

BOSTON UNIVERSITY
COLLEGE OF ENGINEERING

Dissertation

**MODELING AND OPTIMIZATION OF EMERGING ON-CHIP
COOLING TECHNOLOGIES VIA
MACHINE LEARNING**

by

ZIHAO YUAN

B.S., Central Michigan University, 2015
M.S., University of Southern California, 2017

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2022

© 2022 by
ZHAO YUAN
All rights reserved

Approved by

First Reader

Ayse K. Coskun, PhD
Interim Associate Dean of Educational Initiatives
Professor of Electrical and Computer Engineering

Second Reader

Sherief Reda, PhD
Professor of Engineering and Computer Science
Brown University

Third Reader

Ajay Joshi, PhD
Professor of Electrical and Computer Engineering

Fourth Reader

Rabia Yazicigil, PhD
Assistant Professor of Electrical and Computer Engineering

The greatest challenge to any thinker is stating the problem in a way that will allow a solution.

– Bertrand Russell

Acknowledgments

First, I would like to express my most profound appreciation to my advisor, Prof. Ayse Coskun. This work would not have been possible without her help and support. Her guidance and encouragement allowed me to turn this work into multiple conference and journal papers. I am thankful to Prof. Sherief Reda and Prof. Evelyn Wang, who provided substantial guidance to my research. My sincere thanks must also go to my committee members, Prof. Ajay Joshi and Prof. Rabia Yazicigil. They generously gave their time to offer me valuable suggestions and feedback. I wholeheartedly thank my parents, Lixiu Wang and Tianbo Yuan, for their trust, encouragement, and patience. Without their love and support, this dissertation would not have been possible.

I want to thank my collaborators, Prachi Shukla, Geoffrey Vaartstra, Sofiane Chetoui, Sean Nemptow, Carlton Knox, and Mostafa Said, for supporting me with their expertise and intuition and helping me to solve scientific and technical problems.

I also appreciate the support and help from my colleagues and alumni in our research group: Tiansheng Zhang, Fulya Kaplan, Emre Ates, Onur Sahin, Yijia Zhang, Aditya Narayan, Burak Aksar, Mert Toslali, Daniel Wilson, and Anthony Byrne, Efe Sencan. There is no way to express how much it meant to me to have been a member of the Performance and Energy-Aware Computing Laboratory (PeacLab). These brilliant colleagues and friends inspired me and brought fun and happiness throughout my PhD student life. My appreciation also goes to my friends and colleagues from Integrated Circuits and Systems Group (ICSG) and Computer Architecture and Automated Design Lab (CAAD) at Boston University.

I thank Ankush Varma for the internship opportunity at Intel, and I am grateful for the experience. I also thank Palit Nilanjan, Ujjwal Gupta, and Nikethan Baligar for mentoring me during my internship at Intel. I can confidently say I have grown and learned under

their mentorship.

In this thesis, some parts of the contents in Chapter 2 are reprints of the material from the following paper:

- Zihao Yuan, Sherief Reda, and Ayse K. Coskun. Compact Thermal Modeling of Emerging Cooling Technologies for Processors. To appear in *Embedded Cooling of Electronic Devices Book*. Editors: Madhu Iyengar and Mehdi Asheghi. *World Scientific Publishing Company (WSPC)*, 2022.

Some parts of the contents in Chapter 3 are reprints of the material from the following papers:

- Zihao Yuan, Prachi Shukla, Sofiane Chetoui, Sean Nemptzow, Sherief Reda, and Ayse K. Coskun. PACT: An Extensible Parallel Thermal Simulator for Emerging Integration and Cooling Technologies. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 41, no. 4, pp. 1048-1061, April 2022.
- Zihao Yuan, Tao Zhang, Jeroen Van Duren, and Ayse K. Coskun. Efficient Thermal Analysis of Lab-Grown Diamond Heat Spreaders. In *Proceedings of ASME International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems (InterPACK)*, Nov. 2021.

Some parts of the contents in Chapter 4 are reprints of the material from the following papers:

- Zihao Yuan, Geoffrey Vaartstra, Prachi Shukla, Mostafa Said, Sherief Reda, Evelyn Wang, and Ayse K. Coskun. Two-Phase Vapor Chambers with Micropillar Evaporators: A New Approach to Remove Heat from Future High-Performance Chips. In *Proceedings of 19th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM)*, pp. 456-464, May 2019.

- Zihao Yuan, Geoffrey Vaartstra, Prachi Shukla, Sherief Reda, Evelyn Wang, and Ayse K. Coskun. Modeling and Optimization of Chip Cooling with Two-Phase Vapor Chambers. In Proceedings of *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1-6, Jul. 2019.
- Zihao Yuan, Geoffrey Vaartstra, Prachi Shukla, Zhengmao Lu, Evelyn Wang, Sherief Reda, and Ayse K. Coskun. A Learning-Based Thermal Simulation Framework for Emerging Two-Phase Cooling Technologies. In Proceedings of *Design, Automation and Test in Europe (DATE)*, pp. 400-405, 2020.
- Carlton Knox, Zihao Yuan, and Ayse K. Coskun. Machine Learning and Simulation Based Temperature Prediction on High-Performance Processors. To appear in *ASME International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems (InterPACK)*, 2022.

Some parts of the contents in Chapter 5 are reprints of the material from the following papers:

- Zihao Yuan, Geoffrey Vaartstra, Prachi Shukla, Sherief Reda, Evelyn Wang, and Ayse K. Coskun. Modeling and Optimization of Chip Cooling with Two-Phase Vapor Chambers. In Proceedings of *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1-6, Jul. 2019.
- Zihao Yuan, Geoffrey Vaartstra, Prachi Shukla, Zhengmao Lu, Evelyn Wang, Sherief Reda, and Ayse K. Coskun. A Learning-Based Thermal Simulation Framework for Emerging Two-Phase Cooling Technologies. In Proceedings of *Design, Automation and Test in Europe (DATE)*, pp. 400-405, 2020.
- Zihao Yuan and Ayse K. Coskun. Neural Network-based Cooling Design for High-performance Processors. In *iScience*, vol. 25, no. 1, pp. 103-582, Jan. 2022.

As part of this thesis, the work above has been partially funded by NSF CRI (CI-NEW) grant #1730316/1730003/1730389, NSF CCF grant #1910075/1909027, Diamond Foundry, and Boston University's Undergraduate Research Opportunities Program.

**MODELING AND OPTIMIZATION OF EMERGING ON-CHIP
COOLING TECHNOLOGIES VIA
MACHINE LEARNING**

ZIHAO YUAN

Boston University, College of Engineering, 2022

Major Professor: Ayse K. Coskun, PhD
Interim Associate Dean of Educational Initiatives
Professor of Electrical and Computer Engineering

ABSTRACT

Over the last few decades, processor performance has continued to grow due to the down-scaling of transistor dimensions. This performance boost has translated into high power densities and localized hot spots, which decrease the lifetime of processors and increase transistor delays and leakage power. Conventional on-chip cooling solutions are often insufficient to efficiently mitigate such high-power-density hot spots. Emerging cooling technologies such as liquid cooling via microchannels, thermoelectric coolers (TECs), two-phase vapor chambers (VCs), and hybrid cooling options (e.g., of liquid cooling via microchannels and TECs) have the potential to provide better cooling performance compared to conventional cooling solutions. However, these potential solutions' cooling performance and cooling power vary significantly based on their design and operational parameters (such as liquid flow velocity, evaporator design, TEC current, etc.) and the chip specifications. In addition, the cooling models of such emerging cooling technologies may require additional Computational Fluid Dynamics (CFD) simulations (e.g., two-phase cooling), which are time-consuming and have large memory requirements. Given the vast solution space

of possible cooling solutions (including possible hybrids) and cooling subsystem parameters, the optimal solution search time is also prohibitively time-consuming. To minimize the cooling power overhead while satisfying chip thermal constraints, there is a need for an optimization flow that enables rapid and accurate thermal simulation and selection of the best cooling solution and the associated cooling parameters for a given chip design and workload profile.

This thesis claims that combining the compact thermal modeling methodology with machine learning (ML) models enables rapidly and accurately carrying out thermal simulations and predicting the optimal cooling solution and its cooling parameters for arbitrary chip designs. The thesis aims to realize this optimization flow through three fronts. First, it proposes a parallel compact thermal simulator, PACT, that enables speedy and accurate standard-cell-level to architecture-level thermal analysis for processors. PACT has high extensibility and applicability and models and evaluates thermal behaviors of emerging integration (e.g., monolithic 3D) and cooling technologies (e.g., two-phase VCs). Second, it proposes an ML-based temperature-dependent simulation framework designed for two-phase cooling methods to enable fast and accurate thermal simulations. This simulation framework can also be applied to other emerging cooling technologies. Third, this thesis proposes a systematic way to create novel deep learning (DL) models to predict the optimal cooling methods and cooling parameters for a given chip design. Through experiments based on real-world high-power-density chips and their floorplans, this thesis aims to demonstrate that using ML models substantially minimizes the simulation time of emerging cooling technologies (e.g., up to $21\times$) and improves the optimization time of emerging cooling solutions (e.g., up to $140\times$) while achieving the same optimization accuracy compared to brute force methods.

Contents

1	Introduction	1
1.1	Motivation and Key Contributions	1
1.2	Dissertation Organization	7
2	Background and Related Work	8
2.1	Compact Thermal Modeling Methodology	8
2.2	Emerging Cooling Technologies and CTMs	9
2.2.1	TECs	10
2.2.2	Liquid Cooling via Microchannels	13
2.2.3	Hybrid Cooling	15
2.3	Cooling Optimization Methods	17
3	Enabling Fast and Accurate Parallel Thermal Simulations with PACT	18
3.1	Introduction	18
3.2	Background of Existing Compact Thermal Simulators	22
3.3	Proposed SPICE-Based Thermal Simulator	24
3.3.1	PACT Simulation Flow	25
3.3.2	Thermal Netlist and SPICE Circuit Components	26
3.3.3	Extensibility of PACT	28
3.3.4	Compatibility of PACT	34
3.3.5	OpenROAD Interface	34
3.3.6	PACT Solver	36
3.4	Experimental Results	39

3.4.1	Speed Analysis with Complex 2D and Monolithic 3D ICs	40
3.4.2	Full System Simulation of 2.5D Systems with PNoC	41
3.4.3	Liquid Cooling via Microchannels Simulation Results	42
3.4.4	Standard-Cell-Level Validation of PACT against COMSOL and HotSpot	46
3.4.5	Standard-Cell-Level Comparison of PACT against MTA	53
3.5	Case Study: Modeling Diamond Heat Spreaders Using PACT	54
3.5.1	Introduction	55
3.5.2	Materials and Methods	56
3.5.3	Results and Discussions	60
4	Modeling Emerging Cooling Methods via Machine Learning	75
4.1	Introduction	75
4.2	Two-Phase VCs with Micropillar Wick Evaporators CTM	76
4.2.1	Background on VCs	76
4.2.2	Compact Modeling Methodology	77
4.2.3	Dry-out Heat Flux Analytical Model	79
4.2.4	Parametric Study	80
4.2.5	Validation of the Proposed Model	81
4.2.6	Cooling Performance Evaluation	83
4.3	Two-Phase VCs with Hybrid Wick Evaporators CTM	85
4.3.1	Compact Modeling Methodology	85
4.3.2	COMSOL Model	87
4.4	An ML-Based Thermal Simulation Framework for Emerging Two-Phase Cooling Technologies	88
4.4.1	A Temperature-Dependent HTC Simulation Framework	89
4.4.2	An ML-Based Temperature-Dependent HTC Simulation Framework	89
4.4.3	Validation of the ML Model	91

4.4.4	Validation of the ML-Based Temperature-Dependent HTC Simulation Framework	92
4.5	Improved ML-Based Simulation Framework for Two-Phase VCs	94
4.5.1	Improved Compact Modeling of the Two-Phase VCs	95
4.5.2	Improved ML-Based Temperature-Dependent HTC Simulation Framework	98
4.5.3	Cooling Performance Evaluation of Two-Phase VCs on Realistic Mobile System	99
4.6	Predicting Thermal Profiles via ML	100
4.6.1	Methodology	102
4.6.2	Results and Discussions	106
5	Optimizing Emerging Cooling Methods for High-Performance Processors via Deep Learning	116
5.1	Introduction	116
5.2	Two-Phase VCs Optimization Flows	117
5.2.1	Two-Phase VCs with Micropillar Wick Evaporators Optimization via Grid Search	117
5.2.2	Two-Phase VCs with Hybrid Wick Evaporators Optimization via MSA	118
5.3	Emerging Cooling Methods Optimization via CMA-ES	120
5.4	Emerging Cooling Methods Optimization via DL	125
5.4.1	Overall CNN Optimization Architecture	127
5.4.2	Training Data Preparation	128
5.4.3	Hybrid Cooling CNN Architecture	129
5.4.4	Two-Phase VCs with Hybrid Wick Evaporators CNN Architecture .	131
5.4.5	Results and Discussions	133

6	Conclusions and Future Work	140
6.1	Enabling Fast and Accurate Parallel Thermal Simulations with PACT	140
6.2	Modeling Emerging Cooling Methods via Machine Learning	142
6.3	Optimizing Emerging Cooling Methods for High-Performance Processors via Deep Learning	143
	References	146
	Curriculum Vitae	156

List of Tables

3.1	Solvers, cooling methods, and inputs of PACT and of existing compact thermal simulators (e.g., HotSpot (Skadron et al., 2003), 3D-ICE (Sridhar et al., 2014), and ThermalScope (Allec et al., 2008)). BE stands for the Backward Euler solver, and TRAP is a hybrid solver of the Backward Euler and the Trapezoidal method. Full industrial design means that PACT takes real-world standard-cell designs as input, such as designs from OpenROAD.	20
3.2	Information about available solvers in PACT.	37
3.3	Experimental setup of monolithic 3D chip and the SCC-based chip simulations.	41
3.4	Simulation results of the monolithic 3D chip and the SCC-based chip.	41
3.5	Experimental setup of PNoC simulations.	42
3.6	PNoC simulation results.	44
3.7	Validation setup of liquid cooling via microchannels simulations.	46
3.8	Validation setup of HotSpot, COMSOL, and PACT.	48
3.9	Statistics of the realistic MPSoCs from the OpenROAD benchmark set.	49
3.10	Intel i7 6950× steady-state results. ID stands for chip stack identification number. T_{max} stands for maximum temperature ($^{\circ}C$) and ΔT stands for temperature gradient ($^{\circ}C$).	65
3.11	IBM Power9 steady-state results. ID stands for chip stack identification number. T_{max} stands for maximum temperature ($^{\circ}C$) and ΔT stands for temperature gradient ($^{\circ}C$).	65

