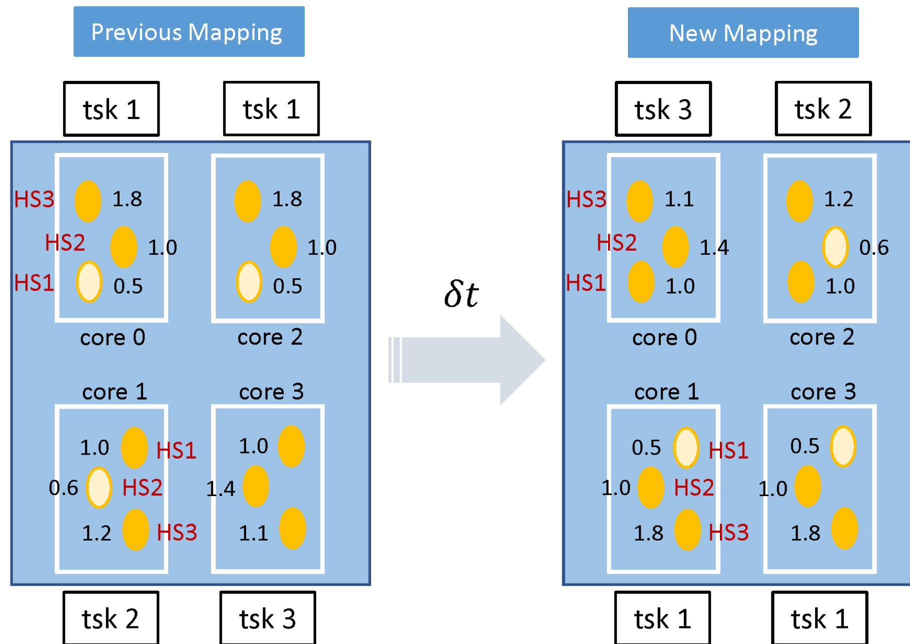


Figure 5.8: Task migration in a management cycle.

order of their preferred cores and the newly mapped cores, respectively. Fig. 5.8 further illustrates the resulted mapping diagram for the management cycle.

Map the Waiting Threads

This is similar to migrating the tasks. Locate the hot spot the targeted task will stress most and order the cores by accumulated hot spot power at that location. For example, if we are to allocate threads of *canneal*, then the cores should be ordered by their HS2 power in the management cycle because HS2 is the worst hot spot stimulated by *canneal*. Following the example shown in Fig. 5.6, Fig. 5.9 shows mapping a waiting thread of task 3 to the best available core, core 1, when core 0,1 and 2 become available simultaneously.

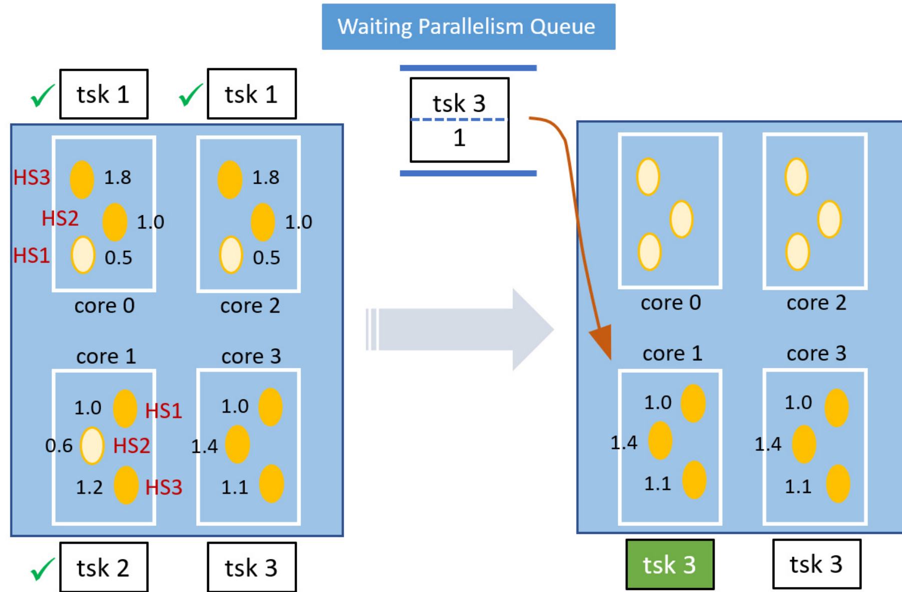


Figure 5.9: Example of mapping a waiting thread to the available cores.

5.5 Experimental Results and Discussions

In this section, we present the results for the proposed hotspot-aware task control method, Hot-Trim, for thermal and reliability management of multi-core processors. We implement and validate our method on a commercial Intel i7-8650U processor that features 4 CPU cores with PARSEC-3.0 and SPLASH-2 benchmark workloads [Bie11, WOT+95] (we write the benchmark workloads as tasks to be brief in this paper).

First, we present the performance of the power density detector neural network. We measured 7200 thermal images with the highest camera frequency (60 Hz) and the synchronized IPCM metric vectors corresponding to each workload application, where 20 applications from PARSEC-3.0 and SPLASH-2 are measured for the network training (80% data) and test procedure (20% data). 20% of training data is used for validation during the training procedure. Fig. 5.10 shows

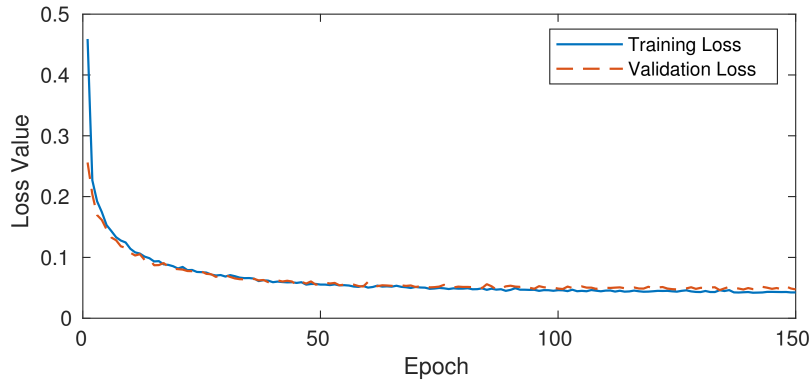


Figure 5.10: Power density detector neural network learning curves.