3.12	PicoSoC steady-state results. ID stands for chip stack identification number. T_{max} stands for maximum temperature ($^{\circ}C$) and ΔT stands for temperature gradient ($^{\circ}C$).	65
3.13	Liquid cooling via microchannels material properties.	72
4.1	Coolant and micropillar parameters.	80
4.2	Comparison between proposed CTM and COMSOL.	83
4.3	Structural properties and simulation parameters.	83
4.4	Optimal geometries (h, d, i) of two-phase VCs and estimated pumping power of liquid cooling via microchannels and microchannel-based two-phase cooling ($G = 560 \text{ kg}/\text{m}^2\text{s}$).	84
4.5	Hybrid wick geometry parameters and valid range.	91
4.6	Worst-case results from the k-fold CV tests. MAE stands for mean absolute error, RMSE stands for root mean square error. The errors are normalized with respect to the golden HTC data.	92
4.7	Hybrid wick geometries for validation tests.	93
4.8	Naming conventions of the parameters.	95
4.9	5-fold CV results.	112
5.1	Details parameters for hybrid cooling CNN alternatives.	130
5.2	Accuracy results for different hybrid cooling CNN regression alternatives.	131
5.3	Accuracy results for different activation functions. Accuracy and R2 score are averaged for liquid flow velocity and current.	132
5.4	Detailed parameters for two-phase VCs with hybrid wick evaporators CNN alternatives.	133
5.5	Accuracy results for different two-phase VCs with hybrid wick evaporators CNN regression alternatives.	133
5.6	Validation results of the proposed CNN architectures.	134

5.7 Predicted parameters using our proposed CNN architectures and baseline parameters generated using the baseline methods for PicoSoC. 136

5.8 IBM Power9 processor power breakdown. 139

5.9 IBM Power9 processor optimal cooling parameters, maximum temperature, and cooling power. 139

List of Figures

2·1	(a) A simple thermal RC circuit (Pedram and Nazarian, 2006). R is the thermal resistor, C is the thermal capacitor, v_0 is the ambient temperature, and v is the temperature of the node. (b) An equivalent RC network to model temperature distribution (Pedram and Nazarian, 2006).	9
2·2	(a) Default grid cell and (b) TEC grid cell (Kaplan et al., 2017).	12
2·3	COMSOL model of the TEC device (Kaplan et al., 2017).	12
2·4	Liquid cooling via microchannels 3D IC architecture (Sridhar et al., 2014).	13
2·5	(a) Default grid cell and (b) Liquid grid cell (Sridhar et al., 2013a).	14
2·6	Front view of a hybrid cooling design (Kaplan et al., 2019).	16
3·1	Temperature profiles for a standard-cell design at various grid resolutions. The maximum temperature and thermal gradient plots are expected to saturate with higher grid resolutions ($> 1024 \times 1024$).	20
3·2	PACT simulation flow.	25
3·3	SPICE circuit component usage in PACT.	28
3·4	The high-level simulation flow with the medium-cost heat sink.	29
3·5	A small section of a liquid-cooled chip stack.	31
3·6	(a) The high-level simulation flow with liquid cooling via microchannels and (b) the additional liquid cooling library file for implementing a CTM for liquid cooling via microchannels.	33
3·7	PNoC simulation framework.	35
3·8	The flow diagram of OpenROAD.	36

3·9	Transient simulation time of a two-layer chip stack.	37
3·10	Thermal maps for running application <i>bt</i> with 96 threads and 10% performance constraint using original PNoC simulation framework and PNoC simulation framework using PACT. MRRG is on the interposer layer. The number of grids used in the simulation is set to 64×64	43
3·11	Transient temperature results for running application <i>hpccg</i> with 96 threads and 10% performance constraint using original PNoC simulation framework and PNoC simulation framework using PACT. The number of grids used in the simulation is set to 64×64 . The left image shows the average power traces and the right image shows the average temperature traces.	44
3·12	(a) The front view of the chip stack and (b) the microchannel layer thermal map (power density = $100 W/cm^2$ and coolant velocity of $0.5 m/s$).	45
3·13	Liquid cooling via microchannels simulation results. The top image shows the maximum temperature difference for each power profile when coolant flow velocity = 0.5, 1, 1.5, and $2 m/s$. The bottom image shows the transient temperature curve of PACT and 3D-ICE when power density = $100 W/cm^2$ and liquid flow velocity = $0.5 m/s$. This case shows the maximum temperature difference between PACT and 3D-ICE.	47
3·14	PACT's thermal maps for the MPSoCs from OpenROAD. The number of grids used in the simulation is set to 256×256 . Different utilization levels (shown next to chip names) affect floorplan, chip size, and power density.	48
3·15	Thermal maps for PicoSoC with 95% utilization simulated using HotSpot, PACT, and COMSOL. The rightmost image shows the error map of PACT when compared to HotSpot. The number of grids used in the simulation is set to 256×256	49

3·16	Steady-state grid temperature validation results (utilization = 95%). MP-SoCs with 95% utilization result in the highest maximum, average, and minimum grid temperature error. The error is calculated with respect to COMSOL.	50
3·17	Steady-state and transient simulation times of PACT. The speedup of PACT against HotSpot is shown on the y-axis. The speedup is computed as the ratio of the simulation times of HotSpot and PACT. Negative values mean HotSpot is faster than PACT for those cases.	50
3·18	Transient validation results. The number of grids used in the simulation is set to 256×256. Due to the space limit, we only show the results that have the highest transient temperature difference.	51
3·19	Synthetic power traces for PACT and HotSpot simulations. Due to the space limit, we only show the results that have the highest temperature difference.	52
3·20	Steady-state and transient simulation time of PACT and MTA.	54
3·21	Chip stacks for IBM Power9 and Intel i7 6950×.	57
3·22	Chip stacks for PicoSoC.	58
3·23	Intel i7 6950× and IBM Power9 floorplans.	58
3·24	The layer stack of the interconnect model.	59
3·25	Power delivery network model.	59
3·26	Chip stacks for steady-state validation.	61
3·27	Chip stacks and die floorplan for transient validation.	62
3·28	Steady-state heat map comparisons of Intel i7 6950× (Chip stacks 1 and 3).	66
3·29	Steady-state heat map comparisons of IBM Power9 (Chip stacks 1 and 3).	66
3·30	Steady-state heat map comparisons of PicoSoC (Chip stacks 1 and 3).	66
3·31	Transient temperature plots for Intel i7 6950× and PicoSoC.	68

3.32	Steady-state silicon layer thickness parametric study results for Intel i7 6950×.	70
3.33	Intel i7 6950× chip stack 1 silicon layer thickness transient parametric study results.	70
3.34	Intel i7 6950× chip stack 3 silicon layer thickness transient parametric study results.	71
3.35	Steady-state heat maps for parametric study of cooling packages. Liquid flow velocity is set to 2.6 <i>m/s</i>	72
3.36	Intel i7 6950× silicon layer steady-state cooling package parametric study results (maximum temperature).	73
3.37	Intel i7 6950× silicon layer transient cooling package parametric study results.	74
4.1	(a) VC structure view, (b) micropillar wick side view, and (c) micropillar wick side view overall view.	77
4.2	(a) Default grid cell and (b) proposed grid cells for modeling two-phase VCs with micropillar wick evaporators.	78
4.3	Parametric study for different micropillar wick geometries. Dry-out heat flux is shown on the right axes.	82
4.4	Comparison of cooling performance of two-phase VCs with micropillar evaporators (VC), liquid cooling via microchannels (liquid), and microchannel-based two-phase cooling (two-phase) when flow velocity = 2.6 <i>m/s</i> and mass flow velocity = 300 <i>kg/m²s</i> . Results are normalized to liquid cooling when $P_{hs} = 2000 \text{ W/cm}^2$	84
4.5	A hybrid wick evaporator cross-section view.	85

4·6	(a) The chip stack of the processing layer and two-phase VCs with hybrid wick evaporators, (b) processing layer grid cell, and (c) hybrid wick layer grid cell.	86
4·7	Temperature-dependent HTC simulation framework and our proposed ML-based temperature-dependent HTC simulation framework.	90
4·8	Floorplans used in validation. Dimensions are in <i>mm</i>	93
4·9	Non-uniform power profile maximum and average temperature error validation results. PD_{H_s} stands for hot spot power density.	94
4·10	(a) The chip stack of the processing layer and two-phase VCs with hybrid wick evaporators, (b) processing layer grid cell, (c) hybrid wick layer grid cell, (d) vapor core grid cell, and (e) condenser grid cell.	96
4·11	Improved ML-based temperature-dependent simulation framework.	99
4·12	Steady-state heat maps.	100
4·13	Transient comparison results.	100
4·14	Diagram of the ML model.	105
4·15	Original total chip power from McPAT, split by the number of enabled cores.	107
4·16	Scaled total chip power, split by the number of enabled cores.	108
4·17	Average power of each application in the NAS parallel benchmarks.	108
4·18	Average temperatures for applications ($^{\circ}C$), split by the number of enabled cores used.	109
4·19	Average temperatures ($^{\circ}C$) for the number of enabled cores to run the applications, split by application and sorted in ascending order by average power.	110
4·20	Histogram of R2 score distribution across 300 different thermal sensor placements used as input to the model.	110

4.21	Histogram of RMSE distribution across 300 different thermal sensor placements used as input to the model.	111
4.22	Selected thermal sensor placement for our experiments. Red markers indicate thermal sensors.	111
4.23	Heat map comparison for the worst case. The left heat map is the golden heat map, and the right heat map is the predicted heat map.	112
4.24	Average LOOCV R2 scores on applications.	113
4.25	LOOCV RMSEs on applications.	114
4.26	Heat map comparisons for the worst case when the number of cores used is equal to 5. The left heat map is the golden heat map, and the right heat map is the predicted heat map.	114
4.27	Average LOOCV R2 scores on enabled cores.	115
4.28	LOOCV RMSEs on enabled cores.	115
4.29	Heat map comparison for the worst case when the number of enabled cores is equal to 10. The left heat map is the golden heat map, and the right heat map is the predicted heat map.	115
5.1	Experimental floorplans. Dimensions are in <i>mm</i>	120
5.2	Proposed optimization flow.	122
5.3	Synthetic chip floorplans.	126
5.4	Results for on-chip temperature constraint = 65°C. The format for two-phase VCs with hybrid wick evaporator is {coolant, hot spot temperature, cooling power}. The format for hybrid cooling is {liquid flow velocity, TEC current, hot spot temperature, cooling power}.	126
5.5	DL-based cooling optimization flow.	127
5.6	Optimal hybrid cooling CNN architecture.	131
5.7	CNN architectures accuracy results.	136

5.8 The correlation plots of the maximum temperatures and cooling costs predicted using the proposed optimization flow against the baseline methods' results. Baseline methods stand for the combination of MSA and CMA-ES. CNN stands for the proposed CNN architectures and optimization flow. All the data are normalized to the maximum value. 138

List of Abbreviations

2RM	2-resistor model
4RM	4-resistor model
BE	Backward Euler
CFD	Computational Fluid Dynamics
CMA-ES	Covariance matrix adaptation evolution strategy
CNN	Convolutional Neural Network
COP	Coefficient of performance
CTM	Compact thermal model
CV	Cross-validation
DEF	Design exchange format
DL	Deep Learning
DSMC	Direct Simulation Monte Carlo
FEM	Finite element method
GDR	Gradient boosting regression
GDS	Graphic data stream
HTC	Heat transfer coefficient
IC	Integrated circuit
ID	Chip stack identification number
ISA	Instruction set architecture
LOOCV	Leave one out cross-validation
MAE	Mean absolute error
MGHPCC	Massachusetts Green High-Performance Computing Center
MIV	Monolithic inter-tier vias
ML	Machine Learning
MPSoC	Multiprocessor system-on-chips
MRRG	Microring Resonator Group
MRR	Microring Resonator
MSA	Multi-start simulated annealing
MSE	Mean square error
MTA	Manchester Thermal Analyzer
NNR	Neural network regression
PACT	A parallel compact thermal simulator
PNoC	Photonic network-on-chip
PWL	Piece-wise linear
RK4	4 th order Runge-Kutta

RMSE	Root mean square error
RTL	Register-transfer level
SPICE	Simulation Program with Integrated Circuit Emphasis
SVR	Support vector regression
TDP	Thermal design power
TEC	Thermoelectric cooler
TIM	Thermal interface material
TSV	Through-silicon vias
VC	Vapor chamber

Chapter 1

Introduction

1.1 Motivation and Key Contributions

Over the last few decades, on-chip power densities have grown tremendously following the down-scaling of transistors. Power densities that reach $1\text{-}2\text{ KW}/\text{cm}^2$ caused by the performance boost of scaling already occur in high-performance chips and result in amplified localized hot spots (Schultz et al., 2016). These on-chip hot spots not only degrade the performance of the chip but also generate larger sub-threshold leakage power and create reliability challenges (Schultz et al., 2016; Saini and Mehra, 2012).

Conventional on-chip cooling solutions such as forced air cooling via fans or pin-fin heat sinks are often insufficient to mitigate high-power-density hot spots and result in over/under-cooling. Emerging cooling technologies such as liquid cooling via microchannels (Dang et al., 2010), thermoelectric coolers (TECs) (Chowdhury et al., 2009), two-phase vapor chambers (VCs) (Bulut et al., 2019), and hybrid cooling options (Yazawa et al., 2012) (e.g., of liquid cooling via microchannels and TECs) have the potential to provide better cooling performance compared to the conventional cooling solutions. However, there is no obvious winner in terms of cooling efficiency among all these emerging cooling technologies. These potential solutions' cooling performance and cooling power vary significantly based on the cooling parameters (such as liquid flow velocity, evaporator design, TEC current, etc.) (Yuan et al., 2020; Yuan et al., 2019a; Yuan et al., 2019b). The cooling technologies and the cooling parameters also need to consider the chip architecture, chip size, floorplan, and the power profiles of the applications running on the given chip.

To minimize the cooling power while satisfying chip thermal constraints, there is a need for an optimization flow that enables rapid and accurate selection of the optimal cooling solution and the associated cooling parameters for a given chip and application profile.