the training loss and validation loss during the training procedure of 150 epochs. We mark that the specific configuration of the MLP network (# of nodes, # of layers, etc.) is not an exact science. In this work, we used one hidden layer with 75 nodes and a dropout layer with a 0.5 ratio between the input and output layers, and the learning rate is 0.0005. We did not observe overfitting on the trained network model. Fig. 5.11 illustrates the comparison between the estimated power density and the measured power density traces at the identified hot spots, where the estimated power density is obtained from the learning-based power density detector neural network and the measured power density is obtained through the thermal-to-power method [SZZ+20b, ZSJT20]. As we can see, the estimated power traces align quite well with the real measured power traces. It should be noted that in the training procedure, thermal and IPCM data is obtained with the highest camera frequency. Whereas in the following management experiments, the cycle period is chosen as 2 seconds and IPCM sampling interval is 200 milliseconds. In our experiments, we observe that the power pattern of the running task usually takes only 200∼300 milliseconds to become steady after launching, as examples shown in Fig. 5.11. The migration and sampling frequencies are relatively low but sufficient. In this work, we obtain the power density estimation from the IPCM once at the end of

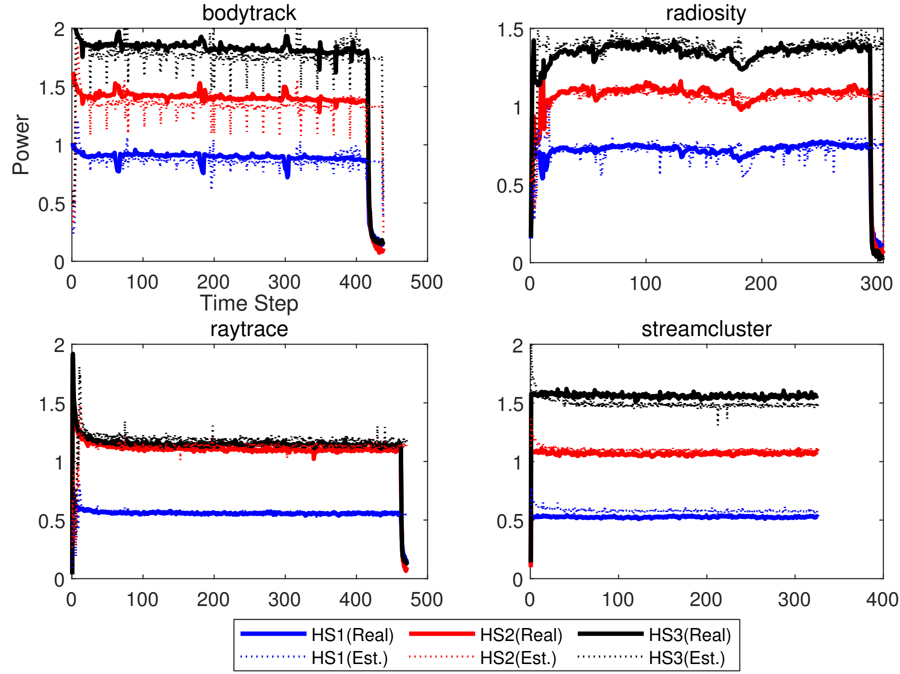


Figure 5.11: Power density ($\mathrm{W/mm^2}$) V.S. time steps (60 Hz) for workloads. Estimated power density compared to the measured power density at the identified hot spots.

every management cycle, i.e. before the next migration, and average the power density estimations after every management cycle of the task and average between cores if running on multiple cores. The computation overhead is reasonably low such that the online inference time is less than 100 μs and the overall computational time regarding the whole Algorithm 1 in one management cycle is between 300∼400 μs. We will present more details in the next subsection.

Second, we compare the performance of the proposed Hot-Trim with existing mapping methods. In this work, we compare three methods, i.e. Linux baseline mapping, temperature-based mapping and the proposed Hot-Trim mapping in terms of runtime performance, thermal behavior and the three critical reliabilities as mentioned earlier. Specifically, the Linux baseline mapping means when allocating the tasks, tasks will be launched without assigning the CPU core affinities.

We let the OS scheduler choose the CPU cores automatically to execute the tasks. For temperature-based mapping, we implement the most popular greed-based mapping policy such that the task is always mapped or migrated to the coolest core based on the thermal measurements of on-chip sensors. If there are multiple tasks executing on multiple cores, the tasks executing on the hot cores will be migrated to the cooler cores. Each mapping method will be deployed to execute the same series of tasks. In the meantime, performance counter metrics and thermal images of the full chip will be captured to investigate the runtime performance, thermal behavior and reliabilities. To make sure the comparison is comprehensive, we have gone through a few different experiment scenarios.

5.5.1 Comparison in System Performance

First, we start by investigating whether the proposed method degrades the runtime performance and how it compares to the Linux baseline, in other words, whether the total execution time is prolonged. If it degrades the original performance seriously then there would be no sense to propose more. The Linux kernel version on the test processor is 5.0.9-301.fc30.x86, and the OS distribution is Fedora 30. Note that in this work we only deploy the task mapping policy but not the DVFS scheduling, instead, we let the OS handle the DVFS as it normally does. Firstly, we compose diverse task series that contain various numbers and types of tasks. Each element in the task series is presented as (*task name*, *# of threads needed*). We also deploy two different input dataset size, *Large* and *Native* for the tasks in the PARSEC-3.0 and SPLASH-2 benchmarks. The user time of tasks with *Large* input size usually lasts for about a few seconds to half a minute, and with *Native* input size lasts for a few minutes.

In our implementation, we deploy python scripts for the high-level control algorithm and machine learning-based power detector and use batch scripts (bash shell) for direct task mapping

Table 5.4: Test Cases of Task Series

Test Cases	Task Series	Input Size
Case 1	[(ferret, 4), (streamcluster, 2), (canneal, 2), (raytrace, 2), (bodytrack, 1), (lu_cb, 2), (radix,2), (dedup, 2), (fft, 2), (vips, 1), (facesim, 2), (freqmine,1), (fluidanimate, 2), (bodytrack, 2), (ferret, 4)]	Large
Case 2	[(freqmine, 2), (blackscholes, 1), (dedup, 1), (canneal, 2), (radix, 2)]	Native
Case 3	[(vips, 2), (blackscholes, 1), (dedup, 1), (radix, 1)]	Native

and migration operations. The tasks and number of parallel threads are randomly chosen and the series of tasks in our test cases are listed in Table 5.4. The management cycle period is chosen as 2 seconds while the processor status and IPCM sampling interval are 200 milliseconds. Since we inspect the total execution time of a series of tasks, there will be no idle time for any core. This means once a task is complete on a core(s), this core(s) will be assigned with the next task immediately unless all the tasks in the queue are finished.

In order to make a fair comparison, each run must be launched under the same initial thermal condition. The chip is totally cooled to the initial temperature (about 30°C) before the next

run. And test cases are run many times to minimize the effects of random factors, such as ambient airflow or on-chip data caching. Please note that the cooling efficiency is forced constant all the time during the experiment. Back-side liquid circulation is at a constant flow rate, besides, the thermal-electric device which transfers heat from the motherboard downwards to the liquid circulation is kept at constant power at 62 Watts.