A key enabler to such a cooling design optimization flow is a set of accurate and fast models for various cooling technologies. A common approach towards this direction is using compact thermal models (CTMs) that model heat dissipation with an equivalent lumped circuit model (Pedram and Nazarian, 2006). However, existing thermal simulation tools are limited by several major challenges that prevent them from providing fast solutions to large problem sizes, which are necessary to conduct standard-cell-level thermal analysis or to evaluate new technologies or large chips. In addition, some specific cooling methods, such as two-phase cooling methods, require additional Computational Fluid Dynamics (CFD) simulations. CFD simulations are often time-consuming and have large memory requirements (e.g., simulating a *mm*-scale chip model can take from hours to multiple days and easily requires tens of GBs of memory). Given the vast solution space of possible cooling solutions (including possible hybrids) and cooling parameters, the optimal solution search time is still prohibitively time-consuming with CTMs (Yuan et al., 2019a). In addition to cooling design choice possibilities, the optimization flow also needs to consider the specific chip design and power profile changes. In this case, using a simple grid search to find the optimal cooling design for even a small-sized chip floorplan and its typical power profile could easily take up to days (Yuan et al., 2019a).

This thesis claims that combining a compact thermal modeling methodology with machine learning (ML) models enables rapidly and accurately predicting the optimal cooling solution and its cooling parameters for arbitrary chip designs.

The thesis aims to realize this optimization flow through three fronts:

- **Enabling fast and accurate parallel thermal simulations with PACT:** We pro-

pose a parallel compact thermal simulator, PACT¹, that enables speedy and accurate standard-cell-level to architecture-level thermal analysis for processors. PACT has the following features: (i) it utilizes the parallelism in modern computing systems to conduct parallel thermal simulations to speed up the process of solving problems with a large number of grid nodes (e.g., for standard-cell-level problems or modeling the ultra-thin layers in a monolithic 3D stack), (ii) it offers support for various steady-state and transient solvers to speed up simulation time while maintaining the desired accuracy level, and (iii) it can be easily extended to support emerging integration and cooling technologies by modifying the thermal netlist. We interface PACT with OpenROAD (Ajayi et al., 2019), an end-to-end silicon compiler to allow the evaluation of thermal behaviors of full standard-cell-level industry designs directly. We validate PACT’s accuracy by comparing it to COMSOL, using full standard-cell-level industrial designs provided by OpenROAD. Compared to COMSOL, PACT has a maximum temperature error of 2.77% for steady-state and 3.28% for transient simulation. To demonstrate the applicability of PACT, we run standard-cell-level to architecture-level thermal simulations with realistic 2D and monolithic 3D integrated circuits (ICs) using PACT and a well-known compact thermal simulator, HotSpot (Skadron et al., 2003). Compared to HotSpot, PACT reduces the steady-state simulation time from more than 3 hours to only 16 minutes for simulating a monolithic 3D IC. PACT also speeds up the transient simulation time of a 256-core 2D IC from more than three days to less than 19 minutes. When simulating the full standard-cell-level industrial designs from OpenROAD, PACT shows up to 232× speedup with the same accuracy level compared to HotSpot. PACT is able to model and evaluate the thermal behaviors of emerging integration (e.g., monolithic 3D) and cooling technologies (e.g., two-phase vapor chambers). Because of the fast simulation speed and the high extensibility, PACT enables the co-design of the computing system and

¹PACT is open-sourced at <https://github.com/peaclab>.

cooling system to achieve better energy efficiency or higher computing performance improvements under the temperature constraints (see Chapter 3 for further details).

To further demonstrate the extensibility and applicability of PACT, we model *lab-grown diamond heat spreaders* in PACT and conduct steady-state and transient thermal simulations with various high-performance chips to evaluate the cooling performance of lab-grown diamond heat spreaders. We run benchmark applications on real-world-like high-performance chips using popular architecture-level performance and power simulators to obtain transient power traces. The generated transient power traces are used as inputs to PACT to perform transient thermal analysis with lab-grown diamond heat spreaders and traditional copper heat spreaders. For each high-performance chip under test, we evaluate the thermal maps, maximum temperature reductions, and thermal gradient reductions with diamond heat spreaders versus traditional copper heat spreaders. Simulation results show that lab-grown diamond heat spreaders achieve maximum temperature and thermal gradient reductions of up to 26.73°C and 13.75°C when compared to traditional copper heat spreaders, respectively (see Chapter 3.5 for further details).

- **Modeling emerging cooling methods via machine learning:** We introduce ML-enabled modeling methodologies for emerging cooling technologies. Two-phase cooling with VCs is a prevalent cooling method for computing systems with tight power and thermal budget (e.g., mobile systems) as it removes heat effectively and requires minimum additional cooling power (Bulut et al., 2019). Developing fast and accurate two-phase VCs thermal models facilitates early-stage design space exploration and co-optimization for the computing system and this cooling technology. In this thesis, we propose a steady-state CTM for two-phase VCs with micropillar wick evaporators and use the CTM to compare the cooling performance (i.e., hot spot temperature reductions and thermal gradients) against liquid cooling via microchannels

and microchannel-based two-phase cooling (see Chapter 4.2 for further details). We also propose a generalized ML-based temperature-dependent heat transfer coefficient (HTC) simulation framework for two-phase cooling solutions. To demonstrate the simulation speedup and accuracy of our proposed simulation framework, we build a CTM for two-phase VCs with hybrid wick evaporators (of nanoporous membrane and microchannels) and integrate it into our proposed simulation framework. Our proposed ML-based simulation framework achieves a $21\times$ speedup compared to the COMSOL with an average error of 0.98°C (see Chapters 4.3 and 4.4 for further details). We further extend the ML-based thermal simulation framework with the support of transient simulation and vapor core modeling (see Chapter 4.5 for further details). To mitigate the on-chip thermal sensors inaccuracies, we propose an ML and simulation-based temperature profile prediction methodology to predict the heat map based on the measurements from the on-chip digital thermal sensors. Experimental results when running a set of realistic benchmark applications show that our proposed ML and simulation-based temperature profile prediction approach accurately predicts temperatures within an error of less than 0.25°C (see Chapter 4.6 for further details).

- **Optimizing emerging cooling methods for high-performance processors via deep learning:** Researchers have developed fast models for various emerging cooling methods (Kaplan et al., 2014; Kaplan and Coskun, 2015; Kaplan et al., 2017; Kaplan et al., 2019; Sridhar et al., 2013a; Sridhar et al., 2010; Sridhar et al., 2013b; Vaartstra et al., 2019). To select and optimize a cooling solution for a given chip and power profile, there is a need for a fast and accurate optimization flow that selects the most power-efficient cooling solution and its cooling parameters for a given target chip and power profile, such that the cooling power is minimized and the chip temperature stays below a safe threshold. To design such a cooling optimization flow, we

perform two main steps. First, we propose the cooling parameters optimization flow to optimize the cooling performance of two-phase VCs with micropillar wick and hybrid wick evaporators. We demonstrate the accuracy and speedup of this cooling parameters optimization flow against the exhaustive search approach. The proposed two-phase VCs optimization flow is capable of finding better (or similar) cooling parameters than the exhaustive search approach with an average speedup of $4.37\times$ (see Chapter 5.2 for further details). Second, we design the target cooling optimization flow that selects the most power-efficient cooling solution and its cooling parameters for a given target chip and power profile (see Chapter 5.3 for further details). However, the optimization time of this cooling optimization flow still takes up to days depending on the granularity of the cooling parameters solution space. To further speed up the optimization time, we propose a systematic way to train novel deep learning (DL) models to predict the optimal cooling methods and cooling parameters for a given chip design at design time. A DL regression model learns the intrinsic information among the chip designs and the cooling solutions, and then generates the optimal cooling solution and the cooling parameters, given a specific chip floorplan and power profile. We have designed multi-output convolutional neural networks (CNNs) to estimate the best cooling method and its cooling design and technology parameters. The cost function used to evaluate the output of the CNN is a function of cooling power, hot spot temperatures, and temperature constraint of the chip. Our results confirm that, when compared to existing optimization methods with the same cost function, our proposed CNN architectures and DL-based optimization flow successfully predict the optimal cooling solution and cooling parameters with a maximum error of less than 4% and a maximum speedup of $140\times$ (see Chapter 5.4 for further details).

1.2 Dissertation Organization

The rest of the thesis begins with a discussion on the background and related work of the compact thermal modeling methodology, emerging cooling methods and their CTMs, and optimization of emerging cooling methods for high-performance processors. In Chapter 3, we introduce PACT, a fast and accurate parallel compact thermal simulator that has high applicability and extensibility. We evaluate PACT's accuracy and speed against existing compact thermal simulators and present a case study on modeling lab-grown diamond heat spreaders to demonstrate PACT's applicability and extensibility. Chapter 4 shows the detail of utilizing the compact thermal modeling methodology with ML models to build a fast and accurate two-phase cooling simulation framework. We also discuss the ML and simulation-based temperature profile prediction methodology in Chapter 4. In Chapter 5, we discuss the optimization methods for the emerging cooling models we build in Chapter 4 and provide the DL-based optimization flow that is used to search for the optimal cooling method and its cooling parameters given a chip design and power profile.

Chapter 2

Background and Related Work

This thesis proposes thermal modeling and optimization of emerging cooling methods using ML models to speed up the thermal simulation time and cooling methods optimization time. In this chapter, we first briefly elaborate on the fundamental of compact thermal modeling methodology. We continue to discuss popular emerging cooling methods for processors and their corresponding CTMs. Finally, we discuss the potential optimization angle to speed up optimization for the optimal cooling solution and its parameters for a given chip design.

2.1 Compact Thermal Modeling Methodology

Compact thermal modeling has been designed to overcome the thermal simulation challenges of long simulation times and large memory requirements. A compact model leverages the duality between electrical and thermal properties to model temperature (Pedram and Nazarian, 2006).

The heat diffusion equation is generally used to describe the heat conduction and calculate the temperature distribution of a chip. For homogeneous materials, the corresponding heat equation is shown as follows:

$$\rho c_p \frac{\partial u}{\partial t} = k \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) + q. \quad (2.1)$$

In Equation (2.1), k is the thermal conductivity (W/mK), ρ is the density of the material (kg/m^3), c_p is the specific heat (J/kgK), q is the power density (W/cm^2), and u is the

temperature ($^{\circ}C$) of the location (x, y, z) at time t . There is a well-known duality between heat flow and electric current. The heat flow (W) passing through a thermal resistor ($^{\circ}C/W$) is represented as the electric current (A) flowing through an electrical resistance (Ω). The corresponding temperature difference ($^{\circ}C$) is equivalent to the voltage drop (V). In addition, there is also a thermal capacitance ($J/^{\circ}C$) that describes how much heat is absorbed, which is represented as the electric capacitance (F). Node temperature is then modeled as the node voltage of an electric RC circuit as shown in Figure 2.1 (a). To model a chip with multiple heat sources, heat conduction from each vertical and horizontal node is modeled as thermal resistance. Node n_k represents the temperature of the circuit block, and the current source, i_k , represents the power consumption of the corresponding node. v_0 is the ambient temperature, and C_{k0} represents the thermal capacitance of the node. A thermal RC network can be built based on the above parameters as shown in Figure 2.1 (b).

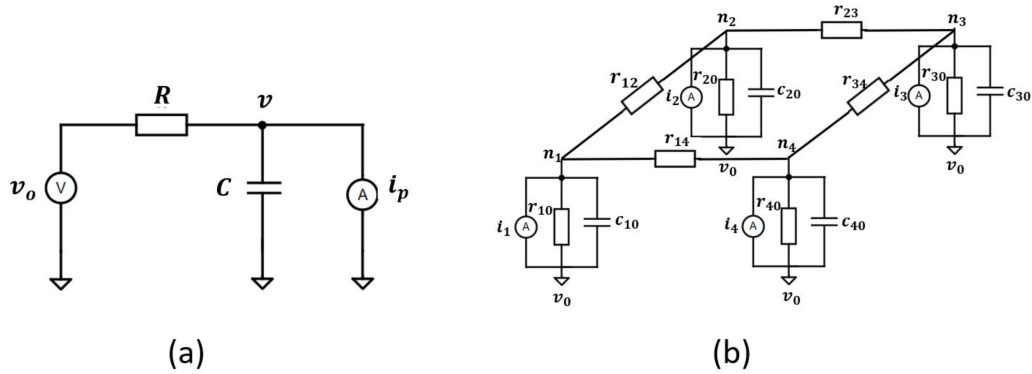


Figure 2.1: (a) A simple thermal RC circuit (Pedram and Nazarian, 2006). R is the thermal resistor, C is the thermal capacitor, v_0 is the ambient temperature, and v is the temperature of the node. (b) An equivalent RC network to model temperature distribution (Pedram and Nazarian, 2006).

2.2 Emerging Cooling Technologies and CTMs

To overcome the high power density challenges for processor cooling, researchers have developed several emerging cooling solutions targeting various architectures and cooling

scenarios: (i) TECs are attractive due to their hot spot mitigation ability; (ii) liquid cooling via microchannels can be used as an inter-layer cooling method to solve the thermal-coupling problems for 3D-stacked chips; and (iii) compared to TEC and liquid cooling via microchannels, hybrid cooling (of liquid cooling via microchannels and TEC) enhances cooling performance while minimizing the cooling power. We next discuss emerging cooling methods and compact thermal modeling methodologies for the aforementioned emerging cooling solutions.

2.2.1 TECs

TEC units have gained attraction due to their abilities to remove heat from high power density hot spots effectively (Chowdhury et al., 2009). A TEC unit operates based on the Peltier effect such that when an electric current passes through a TEC unit, heat is absorbed from one side (cold side) and rejected on the other side (hot side) (Chowdhury et al., 2009; Yazawa et al., 2012). TEC units are typically placed directly above hot spots. The amount of heat removed by a TEC unit depends on the Seebeck coefficient (S), applied current (I), electric resistance (ρ_{TEC}), and the temperature difference between the cold side and hot side ($T_h - T_c$). Superlattice-based thin-film TECs made of Bi_2Te_3 have high figure-of-merit (ZT) and are directly fabricated on the back of a silicon chip (Sahu et al., 2014; Chowdhury et al., 2009). Existing on-chip TEC devices are composed of ultrathin (5–10 μm) Bi_2Te_3 -based p-n thermocouples sandwiched between copper mini-headers and are covered with ceramic plates at the outmost surfaces to provide insulation (Chowdhury et al., 2009). The terms that contribute to the heat flow in a TEC unit are shown as follows:

$$Q_c = N(SIT_c - \frac{T_h - T_c}{R_t} - \frac{1}{2}I^2R_e), \quad (2.2)$$

$$Q_h = N(SIT_h - \frac{T_h - T_c}{R_t} + \frac{1}{2}I^2R_e), \quad (2.3)$$

where SIT_c is the Peltier effect term, $\frac{T_h - T_c}{R_t}$ is the thermal conduction term, and $\frac{1}{2}I^2R_e$ is the Joule heating term represents by the heat generated by passing a current through the TEC. T_c and T_h represent the temperatures of the cold side and the hot side of TEC, respectively. R_t is the thermal resistance and R_e is the electrical resistance of a TEC unit (Kaplan et al., 2019).