As shown in Table 5.5, the proposed technique will not degrade the system performance. Actually, the average execution time of the whole series of tasks is slightly decreased by 1.4$\sim$4.1%. It is interesting that one or two of the slow runs under Linux are considerably longer than the average time, which we are not sure about the reason. However, the execution time by Hot-Trim is quite stable. As mentioned in Section 5.4.2, incoming tasks are launched following a first-in-first-out (FIFO) order assumed by the series (task queue). When conducting the experiment under Linux default mapping, the task execution order is still determined by the FIFO. Essentially, we use a python script to launch the task one after one once there are available cores or previous tasks are done. Task is launched without setting its CPU affinity, hence the core assignment is decided by Linux. In this way, we could maximize the identity of other factors but only leave the mapping decisions to be different when comparing with temperature-based and the proposed algorithm. It is also more realistic that different tasks randomly come in the time axis than launching them all together. Hence, the task order or thread order is not within the scale of this study. The execution time (min, max, and average) listed in Table 5.5 pertains to the variation of a single run of the series of tasks.

On the other hand, the Hot-Trim task mapping algorithm reduces the average core temperature by about $1.21 \sim 1.31$°C degrees and the temperature is constantly lower for all test cases

Table 5.5: Linux V.S. Hot-Trim: Performance and Temperature

Test Case	Total Execution Time (seconds)							Avg. Core Temperature (°C)		
	Linux			Hot-Trim			Avg dt (%)	Linux	Hot-Trim	Reduction (dT)
	Min	Max	Avg	Min	Max	Avg				
Case 1	58.16	62.18	61.03	56.15	61.17	59.84	-2.0%	85.98	84.67	-1.31
Case 2	321.6	358.7	341.1	318.6	333.7	327.2	-4.1%	92.10	90.89	-1.21
Case 3	163.3	165.3	164.0	158.3	163.4	161.7	-1.4%	86.39	85.13	-1.26

compared to Linux baseline as shown in Table 5.5. It should be noted that the temperature reductions are all measured from the thermography system. Temperature at the truly identified hot spots reflects the same trend between the two mapping ways. We measured that the maximum temperature at the identified hot spots for different mapping methods is very similar (at around 95°C). However, the high-temperature duration and temperature spatial distribution vary. The average temperature at the worst hot spot HS3 of each individual core is around 1.5∼2 degrees higher than its average core temperature.

We note that the new task mapping method has no obvious effect on suppressing peak temperature in this study. The main reason is that the maximum temperature at the identified hot spots is determined by some heavy tasks such as *blackscholes*, *bodytrack* and *fluidanimate* regardless of which core they are assigned to as all the cores are homogeneous. As a result, as long as they are executed in the experiments, we will observe the similar maximum temperature regardless of the mapping method used.

5.5.2 Thermal and Reliability Improvement

This subsection compares the thermal behavior and VLSI reliabilities regarding EM, NBTI, and HCI among the three mapping methods. In this experiment, under each mapping method, a series of randomly chosen tasks are released one after one with random intervals between releasing two tasks in the timeline to mimic task allocations in real processors. To make a fair comparison and minimize the effects of random factors, the tasks are chosen in a pseudo-random way as well as the release intervals. In this way, all three mapping techniques will deal with identical workloads and identical arrival times of the workloads. To be simple and without losing the generality, we release 30 randomly chosen tasks one after one intermittently. The time interval (Δt) between releasing two consecutive tasks satisfy a uniform distribution $\Delta t \sim U(4, 12)$ seconds. The minimum and maximum interval are 4 and 12 seconds, respectively. This testing scenario is called Case 4. To further minimize the effects of random factors, the same series of tasks are executed under every mapping technique many times and each run starts under the same initial thermal condition.

Fig. 5.12 compares the temperature at the identified hot spot location HS3 with respect to each core under the three mapping techniques when releasing tasks with interval distribution $\Delta t \sim U(4, 12)$. In our case, HS3 is the most stressed one of the three identified primary hot spots. LB, TB, and HT are briefed for Linux baseline, temperature-based and Hot-Trim mapping, respectively. It can be observed from the plots that most of the time the temperature trend under HT is more similar to TB compared with LB. Mean of the temperature curves shown in Fig. 5.12 at the HS3 for each core are compared in Table 5.6. Thermal performance under Hot-Trim is obviously better than Linux baseline across all cores and is 1.15°C lower on average in this test