To model the thermal gradients caused by the Peltier effect, CTMs of TEC typically use a voltage-controlled current source to represent the heat entering and leaving the TEC grid cell (Chowdhury et al., 2009; Yazawa et al., 2012). The chip is divided into grid cells, and the default grid cells are used to represent the processing layer (Figure 2.2 (a)). A typical TEC grid cell is shown in Figure 2.2 (b). In this figure, the bottom surface of the TEC grid cell represents the cold side of the TEC unit, whereas the bottom surface of the ceramic grid cell corresponds to the hot side.

Several prior approaches have created similar CTMs for TECs and analyzed the cooling performance of TECs. For example, a numerical CTM of TEC that includes both thermal and electrical contact resistance predicts the cooling performance of TECs (Chowdhury et al., 2009). Both the temperature measurements from real prototypes and thermal simulation results using TEC CTM show a maximum temperature reduction of 15°C owing to the cooling ability of TECs. The thermal resistance of TECs plays a significant role in accurately determining the cooling performance of the TECs (Chowdhury et al., 2009). In addition, TECs' cooling performance is also highly correlated with the value of the bias current.

Recent work has introduced another CTM of TEC to estimate the impact of the bias current on the cooling performance of the TEC units (Kaplan et al., 2017). The TEC device and the chip layers they modeled in COMSOL are illustrated in Figure 2.3. The authors validated the accuracy of their proposed CTM of TEC by comparing the temperature results against the TEC model in COMSOL. The reported max, average, and root mean square

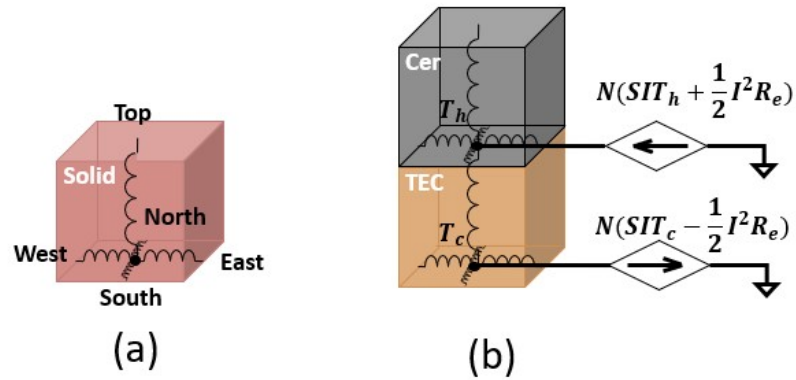


Figure 2-2: (a) Default grid cell and (b) TEC grid cell (Kaplan et al., 2017).

errors (RMSE) of their proposed CTM of TECs against COMSOL are 3.57°C , 2.07°C , and 2.25°C , respectively. Compared to the COMSOL model, the reported speedup of the CTM of TEC is $16900\times$, while the specific speedup number depends on the number of grids used in CTM simulations and the number of nodes used in COMSOL simulations.

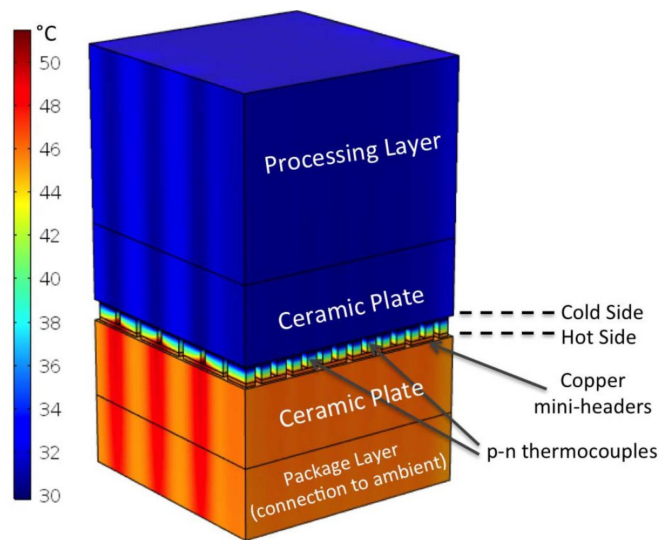


Figure 2-3: COMSOL model of the TEC device (Kaplan et al., 2017).

2.2.2 Liquid Cooling via Microchannels

Liquid cooling via microchannels is an attractive cooling solution that uses the liquid convection effect to remove heat from processors (Sridhar et al., 2014; Dang et al., 2010). It is advantageous as an inter-layer cooling method to solve the strong thermal-coupling issues for 3D-stacked architectures (Sridhar et al., 2014). Adding a liquid microchannel layer between processing layers efficiently solves the thermal problems due to the vertical layer stacking. A typical liquid cooling via microchannels 3D IC is shown in Figure 2-4. The coolant that is pumped from the inlet to the outlet absorbs heat due to the convective heat transfer (liquid convection). The chiller outside the system then cools down the heated coolant. There are two main contributors to the convective heat transfer of the coolant: 1) convective heat transfer from the walls of the channel to the liquid and 2) convective heat transfer in the direction of the liquid flow into and out of the current liquid cell.



Figure 2-4: Liquid cooling via microchannels 3D IC architecture (Sridhar et al., 2014).

To build a CTM for liquid cooling via microchannels, one approach (4-resistor model (4RM)) is to use the thermal resistors to represent the convective heat transfer from side-walls of the microchannels and use a voltage-controlled current source to represent the convective heat transfer along the liquid flow direction (Sridhar et al., 2013a). Figure 2-5 (b) shows the liquid grid cell, and the thermal resistors from sidewalls are defined as follows:

$$R_{top,bottom} = \frac{1}{h_{f,vertical} \cdot w \cdot l}, \quad (2.4)$$

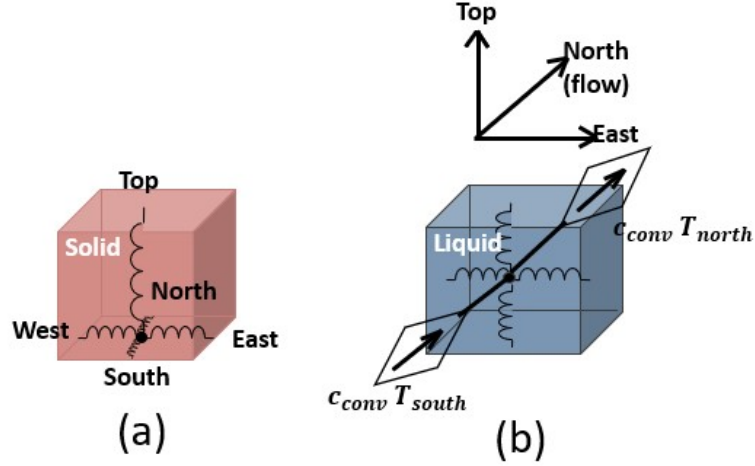


Figure 2.5: (a) Default grid cell and (b) Liquid grid cell (Sridhar et al., 2013a).

$$R_{east,west} = \frac{1}{h_{f,side} \cdot h \cdot l}. \quad (2.5)$$

In Equations (2.4) and (2.5), h , w , and l stand for the height, width, and length of the grid cell, respectively. $h_{f,vertical}$ and $h_{f,side}$ are HTCs for microchannel forced convection. $h_{f,vertical}$ and $h_{f,side}$ are obtained from empirical correlations or numerical presimulations. Prior work provides the following formulas to calculate $h_{f,vertical}$ and $h_{f,side}$ by assuming imposed axial heat flux and radial isothermal conditions:

$$h_{f,vertical} = h_{f,side} = \frac{k_{coolant} \cdot Nu}{d_h}, \quad (2.6)$$

where $k_{coolant}$ is the thermal conductivity of the coolant and d_h is the hydraulic diameters of the channel defined as $\frac{2hw}{h+w}$. Nu is the Nusselt number as a function of channel aspect ratio (Sridhar et al., 2013a). $h_{f,vertical}$, $h_{f,side}$, and Nu may differ under different system assumptions.

The value of the convective heat transfer along the microchannels (i.e., current) is represented as follows (Sridhar et al., 2013a):

$$J_{conv} = c_{conv}(T_{south} - T_{north}), \quad (2.7)$$

where T_{south} and T_{north} are the temperatures at the south and north surfaces of the grid cell. c_{conv} is the liquid convection coefficient, which is defined as $C_v u_{avg,y} \Delta A_y$. C_v is the specific heat capacity, $u_{avg,y}$ is the average liquid flow velocity and $\Delta A_y = wh$. The inlet temperature of the coolant is the boundary condition for this model.

Kaplan *et al.* has validated the 4RM-based CTM against a liquid cooling via microchannels COMSOL model (Kaplan et al., 2017). They ran steady-state simulations for a range of heat flux values of 12.5, 25, 50, and 100 W/cm^2 as well as for different flow velocities = 0.5, 1.0, 1.5, 2.0 m/s , and record the maximum temperature of the processing layer for the 4RM-based CTM and COMSOL. Compared to COMSOL simulations, the 4RM-based CTM has a maximum error of 2.8% with a speedup of 43300 \times .

To further speed up the simulation time of the liquid cooling via microchannels, a 2-resistor model-based (2RM-based) CTM has been designed in the previous work (Sridhar et al., 2010). In the 4RM-based CTM, the thermal cells' boundaries to conform to the solid-liquid interfaces need to be defined, which results in very fine discretization and consequently prohibitively large simulation time. The 2RM-based CTM solves this problem by homogenizing the whole microchannel layer into a single porous medium. In this way, thermal cells are no longer constrained by microchannel dimensions, and even multiple microchannels are covered by a single thermal cell in this homogeneous medium. Sridhar *et al.* (Sridhar et al., 2014) compared the simulation speed and accuracy of 4RM-based CTM and 2RM-based CTM of liquid cooling via microchannels on test 3D chips. The simulation results show that 2RM-based CTM achieves a 375 \times speedup compared to 4RM-based CTM, with an accuracy loss of 7%.

2.2.3 Hybrid Cooling

Hybrid cooling refers to incorporating two or more cooling solutions on the same platform. For example, a hybrid cooling system can be designed using liquid cooling via microchannels and TECs. Using liquid cooling via microchannels effectively removes the background

heat on large chips and 3D-stacked architectures. However, as the liquid flows through the microchannels, it gets hotter, and the ability to remove heat further decreases. TEC is favorable for handling high power densities in a small area. But the additional cooling power cost is substantially increased when cooling a large area. In this case, a hybrid cooling strategy that combines the strengths of liquid cooling via microchannels and TEC provides much better cooling efficiency. A group of works focuses on using TECs to target the hot spots and liquid cooling via microchannels to remove the background heat (Sahu et al., 2014; Yazawa et al., 2012; Hu et al., 2013). A typical hybrid cooling method using TEC and liquid cooling via microchannels is shown in Figure 2.6.

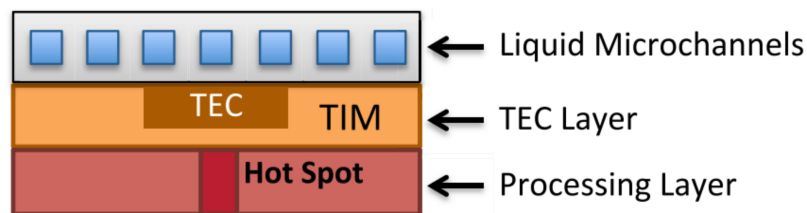


Figure 2.6: Front view of a hybrid cooling design (Kaplan et al., 2019).

There are several ways to create CTMs for hybrid cooling (liquid cooling via microchannels and TEC). One approach is to use a high HTC value to represent liquid cooling via microchannels in addition to a TEC CTM to model hybrid cooling (Yazawa et al., 2012). This method shows a $10\times$ cooling power reduction when compared to using only liquid cooling via microchannels (Yazawa et al., 2012). Another way builds a CTM of hybrid cooling using liquid cooling via microchannels CTM and TEC CTM (Kaplan et al., 2019). This method incorporates the CTMs for liquid cooling via microchannels and TEC from a previous work (Kaplan et al., 2017). Kaplan *et al.* validated the accuracy of this hybrid cooling CTM against COMSOL simulations. This hybrid cooling CTM achieves a $1607\times$ speedup with a max error of less than 5.7°C .

2.3 Cooling Optimization Methods

Researchers have developed optimization methods for emerging cooling methods. For liquid cooling via microchannels, Coskun *et al.* adjusted the flow rate at runtime to minimize the pumping power while satisfying the temperature limit (Coskun et al., 2010). Sabry *et al.* proposed a fuzzy controller to determine the liquid flow velocity at runtime (Sabry et al., 2011). GreenCool reduces the thermal gradients by modulating the channel width of liquid cooling via microchannels at design time (Sabry et al., 2013). Other approaches either optimize or co-optimize the microchannels' number, locations, dimension, and flow rate to achieve better pumping power for a given power profile (Shi and Srivastava, 2013; Sharma et al., 2015; Qian et al., 2013). For TECs, researchers focus on optimizing TEC device geometry and TEC current to maximize coefficient of performance (COP) (Sahu et al., 2014; Yazawa et al., 2012; Taylor and Solbrekken, 2008). For hybrid cooling, Kaplan *et al.* introduced an optimization method that is based on gradient descent to optimize the TEC current and liquid flow velocity to achieve the minimal cooling power while satisfying the temperature constraint (Kaplan et al., 2019). However, there is no comprehensive study to compare the cooling efficiency in cooling performance and cooling power for these cooling technologies. These potential solutions' cooling performance and cooling power vary significantly based on the cooling parameters (liquid flow velocity, TEC current, etc.). The cooling technologies and the cooling parameters also need to consider the chip architecture, chip size, floorplan, and the power profiles of the applications running on the given chip. Given the vast amount of tunable parameters and input designs, there is a need for a fast and accurate optimization method to facilitate cooling solutions design space exploration and optimization.