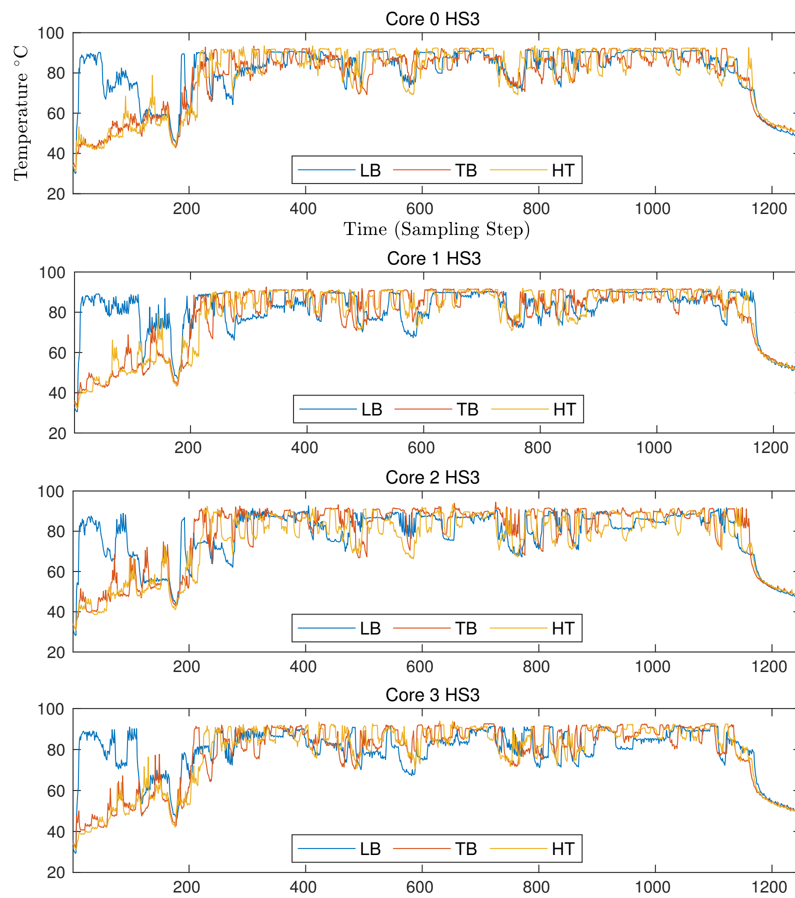


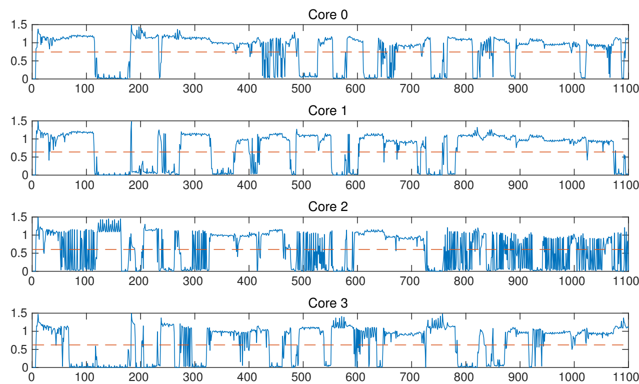
Figure 5.12: Temperature at the identified true hot spot location HS3 for each core under three mapping techniques (Case 4).

Table 5.6: Mean of Temperature over Time at the Worst Stressed Hot Spot Location HS3 of Each Core

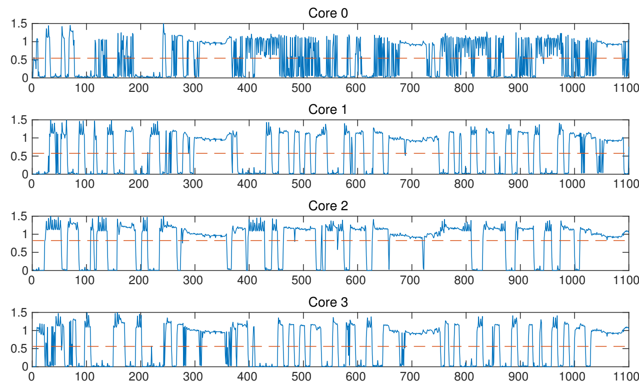
	Alg.	Core 0	Core 1	Core 2	Core 3	Max	Avg
Case 4	**LB**	80.81	80.79	77.43	79.18	80.81	79.55
	TB	78.77	79.32	79.11	78.58	79.32	78.94
	HT	79.18	79.43	76.37	78.62	79.43	78.40
Case 5	**LB**	70.18	65.08	69.64	65.96	70.18	67.72
	TB	66.98	67.52	66.92	67.24	67.52	67.16
	HT	66.76	67.93	64.81	66.71	67.93	66.55

case. Under temperature-based mapping, temperature is quite balanced across all cores, which is expected. Though, its average temperature is still higher than Hot-Trim.

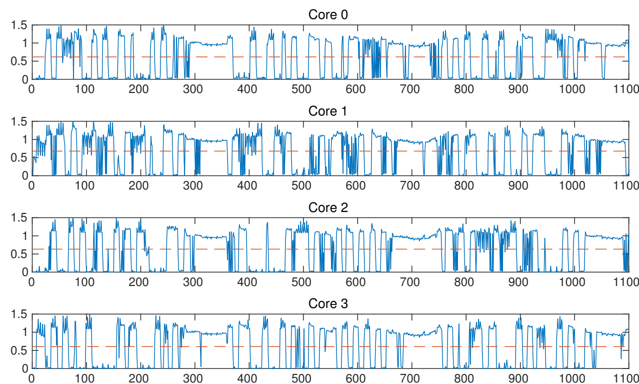
Core frequencies under the three mapping policies are shown in Fig. 5.13. In the experiment case, core frequencies show similar amplitude where the cores operate at around 1$\sim$1.2 GHz while under load and gate to near-zero frequency while they are idle. Frequency throttling seems not to show observable differences among the three mapping methods while the cores are under load. We remark that the system DVFS governor remains untouched during the experiments, hence changes in the frequency pattern for all cores are naturally governed by the system DVFS gover-



(a) LB



(b) TB



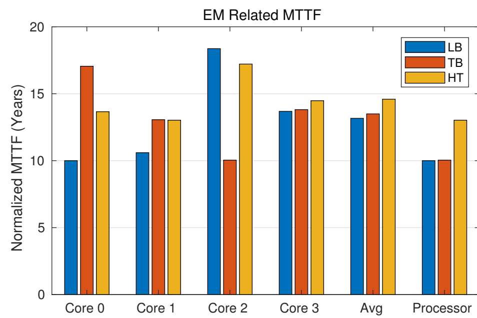
(c) HT

Figure 5.13: Core frequencies (GHz) V.S. time steps when processor under different mapping policies: (a) Linux baseline (b) Temperature-based (c) Hot-Trim. Red lines are the mean lines of each frequency series (Case 4).

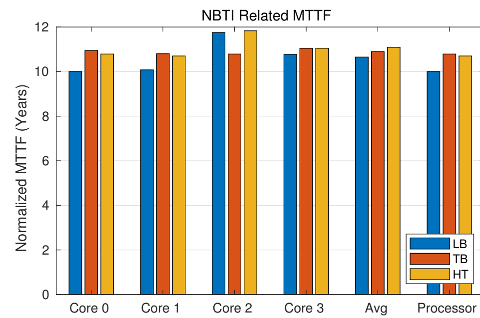
nor, and different mapping methods are treated constantly. It can be observed that the similar core utilization does not necessarily give similar lifetime reliability. A good example is that the averaged frequency of core 1, 2, and 3 under Linux baseline mapping are very close (about 0.62 GHz, Fig. 5.13(a)), however, their lifetime reliabilities vary much more, which we will describe in detail later. This is reasonable because cores can have hot spots stressed differently by executing different tasks.

Although temperature does not distinguish much between temperature-based and Hot-Trim mapping, VLSI reliability performances distinguish quite significantly. When implementing analytical models to calculate reliability effects and MTTF, we take advantage of an existing tool called LifeSim [RRC$^+$18]. LifeSim is a lifetime reliability simulator that offers a module named Reliability Management Unit (RMU). It calculates MTTF by EM and NBTI effects for many-core systems. For the sake of convenience, we take advantage of the RMU module by feeding our real experiment data, such as core frequency and hot spot temperature to characterize the reliability performance. In the meantime we create another script based on equation (5.6,5.7) when calculating MTTF due to HCI, where we treat A_{hci}, $u(V_{dd})$, α and E_a as simple constants.

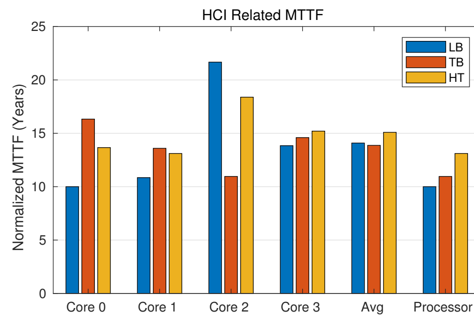
As a result, EM-related MTTF is illustrated as Fig. 5.14(a). Blue, red and yellow bars stand for Linux baseline (*LB*), temperature-based (*TB*) and the proposed Hot-Trim mapping (*HT*), respectively. The y-axis is normalized such that the shortest MTTF among all cores under Linux baseline mapping is 10 years. It can be seen that the EM-related MTTF under temperature-based mapping is significantly shorter than Hot-Trim in terms of both average and processor overall. The right-most bar labeled as *Processor* indicates that the MTTF of the entire processor is determined by the minimum MTTF among all the cores. Core 0 is the most stressed under temperature-based



(a)



(b)



(c)

Figure 5.14: Normalized MTTF considering (a) EM, (b) NBTI and (c) HCI reliability effects regarding test case 4 where task release intervals satisfy $\Delta t \sim U(4, 12)$.

mapping whereas core 2 is the most stressed under Linux baseline and Hot-trim. Hot-Trim stresses the cores much more evenly and leads to the longest average and overall MTTF. In detail, Hot-Trim is 30.2% longer than Linux baseline and 29.6% longer than temperature-based mapping in terms of processor overall lifetime, which are very significant.

MTTF due to NBTI is shown in Fig. 5.14(b). NBTI behavior is quite close between temperature-based and Hot-Trim since the temperature is close between the two mappings, as we know that NBTI is primarily dependent on temperature. The overall MTTF under Hot-Trim is only less than 1% shorter than temperature-based mapping, and 7.0% longer than the Linux baseline. HCI-related MTTF has a similar pattern to the EM-related MTTF, as shown in Fig. 5.14(c). Specifically, Hot-Trim is 31.1% longer than the Linux baseline and 19.6% longer than the temperature-based technique in terms of overall lifetime.

As for the migration energy overhead, thanks to Intel's Performance Monitor, we measured the entire real processor energy consumption executing the series of tasks as 2106.8, 2129.9 and 2118.5 Joules under Linux baseline, Temperature-based and Hot-Trim mapping, respectively. Therefore, the energy variation caused by the algorithm and task migration operations is merely marginal. In our method, the management cycle in the experiment is chosen as 2 seconds and CPU performance sampling interval is 200 milliseconds. We also implemented 1 second per management cycle and 100 milliseconds per sampling interval and found no measurable difference in the results. The resulting algorithm works quite well for running all the benchmarks. In general, one should reasonably choose the length of management cycle and sampling interval depending on the user cases when applying the proposed algorithm. Moreover, due to the estimation error of hot spot power densities and the granularity of management, the lifetimes (Fig. 5.14 and 5.15) indeed show

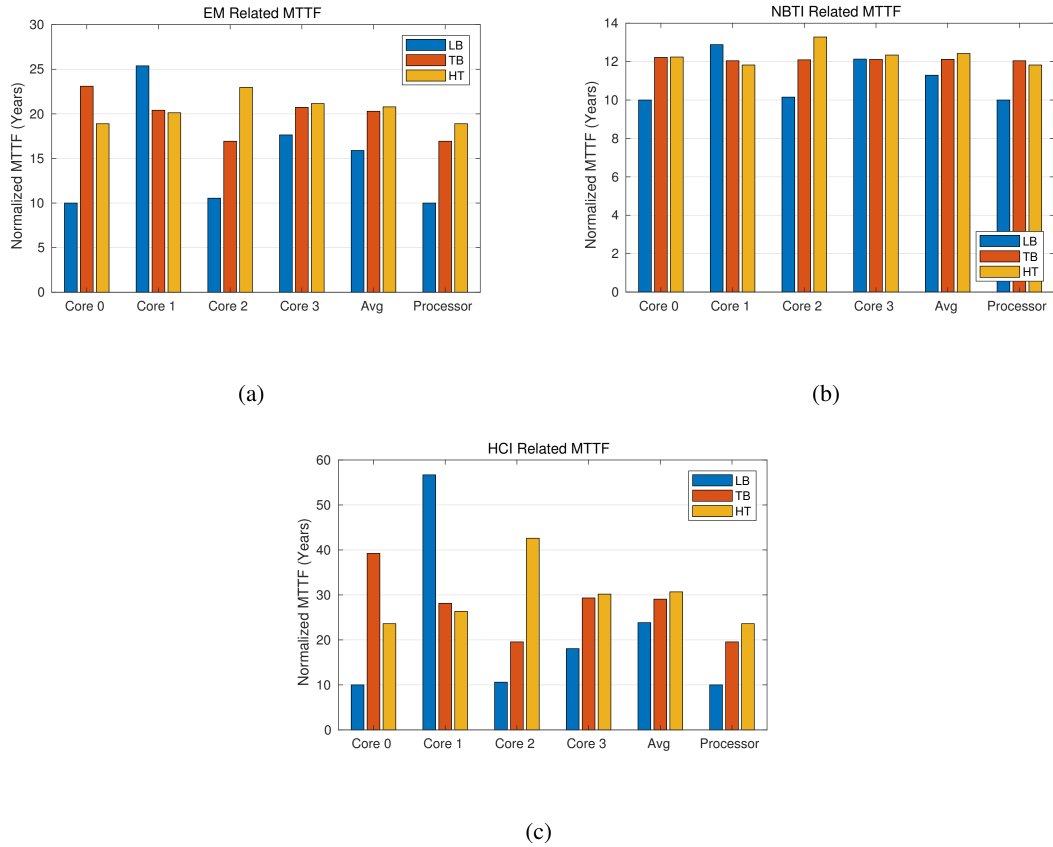


Figure 5.15: Normalized MTTF considering (a) EM, (b) NBTI and (c) HCI reliability effects regarding test case 5 where task release intervals satisfy $\Delta t \sim U(10, 20)$.

some imbalances between different cores. Ideally, the VLSI lifetime reliabilities should be perfectly balanced if the hot spot power densities were perfectly estimated.

We present more results through test case 5 to deliver an insight on the change of task release interval (Δt). The only difference in the settings of test case 5 compared to test case 4 is that the distribution of pseudo-random task release intervals is changed to $\Delta t \sim U(10, 20)$ seconds. This means fewer cores will be busy at the same time as the arrival of tasks are slower. The mean of temperature traces of the dominant hot spot HS3 for the cores are shown in Table 5.6. Further, the reliability performances are presented in Fig. 5.15(a), 5.15(b) and 5.15(c).