Chapter 3

Enabling Fast and Accurate Parallel Thermal Simulations with PACT

3.1 Introduction

Over the last few decades, chip temperature has become one of the essential criteria for designing high-performance, cost-effective, and reliable ICs. Increased power consumption and temperature not only degrade the performance of a chip but also generate larger sub-threshold leakage power and cause reliability challenges (Pedram and Nazarian, 2006). Therefore, thermal analysis is an essential procedure for designing any chip. The conventional thermal analysis relies on finite-element method (FEM) based multiphysics simulators (e.g., COMSOL and ANSYS (COM, 1998; Madenci and Guven, 2015)). However, such commercial simulators are computationally expensive and experience long solution times along with large memory requirements (Yuan et al., 2019a). These limitations make commercial simulators unsuitable for evaluating numerous design alternatives or running time scenarios. Therefore, having fast and accurate thermal analysis is crucial for chip design and thermal optimization.

To address the fast thermal analysis needs, researchers have developed tools using compact thermal modeling methods (Skadron et al., 2003; Sridhar et al., 2014; Kaplan et al., 2019; Sridhar et al., 2013b; Allec et al., 2008). We identify several challenges in existing compact thermal simulators (Skadron et al., 2003; Sridhar et al., 2014; Sridhar et al., 2013b; Allec et al., 2008). First, these thermal simulators target architecture-level thermal simulations only and do not perform standard-cell-level thermal simulations. The input power

information to the existing compact thermal simulators are typically through architecture-level or standard-cell-level performance and power simulation tools. For standard-cell inputs, high grid resolution thermal simulation is necessary for accurate temperature estimation. To demonstrate the necessity of standard-cell-level simulation, we select a high power design (Sparc) from OpenROAD (Ajayi et al., 2019) and carry out steady-state thermal simulations at various granularities. Figure 3-1 shows that architecture-level thermal simulation (e.g., 32×32 , 64×64 , and 128×128) cannot achieve the same accuracy as standard-cell-level simulation (e.g., 256×256 , 512×512 , and 1024×1024), with a maximum temperature inaccuracy of 3.28°C and a thermal gradient inaccuracy of 3.56°C . For thermally-aware circuit or policy design (e.g., thermally-aware dynamic voltage frequency scaling (Hanumaiah and Vrudhula, 2012)), applying architecture-level (low grid resolution) thermal simulations will result in accuracy loss and lead to suboptimal designs or even failures. Therefore, a compact thermal simulator should support fast and accurate standard-cell-level (high grid resolution) simulation for standard-cell design inputs.

Another challenge with existing compact simulators is that they cannot tackle large and complex problems (e.g., standard-cell-level design problems or multi-layered chips such as in monolithic 3D integration (Wong et al., 2007)) as the simulation time rises dramatically when problem size increases. One reason is that thermal simulators are typically designed to be sequential and cannot easily be parallelized. In addition, the solvers embedded in these simulators are often not efficient enough to perform high grid resolution thermal simulations. For example, HotSpot (Skadron et al., 2003) uses explicit adaptive 4^{th} order Runge-Kutta (adaptive RK4) to conduct transient thermal analysis, and this method suffers from numerical instability (Distefano, 1968). Such Forward Euler methods may converge slowly for transient simulation (e.g., on the order of days for a standard-cell-level chip model), depending on the granularity of the chip and the thickness of the chip layers.

A third challenge is that existing compact thermal simulators are either dedicated to a

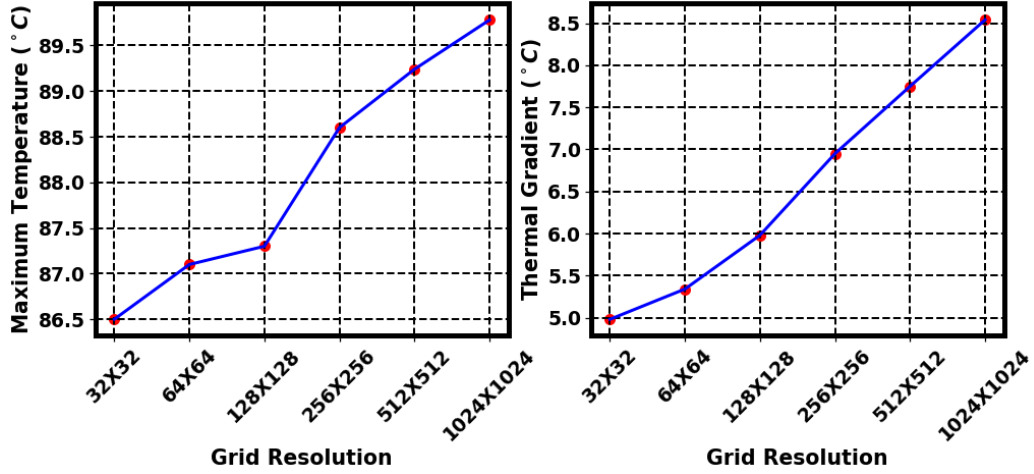


Figure 3-1: Temperature profiles for a standard-cell design at various grid resolutions. The maximum temperature and thermal gradient plots are expected to saturate with higher grid resolutions ($> 1024 \times 1024$).

specific cooling technology or it is difficult and time-consuming to extend them for emerging integration and cooling technologies, such as microchannel-based two-phase cooling, TECs, or two-phase VCs (Kaplan et al., 2019; Yuan et al., 2020; Yuan et al., 2019a). As a result, research that proposes models for such novel cooling methods frequently roll out customized software packages (e.g., (Sridhar et al., 2013b; Sridhar et al., 2014; Kaplan et al., 2019; Yuan et al., 2019b; Yuan et al., 2020)), resulting in a fragmented space of thermal modeling tools. We summarize the solvers, cooling methods, and inputs of popular compact thermal simulators in Table 3.1.

Table 3.1: Solvers, cooling methods, and inputs of PACT and of existing compact thermal simulators (e.g., HotSpot (Skadron et al., 2003), 3D-ICE (Sridhar et al., 2014), and ThermalScope (Allec et al., 2008)). BE stands for the Backward Euler solver, and TRAP is a hybrid solver of the Backward Euler and the Trapezoidal method. Full industrial design means that PACT takes real-world standard-cell designs as input, such as designs from OpenROAD.

Simulator	Steady-state	Transient	Cooling	Inputs
HotSpot	SuperLU	Explicit RK4	N/A	Block/architecture-level
3D-ICE	SuperLU	Backward Euler	Liquid cooling	Block/architecture-level
ThermalScope	Gauss Seidel	Trapezoidal	N/A	Block/architecture-level
PACT	KLU, KSparse AztecOO, Belos	TRAP, BE, Gear	Liquid cooling and easily extensible	Block/architecture-level and full industrial design

This chapter introduces a SPICE-based¹ PARallel Compact Thermal simulator (PACT) that enables speedy and accurate thermal analysis for processors. Recent advances in SPICE (Massobrio and Antognetti, 1993; Hutchinson et al., 2002; Zhou et al., 2008; Liu et al., 2009; Biolek and Biolek, 2014; Vladimirescu, 1994) solve many of the computational challenges associated with modeling electric circuits, and PACT leverages these improvements toward thermal modeling and analysis. Unlike existing thermal simulators that cannot easily solve standard-cell-level simulation problems, PACT supports parallel computing with various types of solvers to provide fast and accurate standard-cell-level to architecture-level² thermal analysis, regardless of problem size. We also provide an interface between OpenROAD and PACT, allowing PACT to take industrial standard-cell designs through this interface directly. Users are able to carry out either fine-granularity (each standard cell is covered by a grid cell) or coarse-granularity (a grid cell covers multiple standard cells) thermal simulations for standard-cell designs, owing to the fast parallel thermal simulation and the OpenROAD interface. Experimental results confirm that, compared to HotSpot, PACT reduces the steady-state simulation time of a monolithic 3D IC from more than 3 hours to less than 16 minutes. PACT also speeds up the simulation time of a 256-core 2D IC from more than three days to less than 19 minutes compared to HotSpot. In addition, users are able to easily extend PACT to model various emerging integration and cooling technologies by adding dependent/independent sources, resistors, and capacitors.

The rest of the chapter starts with a discussion on existing thermal simulators (Section 3.2). Section 3.3 elaborates on the simulation flow, thermal netlist generation, and compact modeling of various emerging technologies in PACT. We demonstrate the impact of PACT by simulating realistic 2D ICs, monolithic 3D ICs, die-stacked 3D ICs with liquid cooling,

¹SPICE stands for Simulation Program with Integrated Circuit Emphasis.

²Standard-cell-level thermal simulation refers to a high grid resolution simulation (i.e. a grid node occupies one or more standard cells). Architecture-level thermal simulation refers to a relatively low grid resolution simulation (i.e., several grid nodes often occupy a hardware block).

and chips with photonic network-on-chip (PNoC) in Section 3.4. Section 3.4 also shows the validation and speed analysis of PACT using full industrial designs from OpenROAD. To further demonstrate the extensibility and applicability of PACT, we model lab-grown diamond heat spreaders in PACT and evaluate the cooling performance of lab-grown diamond heat spreaders using PACT in Section 3.5.

3.2 Background of Existing Compact Thermal Simulators

To maintain safe chip temperatures, researchers have proposed various solutions including design-time thermal management techniques (Eris et al., 2018; Pedram and Nazarian, 2006) and runtime policies such as dynamic voltage frequency scaling (Bao et al., 2009; Lee et al., 2010), task scheduling (Tang et al., 2008; Zhou et al., 2010; Coskun et al., 2007), and thread migration (Sheikh et al., 2012; Beigi and Memik, 2016). Several emerging cooling technologies such as liquid cooling via microchannels (Sridhar et al., 2014; Dang et al., 2010; Coskun et al., 2010; Coskun et al., 2009), TECs (Kaplan et al., 2019; Chowdhury et al., 2009; Paterna and Reda, 2013), two-phase cooling (Yuan et al., 2019a; Sridhar et al., 2013b), and hybrid cooling (such as a hybrid design of liquid cooling via microchannels and TECs (Kaplan et al., 2019; Yazawa et al., 2012)) have also been proposed by the researchers to mitigate the high chip temperatures. These solutions often rely on fast and accurate thermal analysis to enable design exploration and optimization of their design parameters and runtime knobs.

However, existing FEM-based thermal simulators experience high computational complexity and memory usage when modeling large and complex chips or conducting standard-cell-level analysis. For example, simulating the transient behavior of a realistic chip with a high grid resolution can take from several hours to days and easily requires beyond tens of GBs of memory (Kaplan et al., 2019). Compact thermal modeling methodology is a popular solution to solve the long simulation time problem (Huang et al., 2004). Several

compact thermal simulators have been designed to model the full-chip temperature behavior, and emerging cooling solutions (Sridhar et al., 2014; Skadron et al., 2003; Sridhar et al., 2013b; Pedram and Nazarian, 2006). Skadron *et al.* introduced HotSpot, an architectural thermal simulator that utilizes the CTM method to conduct the thermal analysis for processors (Skadron et al., 2003). The latest version of HotSpot uses a sparse matrix direct solver (SuperLU (Skadron et al., 2003)) to obtain steady-state temperature profiles and an adaptive RK4 method to compute the transient thermal behavior (Li, 2005). However, Forward Euler methods such as explicit adaptive RK4 suffers from numerical instability issues (Distefano, 1968). As the number of grids increases or layer thickness decreases to the nanometer level, adaptive RK4 continuously decreases the minimum simulation step size, which slows down the simulation speed significantly. For instance, transient simulation of thin layers (such as in a monolithic 3D system) with a high grid resolution takes more than a day in HotSpot. There exist other compact thermal simulators that focus on modeling specific types of emerging cooling technologies (Sridhar et al., 2014; Sridhar et al., 2013b). However, a common issue in these compact thermal simulators (Skadron et al., 2003; Sridhar et al., 2014; Sridhar et al., 2013b; Yuan et al., 2020; Allec et al., 2008; Pedram and Nazarian, 2006) is that these simulators only perform sequential thermal simulations and are hard to modify to support parallel thermal simulations. As the problem size increases, the simulation time increases significantly, especially for transient thermal analysis of standard-cell design.

To speed up standard-cell-level thermal simulations, Green's function is a promising solution to conduct efficient simulation for high grid resolution thermal simulations (Varshney et al., 2019). However, if the geometry of the chip or boundary condition changes, Green's function needs to be recomputed or resimulated (Ziabari et al., 2014). Other works have either introduced fast thermal simulation algorithms (Tan et al., 2008; Liu et al., 2006; Yu et al., 2013) or used hardware platforms (CPU-GPU platforms) (Liu et al., 2014) to ac-

celerate the thermal simulations. However, these works focus solely on architecture-level thermal simulations, and their methods have not been demonstrated to be applicable to emerging integration and cooling technologies.

Another potential solution is to use the SPICE simulator to build the thermal network and carry out thermal simulations (Chiang et al., 2001; Wang and Chen, 2004). However, these works model the thermal effects and reliability of interconnects and do not focus on using the SPICE simulator for full system thermal analysis. Moreover, these works are not open-sourced and cannot be extended to support emerging integration and cooling technologies.

PACT aims to address the fragmentation in the thermal modeling tool space and provides a single tool to conduct efficient thermal evaluation from standard-cell-level to architecture-level for various chip integration and cooling technologies. A key distinguishing feature of PACT is its inherent parallelism, which speeds up the simulation time for standard-cell-level thermal simulations while maintaining high accuracy. As PACT is a SPICE-based simulator, it can be easily extended to support and evaluate chip designs with emerging cooling technologies. Moreover, PACT allows users to decide whether they want a faster convergence speed or a more accurate thermal profile by supporting various steady-state and transient solvers.

3.3 Proposed SPICE-Based Thermal Simulator

PACT is a SPICE-based standard-cell-level to the architecture-level parallel compact thermal simulator. To explain how PACT works, we first go over the simulation flow of PACT and then discuss the core of PACT, which is a thermal netlist. A thermal simulator should support the modeling of various emerging integration and cooling technologies and be compatible with architecture-level performance and power simulators. Because of the simple structure of PACT's thermal netlist and the available SPICE component library, it's easy

to extend PACT to support various emerging integration and technologies. We illustrate the extensibility of PACT by modifying the thermal netlist to support the modeling of conventional heat sinks, 3D ICs (die-stacked 3D and monolithic 3D), and liquid cooling via microchannels. We show the compatibility of PACT with popular architecture-level performance and power simulators by creating a 2.5D PNoC simulation framework. Since PACT acquires full industrial designs from OpenROAD, we also elaborate on the interface between PACT and OpenROAD. The SPICE engine also provides PACT with various steady-state and transient solvers, which benefits PACT in simulation speed. We discuss the available solvers in PACT and demonstrate why the solver selection is essential for evaluating the thermal behavior of processors.