Moreover, proactive dynamic thermal management (PDTM) can be our future research work. The temperature at the identified hot spots can be proactively predicted because the real-time power information at those hot spots can be accurately estimated from Intel's Performance Counter Monitors (Section 5.4.1). With the predicted temperature in advance, we can apply a more comprehensive thermal and reliability management model [FLWZ17, CL20].

5.6 Summary

This work proposes a new dynamic thermal and reliability management framework via task mapping and migration to improve thermal performance and reliability of commercial multi-core processors considering workload-dependent thermal hot spot stress. The new method is motivated by the observation that different workloads activate different spatial power and thermal hot spots within each core of processors. Existing run-time thermal management, which is based on on-chip location-fixed thermal sensor information, can lead to suboptimal management solutions as the temperatures provided by those sensors may not be the true hot spots.

The new method, called *Hot-Trim*, utilizes a machine learning-based approach to characterize the power density hot spots across each core, then a new task mapping/migration scheme is developed based on the hot spot stresses. Compared to existing works, the new approach is the first to optimize VLSI reliabilities by exploring workload-dependent power hot spots. The advantages of the proposed method over the Linux baseline task mapping and the temperature-based mapping method are demonstrated and validated on real commercial chips.

Experiments on a real Intel Core i7 quad-core processor executing PARSEC-3.0 and SPLASH-2 benchmarks show that, compared to the existing Linux scheduler, core and hot spot

temperature can be lowered by 1.15∼1.31°C. In addition, *Hot-Trim* can improve the chip's EM, NBTI and HCI related reliability by 30.2%, 7.0% and 31.1% respectively compared to Linux baseline without any performance degradation. Furthermore, it improves EM and HCI related reliability by 29.6% and 19.6% respectively, and at the same time even further reduces the temperature by half a degree compared to the conventional temperature-based mapping technique. This work is published in [ZST22, ZSG+20].

Chapter 6

Conclusions

In this article, we reviewed the culmination of our work and shared our contributions to the areas of pre-silicon IC reliability analysis, post-silicon power estimation, and a novel hotspot-aware task allocation method.

6.1 Post-Silicon Full-Chip Power Map Modeling

In Chapter 2, we have proposed accurate full chip steady-state power density and thermal map estimation methods for commercial multi-core microprocessors operating under normal conditions with heat sink cooling. The proposed thermal to power map recovery method, based on the first principle of heat transfer, is very efficient and fast, which in contrast with existing nonlinear optimization based methods. Once accurate power density map was estimated and validated with FEM thermal models for back cooling using the IR thermography setup, we proposed a method to build accurate thermal models for commercial processors under heat sink cooling. The methodology is validated by the on-chip sensor reading of the chip once their exact locations are estimated.

Experimental results on a Intel i7-8650U 4-core processor with back side cooling have shown that average absolute error of the computed thermal maps compared to the measured thermal maps is around 1.3°C. Furthermore, the computed thermal maps and the measured thermal maps have 96% similarity (2D correlation). Furthermore, we compared the proposed power map estimation method with a state-of-art Blind Power Identification (BPI) method. The proposed method is at least 100× faster and higher resolution than the BPI method.

Regarding the future work, the proposed method can actually handle transient powers if the input thermal maps become time series maps. The resulting power maps will be time series maps. For different workloads, we need to characterize them in the back-cooling situations via thermal imaging system as we did in this work first. Then we can compute their thermal maps. However, for the real-time and on-line applications, we can't do the off-line thermal-imaging characterization for each individual workload. Machine learning based power map estimation can be applied such as using deep neural networks (DNN) as the methods demonstrated in our machine learning based full-chip thermal map estimation method [SZZ+20a, SZAT21].

6.2 Power Estimation for Commercial TPUs

In Chapter 3, we have proposed a machine-learning-based approach for real-time estimation of full-chip power maps for commercial Google Coral M.2 TPU chips for the first time. The new method focuses on the DNN inference applications on the TPU and apply workload-related features such as the hyperparameters of the DNN networks and TPU resource information generated by TPU compilers as the input of the deep neural network models. To build the dynamic power density map model, we applied generative adversarial networks (GAN) to model the power den-

sity map based on the selected workload-dependent features. Our study showed that the estimated total powers match the manufacturer's total power measurements extremely well. Experimental results further showed that the predictions of power maps are quite accurate, with the RMSE of only $4.98\mathrm{mW/mm^2}$, or 2.6% of the full-scale error. The speed of deploying the proposed approach on an Intel Core i7-10710U is as fast as $6.9\mathrm{ms}$, which is suitable for real-time estimation.

6.3 Full-Chip Thermal Characterization With Heat Sink Cooling

In Chapter 4, we have proposed a novel full-chip steady-state thermal map estimation and characterization approach for the commercial multi-core processors when they run under heat sink cooling, which cannot be directly measured by thermal imaging systems. Additionally, experimental results show 2.2°C average absolute error over a 56 degrees temperature range under heat sink, which indicates about 3.9% error between the computed heat maps and the real thermal maps.

6.4 Thermal and Reliability Management Considering Hot Spots

In Chapter 5, we have proposed a new hot-spot-aware task mapping scheme named *Hot-Trim* to improve the reliability and thermal performance of commercial multi-core processors without degrading the system execution performance. Our method is motivated by the observation that the power density hot spots in cores and their reliability in a multi-core processor are workload dependent and thus can be exploited to improve the reliability of the system. Experiments on a real Intel Core i7 quad-core processor executing PARSEC-3.0 and SPLASH-2 benchmarks show that the core and hot spot temperature can be even reduced by 1.15∼1.31°C. *Hot-Trim* can improve the chip's EM, NBTI and HCI related reliability by 30.2%, 7.0% and 31.1% respectively compared to

Linux baseline without any performance degradation. Furthermore, it improves EM and HCI related reliability by 29.6% and 19.6% while further reduces the temperature by half a degree compared to the conventional temperature-based mapping technique, proving that temperature per-core sensing may not lead to the optimal reliability management solution.

Bibliography

[A$^+$15] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[ACB17] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv e-prints*, page arXiv:1701.07875, Dec. 2017.

[AH15] H. Amrouch and J. Henkel. Lucid infrared thermography of thermally-constrained processors. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 347–352, July 2015.

[AMD] AMD. AMD uProf. `https://developer.amd.com/amd-uprof/`.

[ANR74] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, Jan 1974.

[AvE$^+$14] H. Amrouch, V. M. van Santen, T. Ebi, V. Wenzel, and J. Henkel. Towards interdependencies of aging mechanisms. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 478–485, Nov 2014.

[BBVB16] Francesco Beneventi, Andrea Bartolini, Pascal Vivet, and Luca Benini. Thermal analysis and interpolation techniques for a logic+ wideio stacked dram test chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(4):623–636, 2016.

[Bie11] Christian Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[BM01] David Brooks and Margaret Martonosi. Dynamic thermal management for high-performance microprocessors. pages 171–182, Jan. 2001.

[CL20] Kun-Chih Jimmy Chen and Yuan-Hao Liao. Adaptive machine learning-based temperature prediction scheme for thermal-aware noc system. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2020.

[CNR10] Ryan Cochran, Abdullah Nama Nowroz, and Sherief Reda. Post-silicon power characterization using thermal infrared emissions. pages 331–336, New York, NY, USA, 2010. ACM.

[CRW07] Ayse K Coskun, Tajana Simunic Rosing, and Keith Whisnant. Temperature aware task scheduling in MPSoCs. pages 1659–1664, April 2007.

[DK96] A. Dasgupta and R. Karri. Electromigration reliability enhancement via bus activity distribution. In *33rd Design Automation Conference Proceedings, 1996*, pages 353–356, 1996.

[DNR13] K. Dev, A. N. Nowroz, and S. Reda. Power mapping and modeling of multi-core processors. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 39–44, Sept 2013.

[DSM+14] Anup Das, Rishad A Shafik, Geoff V Merrett, Bashir M Al-Hashimi, Akash Kumar, and Bharadwaj Veeravalli. Reinforcement learning-based inter-and intra-application thermal optimization for lifetime improvement of multicore systems. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014.

[EBSA+12] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. *Micro, IEEE*, 32(3):122–134, May 2012.

[Edg] Edge TPU Compiler. Available from coral.ai/docs/edgetpu/compiler.

[FLWZ17] Yuxiang Fu, Li Li, Kun Wang, and Chuan Zhang. Kalman predictor-based proactive dynamic thermal management for 3-d noc systems with noisy thermal sensors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(11):1869–1882, 2017.

[GJK+08] Siva P. Gurrum, Yogendra K. Joshi, William P. King, Koneru Ramakrishna, and Martin Gall. A compact approach to on-chip interconnect heat conduction modeling using the finite element method. *Journal of Electronic Packaging*, 130:031001.1–031001.8, September 2008.

[GMQ10] Yang Ge, P Malani, and Qinru Qiu. Distributed task migration for thermal management in many-core systems. pages 579–584, 2010.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[GW02] Y. C. Gerstenmaier and G. Wachutka. Rigorous model and network for transient thermal problems. *Microelectronics Journal*, 33:719–725, September 2002.

[HGV+06] Wei Huang, Shougata Ghosh, Siva Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R. Stan. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. 14(5):501–513, May 2006.

[HV14] Vinay Hanumaiah and Sarma Vrudhula. Energy-efficient operation of multicore processors by DVFS, task migration, and active cooling. 63(2):349–360, February 2014.

[HYST14] Xin Huang, Tan Yu, Valeriy Sukharev, and Sheldon X.-D. Tan. Physics-based Electromigration Assessment for Power Grid Networks. In *Proceedings of the 51st Design Automation Conference*, DAC '14, pages 1–6, New York, NY, Jun. 2014. ACM Press.

[IM03] Canturk Isci and Margaret Martonosi. Runtime power monitoring in high-end processors: methodology and empirical datar. In *the 36th Annual International Symposium on Microarchitecture*, pages 93–104, 2003.

[Int] Intel. Intel Performance Counter Monitor (PCM). `https://software.intel.com/en-us/articles/intel-performance-counter-monitor`.

[ISKAK15] Arman Iranfar, Soheil Nazar Shahsavani, Mehdi Kamal, and Ali Afzali-Kusha. A heuristic machine learning-based algorithm for power and thermal management of heterogeneous mpsocs. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 291–296. IEEE, 2015.

[ITR03] Critical Reliability Challenges for The International Technology Roadmap for Semiconductors (ITRS), 2003. In International Sematech Technology Transfer Document 03024377A-TR, 2003.

[JM01] Russ Joseph and Margaret Martonosi. Run-time power estimation in high-performance microprocessors. pages 135–140, 2001.

[KCS12] Joonho Kong, Sung Woo Chung, and Kevin Skadron. Recent thermal management techniques for microprocessors. *ACM Comput. Surv.*, 44(3):13:1–13:42, jun 2012.

[KES+14] Heba Khdr, Thomas Ebi, Muhammad Shafique, Hussam Amrouch, and Jörg Henkel Karlsruhe. mdtm: Multi-objective dynamic thermal management for on-chip systems. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, 2014.

[KSH+06] K.Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. 2006.

[KYM+99] Naohiko Kimizuka, Toyoji Yamamoto, Tohru Mogami, Ken Yamaguchi, Kazuhiro Imai, and Toshiharu Horiuchi. The impact of bias temperature instability for direct-tunneling ultra-thin gate oxide on mosfet scaling. In *1999 Symposium on VLSI Technology. Digest of Technical Papers (IEEE Cat. No. 99CH36325)*, pages 73–74. IEEE, 1999.

[LFQ12] Guanglei Liu, Ming Fan, and Gang Quan. Neighbor-Aware Dynamic Thermal Management for Multi-core Platform. pages 187–192, 2012.

[LLJ+06] P. Liu, H. Li, L. Jin, W. Wu, S. X.-D. Tan, and J. yang. Fast thermal simulation for runtime temperature tracking and management. 25(12):2882–2893, Dec. 2006.

[LTB15] Shiting Lu, Russell Tessier, and Wayne Burleson. Reinforcement learning for thermal-aware many-core task allocation. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 379–384, 2015.

[LTHW15] Z. Liu, S. X.-D. Tan, X. Huang, and H. Wang. Task migrations for distributed thermal management considering transient effects. 23(2):397–401, 2015.

[LTPT09] D. Li, S. X.-D. Tan, E. H. Pacheco, and M. Tirumala. Architecture-level thermal characterization for multi-core microprocessors. 17(10):1495–1507, 2009.

[LVRW95] C. Lasance, H Vinke, H Rosten, and K.-L. Weiner. A novel approach for the thermal characterization of electronic parts. In *Proceedings of the IEEE 11th Annual Semiconductor Thermal Measurement and Management Symposium*, pages 1–9, 1995.

[LZJ+23] Jincong Lu, Jinwei Zhang, Wentian Jin, Sachin Sachdeva, and Sheldon X.-D. Tan. Learning based spatial power characterization and full-chip power estimation for commercial tpus. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ASPDAC '23, pages 98–103, New York, NY, USA, 2023. Association for Computing Machinery.