3.3.1 PACT Simulation Flow

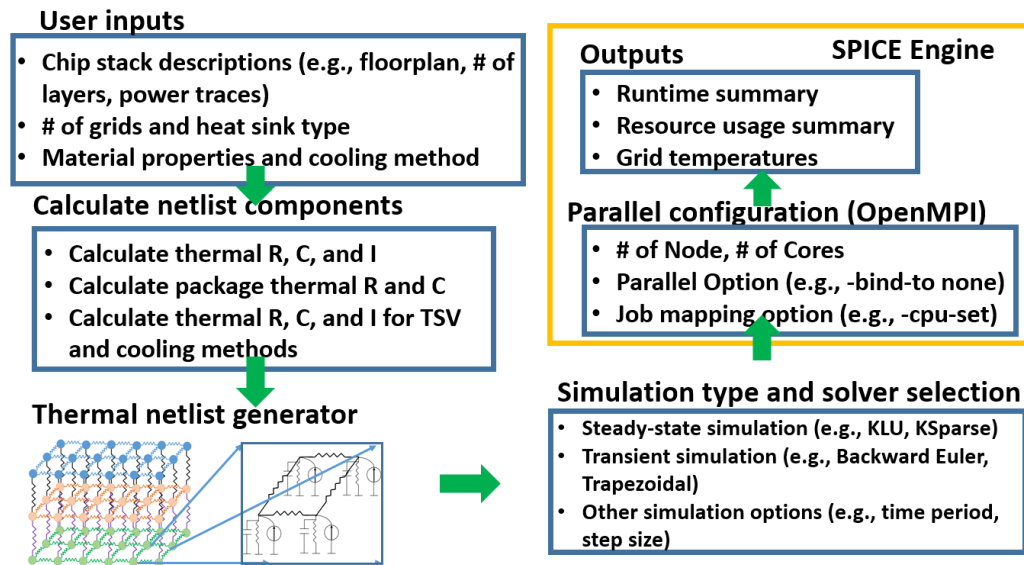


Figure 3-2: PACT simulation flow.

Figure 3-2 shows the simulation flow of PACT. The simulation steps are as follows: (i) users pass information about the chip stack (such as number of layers, floorplans, or power traces), material properties (including thermal resistivity and specific heat), problem size

(number of grids), heat sink type, and cooling method to PACT, (ii) PACT calculates the lateral and vertical thermal resistance as well as thermal capacitance for each grid. For the layers that consume power, PACT also computes the power consumption of each grid. For emerging cooling layers, PACT determines the corresponding cooling parameters based on the cooling design and the input. In the meantime, PACT builds the heat sink requested by users, (iii) PACT calculates and assigns R, C, and power values to the corresponding resistors, capacitors, and independent current sources and uses these circuit components to build a thermal netlist, (iv) PACT allows the users to specify the type of simulation (steady-state or transient) as well as the solvers, (v) users can also enable parallel thermal simulations by specifying the number of cores and nodes via OpenMPI (OpenMPI, 2014). PACT utilizes hypergraph partitioning via the Zoltan library (Boman et al., 2012) and subdivides and distributes the thermal netlist to the available processors. The Zoltan library provides an effective load balancer and seeks to minimize the message passing overhead among processors (Boman et al., 2012), and (vi) PACT solves the RC thermal netlist using the SPICE engine of PACT and outputs the grid temperatures along with the simulation time and resource usage summary.

3.3.2 Thermal Netlist and SPICE Circuit Components

Similar to other compact simulators, PACT also calculates the thermal resistor, capacitor, and heat flow values using Equations (3.1-3.4). R_x , R_y , and R_z are the thermal resistance along the x, y, and z directions, respectively. C is the thermal capacitance of the grid node. R_λ and c_p are the thermal resistivity (mK/W) and specific heat capacity (J/m^3K) of the material, respectively. w , l , and t are the grid node's width, height, and thickness, respectively. To calculate the heat flow values, PACT uniformly divides the power profile of the chip into grids based on the predefined grid resolution. Then it creates a power matrix (W) to assign power to each grid to represent the heat flow. Since PACT is a SPICE-based simulator, PACT directly uses the circuit components available in the SPICE library to construct the

thermal netlist. To extend PACT to support emerging integration and cooling technologies, users need to add additional libraries or utility functions and modify the thermal netlist. It is straightforward to build and modify the thermal netlist by adding and deleting the circuit components or changing the connection of the thermal grids in PACT. Figure 3-3 shows the component symbol, component name in SPICE, and equivalent terminology in PACT. For steady-state simulation, PACT only uses resistors, voltage sources, and current sources to build the thermal netlist and conducts operating point analysis (.OP in SPICE) to solve the thermal netlist. For transient simulation, PACT also calculates the thermal capacitance of the corresponding grid node. To construct the thermal netlist for emerging cooling technologies, users need to add the circuit components from the SPICE library to model the unique cooling behavior of that cooling method. For instance, additional voltage-controlled current sources need to be added to the thermal netlist to model the heat conduction along the microchannel of the liquid cooling via the microchannel method. For modeling the additional vertical heat conduction provided by the TEC units, voltage-controlled current sources need to be included between the normal grid node and the TEC grid node. For transient thermal simulations with real power traces, PACT uses the piece-wise linear (PWL) function component and stores the power traces for each grid node in the corresponding PWL component to conduct transient analysis (.TRAN in SPICE).

$$R_x = \frac{R_\lambda \cdot w}{l \cdot t} \quad (3.1)$$

$$R_y = \frac{R_\lambda \cdot l}{w \cdot t} \quad (3.2)$$

$$R_z = \frac{R_\lambda \cdot t}{w \cdot l} \quad (3.3)$$

$$C = c_p \cdot w \cdot l \cdot t \quad (3.4)$$





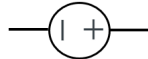

Symbol	Component name	Equivalent terminology in PACT
	Resistor	Thermal Resistor
	Capacitor	Thermal Capacitor
	Current source	Heat flow (power)
	Voltage-controlled current source	Liquid convection in microchannel grid
	Voltage source	Assign initial temperature and ambient temperature
	PWL current source	Enable transient thermal simulation with step response or real power traces

Figure 3.3: SPICE circuit component usage in PACT.

3.3.3 Extensibility of PACT

As we discussed in Section 3.3.2, building the thermal netlist in PACT using SPICE simplifies the construction and modification of the netlist, which enhances the extensibility of PACT. This section gives several examples to demonstrate how we extend PACT to support new technologies, such as different kinds of heat sinks, 3D ICs, and liquid cooling via microchannels.

Heat Sink

Many different kinds of heat sinks can be modeled using PACT. In the current version of PACT, we support a medium-cost heat sink that is adopted from a recent work (Skadron et al., 2003) and a fixed-air convection HTC heat sink.

The medium-cost heat sink represents a combination of the heat spreader, heat sink, and fan and is used to mimic the realistic heat sinks in processors and servers (Skadron et al., 2003). By modifying the size, material, and air convection HTC of this medium-cost heat sink, it is able to model heat sinks for mobile chips. To build this type of heat sink, we add two additional layers on top of the chip to represent the heat spreader and heat sink.

In addition to the normal heat spreader and heat sink grid nodes that connect to the chip nodes, we only need to add 12 additional heat sink and heat spreader nodes on top of the original thermal netlist and populate the resistance and capacitance as the thermal resistors and capacitors attached to these nodes (Skadron et al., 2003). Similar to HotSpot, four of the additional nodes are assigned to the periphery of the heat spreader, while the remaining eight nodes (four inner nodes and four outer nodes) are assigned to the periphery of the heat sink. The thermal resistance and capacitance of the additional nodes of the heat spreader and heat sink are calculated based on the size, thickness, air convection resistivity, thermal conductivity, and specific heat of the heat sink and heat spreader. We show the high-level simulation flow for enabling this medium-cost heat sink in Figure 3.4. The heat spreader and heat sink specifications have to be specified through the PACT front-end. The medium-cost heat sink utility functions are added to the PACT's back-end to calculate the additional thermal resistance and thermal capacitance introduced by this medium-cost heat sink.

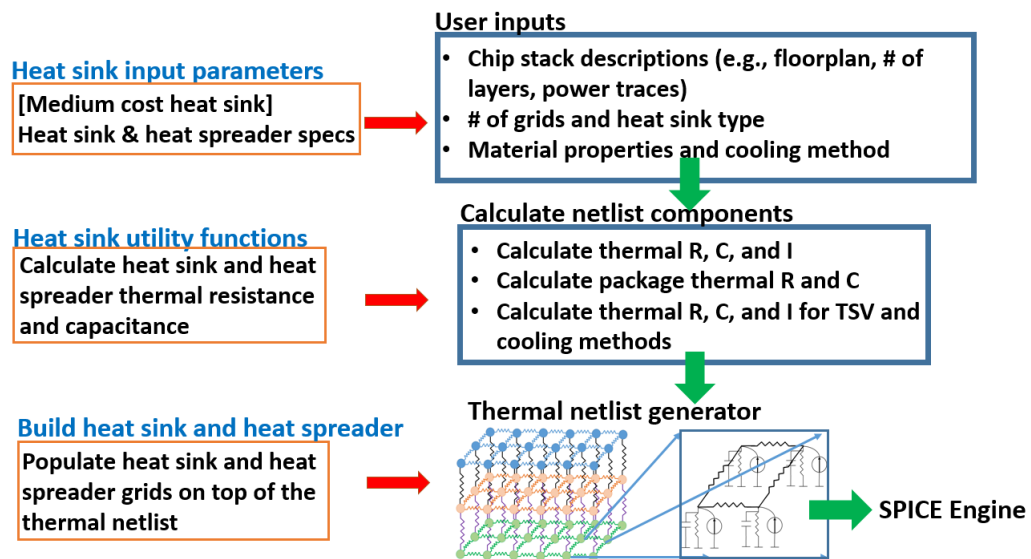


Figure 3-4: The high-level simulation flow with the medium-cost heat sink.

Since simulations of some emerging cooling technologies (e.g., liquid cooling via microchannels and two-phase cooling) require a fixed-air convection HTC heat sink or even

no heat sink on top of the chip, it is not realistic to use the medium-cost heat sink (Sridhar et al., 2014; Sridhar et al., 2013b; Kaplan et al., 2019; Yuan et al., 2019a; Yuan et al., 2020). Due to this reason, PACT also provides a fixed-air convection HTC heat sink where the vertical thermal resistance of the heat sink is the air convection HTC. PACT replaces the heat spreader and heat sink with a dummy layer and connects it to the ground with a vertical thermal resistance calculated using the fixed-air convection HTC (Kaplan et al., 2019).

Modeling Layers with Heterogeneous Materials

Unlike the typical 2D chips, 3D ICs need additional through-silicon vias (TSVs) or monolithic inter-tier vias (MIVs) to enable inter-layer communication and power delivery to the tiers. Therefore, thermal simulators should be able to model heterogeneous materials within one layer. Similar to the 3D extension in HotSpot, PACT is also capable of modeling layers with heterogeneous materials (Meng et al., 2012; Skadron et al., 2003). For a layer with homogeneous material, PACT assigns the same vertical and horizontal thermal resistance and thermal capacitance to each resistor and capacitor component inside of this layer, respectively. For heterogeneous material nodes in a layer, PACT directly modifies the thermal resistance and thermal capacitance of the corresponding heterogeneous nodes and creates thermal resistance and capacitance matrices to generate the thermal netlist.

Liquid Cooling via Microchannels in PACT

PACT offers standardized interfaces for easy integration of various compact models of emerging cooling techniques. These models are imported as Python modules in PACT. A sample liquid cooling via microchannels chip stack is shown in Figure 3-5. Both the bottom and top layers are silicon dies in this chip stack. The liquid microchannel layer is placed in the middle to mitigate the strong vertical thermal coupling issue for 3D stacking architectures. We adopt the liquid cooling via microchannels compact modeling methods from

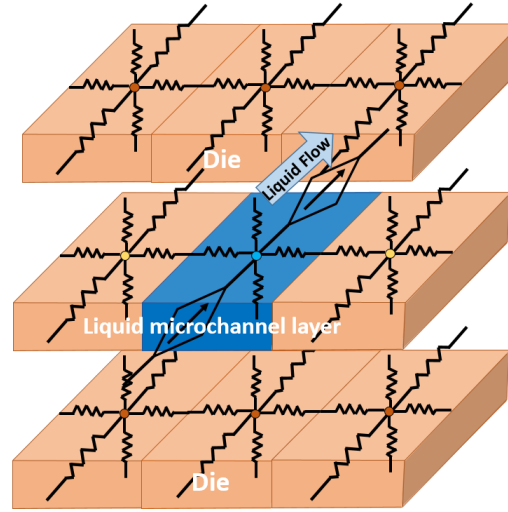


Figure 3-5: A small section of a liquid-cooled chip stack.

recent work (Sridhar et al., 2014; Kaplan et al., 2019). Unlike a typical compact thermal grid that consists of six thermal resistors for each node to represent the heat conduction from north, south, east, west, top, and bottom directions, a liquid microchannel grid node has only four thermal resistors, which represent the heat conduction between the coolant and the microchannel walls. In PACT, the thermal resistance of a liquid microchannel grid node is calculated based on the vertical and side wall HTCs (i.e., $h_{f,vertical}$ and $h_{f,side}$, respectively) as shown in Equation (3.5) (Sridhar et al., 2014; Kaplan et al., 2019). Nu , $k_{coolant}$, and d_h are Nusselt number, the thermal conductivity of the coolant, and the hydraulic diameter of the channel, respectively. The additional voltage-controlled current source models the liquid convection effect inside the microchannel. Equation (3.6) shows the relationship between the current, J_{conv} , and liquid convection coefficient, c_{conv} . PACT uses c_{conv} as the transconductance of the voltage-controlled current source and $\{T_{in}, T_{out}\}$ as the voltage controlling nodes. T_{in} is the average voltage of the previous microchannel node and current microchannel node, and T_{out} is the average voltage of the current microchannel

node and the next microchannel node.

$$h_{f,vertical} = h_{f,side} = \frac{k_{coolant} \cdot Nu}{d_h} \quad (3.5)$$

$$J_{conv} = c_{conv}(T_{in} - T_{out}) \quad (3.6)$$

We show how to implement liquid cooling via microchannels grid nodes in Figure 3-6. All the liquid cooling input parameters (e.g., liquid flow velocity, thermal resistivity, specific heat capacity, etc.) must be specified as user inputs. Users have to create a Python module (Liquid.py) to define the vertical and side walls' thermal resistance and the liquid convection coefficient. The thermal resistance and liquid convection coefficient are then used to create the thermal netlist, where vertical and side walls' thermal resistance are modeled as electric resistors. The liquid convection coefficient is used to model the voltage-controlled current source. In addition, users also need to define the liquid grid type (e.g., the virtual temperature node is placed at the center of the grid node and not at the bottom of the grid node). To obtain the thermal resistance and liquid convection coefficient, PACT calls the correct liquid cooling library (Liquid.py). In this way, the modeling methodology of liquid cooling via microchannels grid node in PACT can be applied to model the grid nodes of microchannel-based two-phase cooling and TEC units by creating their respective compact libraries (i.e., Python modules).

As we see in Figures 3-4 and 3-6, to support emerging integration and cooling technologies in PACT, users only need to add their additional cooling method libraries and the existing circuit components from the SPICE simulator library to create a new thermal netlist based on the current design. To model a new cooling technology in PACT, users need first to create the CTM of the cooling method and then map the CTM components to circuit components. For example, voltage-controlled current sources is used in the CTM to model latent heat transfer. The thermal netlist code is well-structured and requires minimal changes to support emerging technologies. It is also possible for users to extend the SPICE

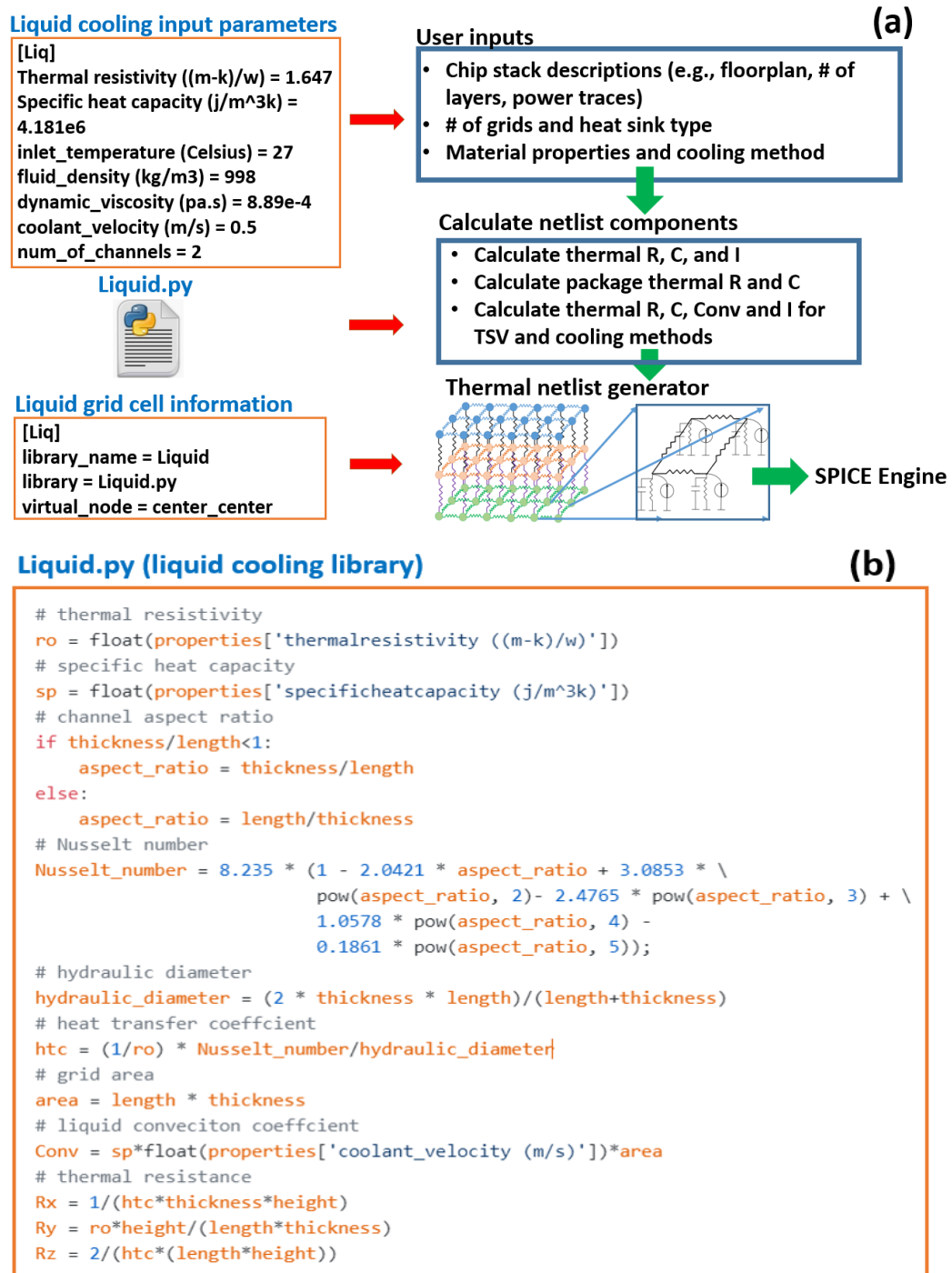


Figure 3-6: (a) The high-level simulation flow with liquid cooling via microchannels and (b) the additional liquid cooling library file for implementing a CTM for liquid cooling via microchannels.

library with a self-defined circuit component to support other emerging cooling technologies. Depending on the SPICE engine integrated with PACT, users are able to either modify the .lib file or create a new component written in Verilog-A (Hutchinson et al., 2002).

3.3.4 Compatibility of PACT

To show the compatibility with architecture-level performance/power simulators, we integrate PACT with Sniper (Carlson et al., 2011) and McPAT (Li et al., 2009), and create a PNoC cross-layer simulation framework to model the system performance and PNoC power under different activated laser wavelengths and Microring Resonators (MRRs) lock status. The PNoC simulation framework is adopted from recent work (Narayan et al., 2019) and shown in Figure 3-7. The original simulation framework uses HotSpot as the thermal engine; we replace HotSpot with PACT to evaluate the temperature of the PNoC. POPSTAR is a 2.5D manycore system with a PNoC architecture, and it has been modeled in Sniper. McPAT is used to compute the core and cache power consumption, while PACT determines the temperatures of all the Microring Resonator Groups (MRRGs). We show the temperature validation results against the original PNoC simulation framework in Section 3.4.2.

3.3.5 OpenROAD Interface

OpenROAD is a top-level register-transfer level (RTL) to graphic data stream (GDS) flow, which generates post-routing design exchange format (DEF) files of a given circuit (Ajayi et al., 2019). We use OpenROAD to get spatial power information at the standard-cell-level. Figure 3-8 shows the flow diagram of using OpenROAD (Ajayi et al., 2019) to generate an industrial input for PACT. Using the DEF files, we generate the power values for every single instance in the design using OpenSTA³ (Ajayi et al., 2019), which is a static timing analysis tool from parallax software that recently went open-source and sup-

³OpenSTA: <https://github.com/The-OpenROAD-Project/>.

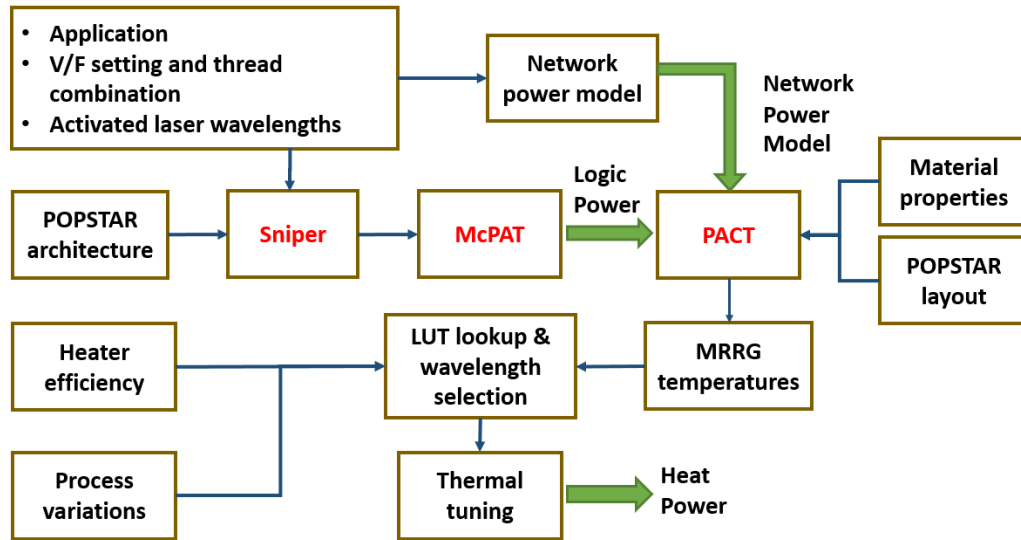


Figure 3-7: PNoC simulation framework.

ports gate-level simulation. OpenSTA is included in the OpenROAD project, and the power reporting mechanism is similar to Synopsys PrimeTime (Ajayi et al., 2019). The accuracy of OpenSTA was verified against industrial tools by its developer. Using the DEF files, every single instance in the circuit is passed to OpenSTA (Ajayi et al., 2019) while providing the standard-cell library files (lib and lef) and the operating frequency. Finally, based on the die dimensions and the number of grid nodes the user desires, we compute the power per grid node by identifying the gates that belong to each single grid node based on their coordinates, and then compute the grid node power by summing the power values of all the gates that belong to it. Further details on power map generations using OpenROAD for standard-cell designs and usage of the interface can be found in the previous work (Ajayi et al., 2019) and PACT’s GitHub repository. Since OpenROAD is an open-source project, users are able to directly utilize this interface to create standard-cell-level power maps and perform thermal simulations. For other commercial EDA design flows (e.g., Cadence and Synopsys), PACT can also be used as the backend thermal simulator with the same interface.

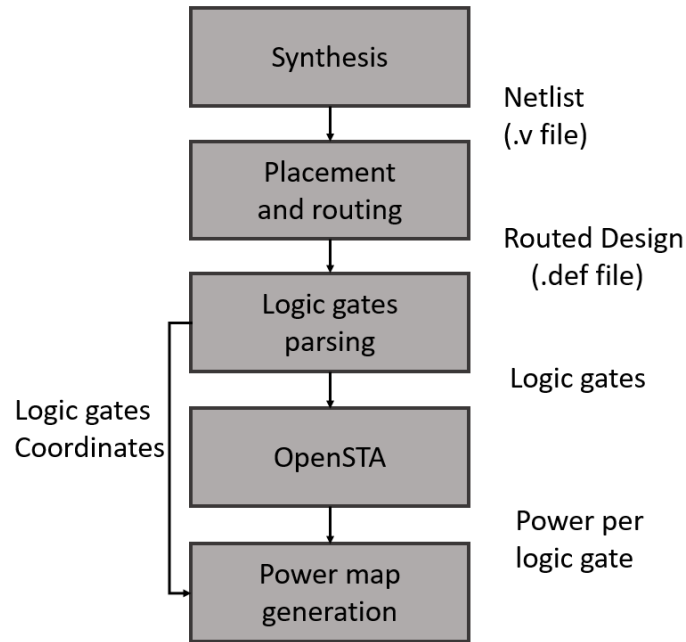


Figure 3-8: The flow diagram of OpenROAD.

3.3.6 PACT Solver

The steady-state and transient solvers in existing compact thermal simulators such as HotSpot are not comprehensive enough to model and simulate different chip architectures. For instance, we model and simulate a two-layer chip stack's transient behavior with a grid resolution equal to 50×50 . The sampling interval is set to $3.33 \mu\text{s}$, and the end time is set to $666 \mu\text{s}$ (total of 200 steps). We sweep the layer thickness from $100 \mu\text{m}$ to 100nm and show the simulation time results in Figure 3.9. The simulation time increases by more than $2880 \times$ when the chip thickness decreases from $100 \mu\text{m}$ to 100nm . As we discussed in Section 3.2, the reason behind this simulation time burst is the numerical instability issue of RK4. Forward Euler methods provide high accuracy and simulation speed for non-stiff equations, but for stiff equations (such as modeling thin layers in HotSpot), the simulation time is extremely long (Distefano, 1968).

Unlike other compact thermal simulators, PACT supports various steady-state solvers (e.g., KLU, SuperLU, and AztecOO) and transient solvers (such as Trapezoidal, Backward

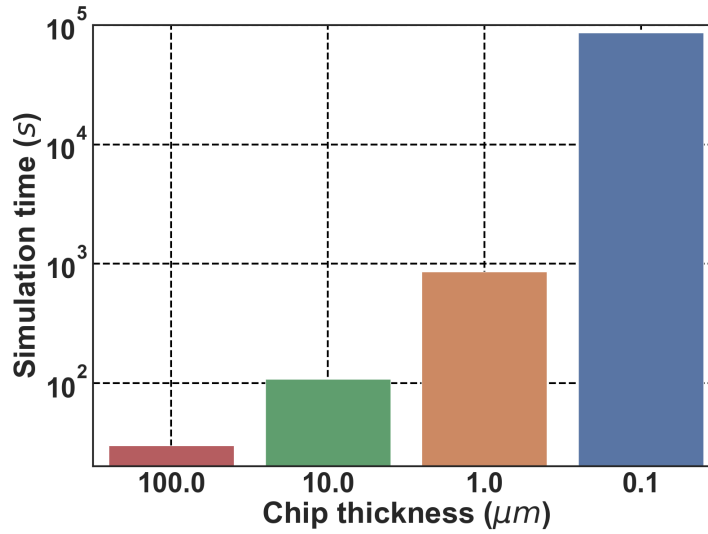


Figure 3-9: Transient simulation time of a two-layer chip stack.

Table 3.2: Information about available solvers in PACT.

Solver	Type	Mode	Simulation type
KLU	direct	serial and parallel	steady-state
KSparse	direct	serial and parallel	steady-state
SuperLU	direct	serial and parallel	steady-state
AztecOO	iterative	parallel	steady-state
Belos	iterative	parallel	steady-state
Backward-Euler	implicit	serial and parallel	transient
Trap	trapezoidal	serial and parallel	transient
Gear	linear Multistep	serial and parallel	transient

Euler, and Gear) (Hutchinson et al., 2002). We list the information of available solvers in PACT in Table 3.2. KLU, KSparse, and SuperLU are serial solvers. However, suppose the users use parallel settings with these serial solvers. In that case, the thermal netlists are evaluated and assembled in parallel, which is significantly more efficient compared to only using a single processor to evaluate and assemble the netlist (Hutchinson et al., 2002). These solvers comprehensively apply PACT to solve thermal netlists from various chip architecture designs at different simulation granularities.