[MO14] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv e-prints*, page arXiv:1411.1784, November 2014.

[Mul14] COMSOL Multiphysics. Heat transfer module user's guide. *COMSOL version*, 4, 2014.

[NWR13] A. Nowroz, G. Woods, and S. Reda. Power mapping of integrated circuits using ac-based thermography. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(08):1398–1409, aug 2013.

[OT12] Fabian Oboril and Mehdi B. Tahoori. Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, pages 1–12, 2012.

[PMJH20] S. Pagani, P. D. S. Manoj, A. Jantsch, and J. Henkel. Machine learning for power, energy, and thermal management on multicore processors: A survey. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(1):101–116, 2020.

[PSSK13] S. Paek, W. Shin, J. Sim, and L. Kim. Powerfield: A probabilistic approach for temperature-to-power conversion based on markov random field theory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(10):1509–1519, 2013.

[RB18] Sherief Reda and Adel Belouchrani. Blind Identification of Power Sources in Processors, 2018. https://github.com/scale-lab/BPI.

[RDB18a] Sherief Reda, Kapil Dev, and Adel Belouchrani. Blind identification of thermal models and power sources from thermal measurements. *IEEE Sensors Journal*, 18(2):680–691, 2018.

[RDB18b] Sherief Reda, Kapil Dev, and Adel Belouchrani. Blind identification of thermal models and power sources from thermal measurements. *IEEE Sensors Journal*, 18(2):680–691, 2018.

[RRC+18] R Rohith, Vijeta Rathore, Vivek Chaturvedi, Amit Kumar Singh, Srikanthan Thambipillai, and Siew-Kei Lam. Lifesim: A lifetime reliability simulator for manycore systems. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 375–381, 2018.

[SABR04] J. Srinivasan, S.V. Adve, P. Bose, and J.A. Rivers. The case for lifetime reliability-aware microprocessors. In *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, pages 276–287, 2004.

[SABR05] J. Srinivasan, S.V. Adve, Pradip Bose, and J.A. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In *32nd International Symposium on Computer Architecture (ISCA'05)*, pages 520–531, 2005.

[SKC+19] Z. Sun, T. Kim, M. Chow, S. Peng, H. Zhou, H. Kim, D. Wong, and S. X.-.D Tan. Long-term reliability management for multitasking gpgpus. In *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 213–216, July 2019.

[SZA+19] S. Sadiqbatcha, H. Zhao, H. Amrouch, J. Henkel, and S. X.-D. Tan. Hot spot identification and system parameterized thermal modeling for multi-core processors through infrared thermal imaging. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019.

[SZAT21] Sheriff Sadiqbatcha, Jinwei Zhang, Hussam Amrouch, and Sheldon X.-D. Tan. Real-time full-chip thermal tracking: A post-silicon, machine learning perspective. *IEEE Transactions on Computers*, 2021.

[SZZ+20a] S. Sadiqbatcha, Y. Zhao, J. Zhang, H. Amrouch, J. Henkel, and S. X. D. Tan. Machine learning based online full-chip heatmap estimation. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 229–234, 2020.

[SZZ+20b] Sheriff Sadiqbatcha, J. Zhang, H. Zhao, H. Amrouch, J. Hankel, and Sheldon X.-D. Tan. Post-silicon heat-source identification and machine-learning-based thermal modeling using infrared thermal imaging. 2020.

[Tay13] Michael Taylor. A landscape of the new dark silicon design regime. 33(5):8–19, October 2013.

[TTK+19] Sheldon X.-D. Tan, Mehdi Tahoori, Taeyoung Kim, Shengcheng Wang, Zeyu Sun, and Saman Kiamehr. *VLSI Systems Long-Term Reliability – Modeling, Simulation and Optimization*. Springer Publishing, 2019.

[TYMH95] Eiji Takeda, Cary Y Yang, and Akemi Miura-Hamada. Hot-carrier effects in mos devices. 1995.

[WFMS09] X. Wang, S. Farsiu, P. Milanfar, and A. Shakouri. Power trace: An efficient method for extracting the power dissipation profile in an ic chip from its temperature map. *IEEE Transactions on Components and Packaging Technologies*, 32(2):309–316, 2009.

[Wik] WikiChip. Core i7-8650u - intel. https://en.wikichip.org/wiki/intel/core_i7/i7-8650u.

[WJY+07] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan. Efficient power modeling and software thermal sensing for runtime temperature monitoring. 12(3):1–29, 2007.

[WMT+16] Hai Wang, Jian Ma, Sheldon X.-D. Tan, Chi Zhang, He Tang, Keheng Huang, and Zhenghong Zhang. Hierarchical dynamic thermal management method for high-performance many-core microprocessors. 22(1):1:1–1:21, July 2016.

[WOT+95] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The splash-2 programs: Characterization and methodological considerations. *ACM SIGARCH computer architecture news*, 23(2):24–36, 1995.

[ZGL+20] An Zou, Karthik Garimella, Benjamin Lee, Christopher Gill, and Xuan Zhang. F-lemma: Fast learning-based energy management for multi-/many-core processors. In *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, MLCAD '20, page 43–48, New York, NY, USA, 2020. Association for Computing Machinery.

[ZLS+04] S. Zafar, B.H. Lee, J. Stathis, A. Callegari, and Tak Ning. A model for negative bias temperature instability (nbti) in oxide and high /spl kappa/ pfets 13/spl times/-c6d8c7f5f2. In *Digest of Technical Papers. 2004 Symposium on VLSI Technology, 2004.*, pages 208–209, 2004.

[ZSG+20] Jinwei Zhang, Sheriff Sadiqbatcha, Yuanqi Gao, Michael O'Dea, Nanpeng Yu, and Sheldon X.-D. Tan. Hat-drl: Hotspot-aware task mapping for lifetime improvement of multicore system using deep reinforcement learning. In *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, MLCAD '20, page 77–82, New York, NY, USA, 2020. Association for Computing Machinery.

[ZSJT20] J. Zhang, S. Sadiqbatcha, W. Jin, and Sheldon X.-D. Tan. Accurate power density map estimation for commercial multi-core microprocessors. pages 1085–1090, 2020.

[ZSO+21] Jinwei Zhang, Sheriff Sadiqbatcha, Michael O'Dea, Hussam Amrouch, and Sheldon X.-D. Tan. Full-chip power density and thermal map characterization for commercial microprocessors under heat sink cooling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2021.

[ZST22] Jinwei Zhang, Sheriff Sadiqbatcha, and Sheldon X.-D. Tan. Hot-trim: Thermal and reliability management for commercial multi-core processors considering workload dependent hot spots. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2022.