There is a speed trade-off among different solvers and simulation modes (parallel or serial) in PACT (Hutchinson et al., 2002; Higham, 2002). The simulation mode, number

of cores, problem size, and solver type determine the thermal simulation's overall accuracy and running time. For example, TRAP (Heroux et al., 2005) is a hybrid solver of the Backward Euler and the Trapezoidal method, and for the chip stack used in Figure 3-9 with 100 nm thickness, the simulation time of PACT using TRAP solver takes less than 29 seconds. As another example, KLU is a direct solver that is used for single-core steady-state simulation, while AztecOO is an iterative steady-state solver, and it outperforms KLU for multicore simulations. For standard-cell-level thermal simulations, AztecOO is preferred since it enables parallel thermal simulations. For architecture-level thermal simulations, KLU outperforms AztecOO mainly because the problem size is small, and the additional communication cost of multicore processing is more significant than single-core simulations. Another example is that for certain thermal netlists, using an iterative solver (e.g., AztecOO) to conduct steady-state simulations may result in a convergence error in PACT (Hutchinson et al., 2002). In this case, PACT notifies the users of the convergence error and suggests the users use a direct solver (e.g., KLU) instead.

Since the SPICE engine is designed from the ground up to be distributed-memory parallel, all of these solvers support parallel simulation via OpenMPI (Hutchinson et al., 2002). However, for existing compact thermal simulators such as HotSpot, 3D-ICE, and ThermalScope, the designers have not considered the standard-cell-level simulation problem and how to utilize the benefits of multicore and multiprocessor simulations with a server cluster to tackle this problem. Therefore, PACT can be parallelized to achieve notable speedup compared to running thermal simulations via existing compact thermal simulators. We discuss the comparison results between PACT and HotSpot for running standard-cell-level thermal simulations for complex chip designs in Section 3.4.4.

3.4 Experimental Results

This section demonstrates the advantages of running parallel thermal simulations with PACT. We first run steady-state and transient simulations with large and complex realistic 2D and monolithic 3D multiprocessor system on a chip (MPSoCs) and compare the simulation speed to HotSpot. Then, we show thermal evaluation results against a PNoC simulation framework with HotSpot to show the compatibility of PACT with respect to popular architectural performance and power simulators. In addition, we validate the accuracy of the liquid cooling via microchannels CTM integrated with PACT and compare the simulation time to 3D-ICE. Finally, to validate the accuracy of PACT, we compare the standard-cell-level steady-state and transient simulation thermal profiles to those obtained using a FEM-based simulator, COMSOL. We also compare the simulation results of HotSpot against COMSOL to show that PACT has the same accuracy as a popular state-of-the-art compact thermal simulator. Since PACT is a parallel thermal simulator, we also compare the simulation speed of PACT to HotSpot using parallel simulation mode. In addition, we also compare the accuracy and running time of PACT to Manchester Thermal Analyzer (MTA) (Ladenheim et al., 2016).

PACT is written in Python, and we use Xyce 6.12 with OpenMPI 3.1.4 as our SPICE engine for all the experiments (Hutchinson et al., 2002; OpenMPI, 2014). We perform our simulations on the Massachusetts Green High-Performance Computing Center (MGHPCC). MGHPCC consists of hundreds of compute nodes, and each node has at least 128 GB of memory and two sockets. We run on nodes that contain two Intel Xeon E5-2680 v4 CPUs, each with 14 2-way hyper-threaded cores. We use at most four nodes (112 cores) in each of our experiments.

3.4.1 Speed Analysis with Complex 2D and Monolithic 3D ICs

We use PACT and HotSpot to simulate two large and complex chips to demonstrate the applicability and advantages of PACT. We simulate a 256-core processor (2D IC) inspired by the Intel SCC scaled to 22 *nm* (Eris et al., 2018), and a 33-layer monolithic 3D IC adopted from recent work (Shukla et al., 2019). For the 256-core SCC-based chip, the core architecture is based on the IA-32 core (Howard et al., 2011). We obtain power profiles of a simulated SCC-based chip from recent work (Eris et al., 2018). For our simulations, we select the power profile that results in the highest thermal gradient and chip temperature of the SCC-based chip to extract the most interesting thermal profile of the chip. The selected power profile has a hot spot power density of 216.6 W/cm^2 . We summarize the experimental setup in Table 3.3. We use the same medium-cost heat sink in both HotSpot and PACT and report the simulation speed results in Table 3.4. We observe in these results that PACT is favorable for solving standard-cell-level problems due to its ability to conduct parallel thermal simulations. For the monolithic 3D chip, when the number of grids = 200×200 , PACT takes less than 19 minutes to finish both steady-state and transient simulations. On the other hand, it takes HotSpot 3 hours to finish the steady-state simulation and more than three days for transient. Another advantage of using PACT is that users are able to select different types of solvers. We observe that the HotSpot numerical instability problem in transient simulations is exaggerated for the thin layers in Mono3D (thickness $< 1 \mu\text{m}$), which makes HotSpot and Forward Euler solver unsuitable for simulating thin layer chips. For standard-cell-level thermal simulations such as Intel SCC-based chip, when compared to HotSpot, PACT achieves a maximum speedup of $1.9 \times$ and $232 \times$ for steady-state and transient simulations, respectively. The reason behind this speedup is that as the problem size increases at a finer granularity, the direct steady-state solver (SuperLU) in HotSpot significantly slows down due to its significant memory usage. However, PACT automatically uses AztecOO, an iterative solver with parallel mode for finer grid resolutions to speed up

the thermal simulations. For standard-cell-level thermal simulations with large and complex chips, PACT outperforms HotSpot in terms of steady-state and transient simulation times. Most importantly, since most of the runtime thermal management policies are based on the transient behavior of the chip thermal profile, having a fast transient thermal simulation is particularly important.

Table 3.3: Experimental setup of monolithic 3D chip and the SCC-based chip simulations.

Chip	Simulator	# of Grids (row)	Step Size (μs)	# of Steps	# of Cores	Solver
Mono3D	HotSpot 6.0	50,100,200	3.33	5	N/A	SuperLU
	PACT	50,100,200	3.33	5	8	AztecOO
SCC	HotSpot 6.0	256,512,1024	3.33	100	N/A	RK4
	PACT	256,512,1024	3.33	100	8	Trap

Table 3.4: Simulation results of the monolithic 3D chip and the SCC-based chip.

Simulations	Chip	# of Grids	HotSpot running time	PACT running time
Steady-state	Mono3D	50 \times 50	1min5s	59s
		100 \times 100	13min11s	5min54s
		200 \times 200	3hrs2min	15min53s
	SCC	256 \times 256	24.7s	23s
		512 \times 512	3min19s	2min15s
		1024 \times 1024	26min32s	13min55s
Transient	Mono3D	50 \times 50	>3 day	2min23s
		100 \times 100	>3 day	6min21s
		200 \times 200	>3 day	18min48s
	SCC	256 \times 256	21min45s	1min1s
		512 \times 512	5hr38s	5min20s
		1024 \times 1024	>3 day	18min33s

3.4.2 Full System Simulation of 2.5D Systems with PNoC

We obtain the power profiles from running the original PNoC simulation framework (using HotSpot as the thermal simulator) with multithreaded applications from HPCCG (Edwards et al., 2008), UHPC (Campbell et al., 2012), and NAS parallel benchmarks (Bailey et al., 1991) with a different number of thread combinations. And compare PACT’s simulation results to the results generated using the original PNoC simulation framework. For the

transient power traces, we collect the average power value every 100 million instructions. We summarize the experimental setup in Table 3.5. The detailed model, architecture, policy, and experimental setup can be found in previous work (Narayan et al., 2019; Narayan et al., 2020). Since MRRG temperatures directly determine the heat power, we only compare the temperature results of PACT to HotSpot. Figure 3-10 shows the thermal maps of application *bt* with 96 threads simulated using both the original PNoC simulation framework and the PNoC simulation framework with PACT. Note that, MRRG is placed on the interposer layer. PACT thermal maps are almost identical to the thermal maps generated using HotSpot. We also show the transient simulation results compared to HotSpot in Figure 3-11. Table 3.6 shows the maximum and average temperature difference for these two PNoC simulation frameworks across all the experiments. As we see in the table, compared to the original PNoC simulation framework, the PNoC simulation framework with PACT has less than 1% maximum temperature difference, which demonstrates that PACT is also compatible with popular architecture-level performance and power simulators.

Table 3.5: Experimental setup of PNoC simulations.

Applications	<i>bt, ft, hpccg, is, lu, mg, shock, sp</i>
VF Settings	$V = 0.85 V, f = 533 MHz$
Average Core Power	0.83 W
# of threads	48,96
# of grids	64×64
Performance Threshold	10 %
# of cores in PACT	1
Solver in PACT	KLU
Heat Sink	Medium cost heat sink
# of instructions	10 billion

3.4.3 Liquid Cooling via Microchannels Simulation Results

To investigate the accuracy of the liquid cooling via microchannels model in PACT, we directly compare the steady-state and transient simulation results against 3D-ICE, which

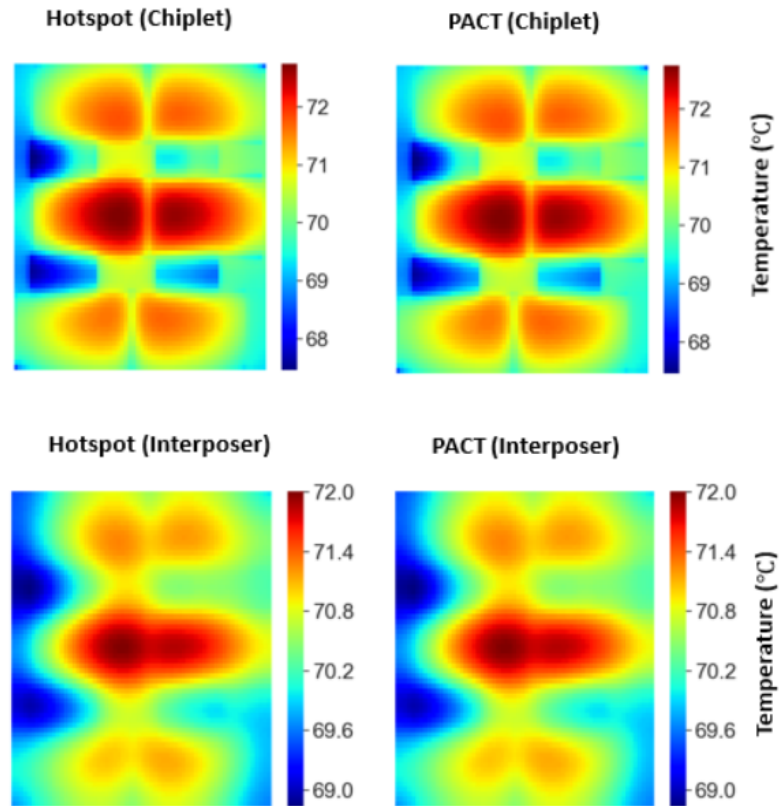


Figure 3-10: Thermal maps for running application *bt* with 96 threads and 10% performance constraint using original PNoC simulation framework and PNoC simulation framework using PACT. MRRG is on the interposer layer. The number of grids used in the simulation is set to 64×64 .

has already been validated against real prototypes (Sridhar et al., 2014). We select a liquid cooling chip stack as shown in Figure 3-12(a) and model it in both PACT and 3D-ICE. We summarize the validation setup in Table 3.7. Note that, we set the grid resolution to 1000×5 for these experiments and use the same setup in PACT and 3D-ICE. While 3D-ICE does not allow for arbitrary granularities in the liquid microchannel layer, especially for the liquid microchannels, PACT does support arbitrary grid resolutions for the whole liquid cooling chip stack, including the liquid microchannel layer. We summarize the simulation results of PACT and 3D-ICE in Figure 3-13. ΔT is the temperature difference between the temperature of the current step and the coolant inlet temperature. For steady-state

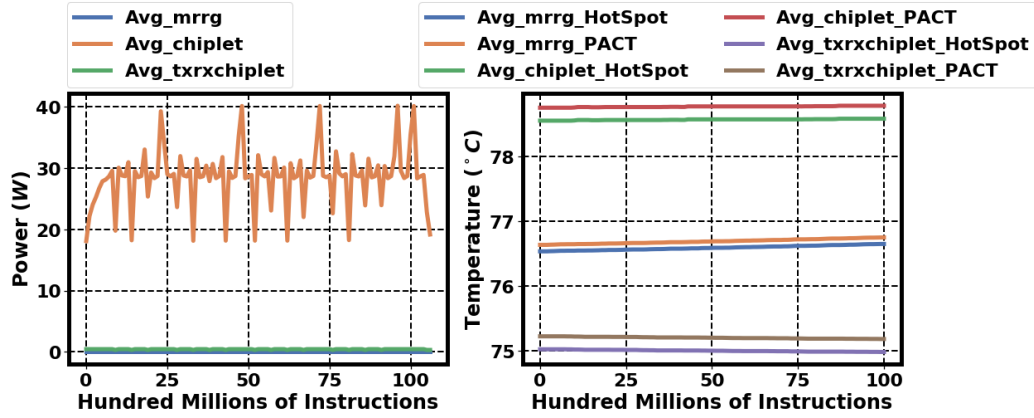


Figure 3-11: Transient temperature results for running application *hpccg* with 96 threads and 10% performance constraint using original PNoC simulation framework and PNoC simulation framework using PACT. The number of grids used in the simulation is set to 64×64 . The left image shows the average power traces and the right image shows the average temperature traces.

Table 3.6: PNoC simulation results.

# of threads	Apps	Max diff (°C)	Avg diff (°C)
48	bt	0.08	<0.05
	ft	0.08	<0.05
	hpccg	0.47	0.15
	is	0.11	<0.05
	lu	0.34	0.09
	mg	0.02	<0.05
	shock	0.12	<0.05
	sp	0.41	0.09
96	bt	0.31	0.08
	ft	0.37	0.16
	hpccg	0.67	0.19
	is	0.35	0.05
	lu	0.19	<0.05
	mg	0.38	0.16
	shock	0.55	0.21
	sp	0.61	0.27

and transient simulations, PACT shows a maximum temperature difference of 0.41°C and 1.12°C , respectively. Compared to 3D-ICE, PACT shows up to $1.6\times$ and $2.05\times$ speedup

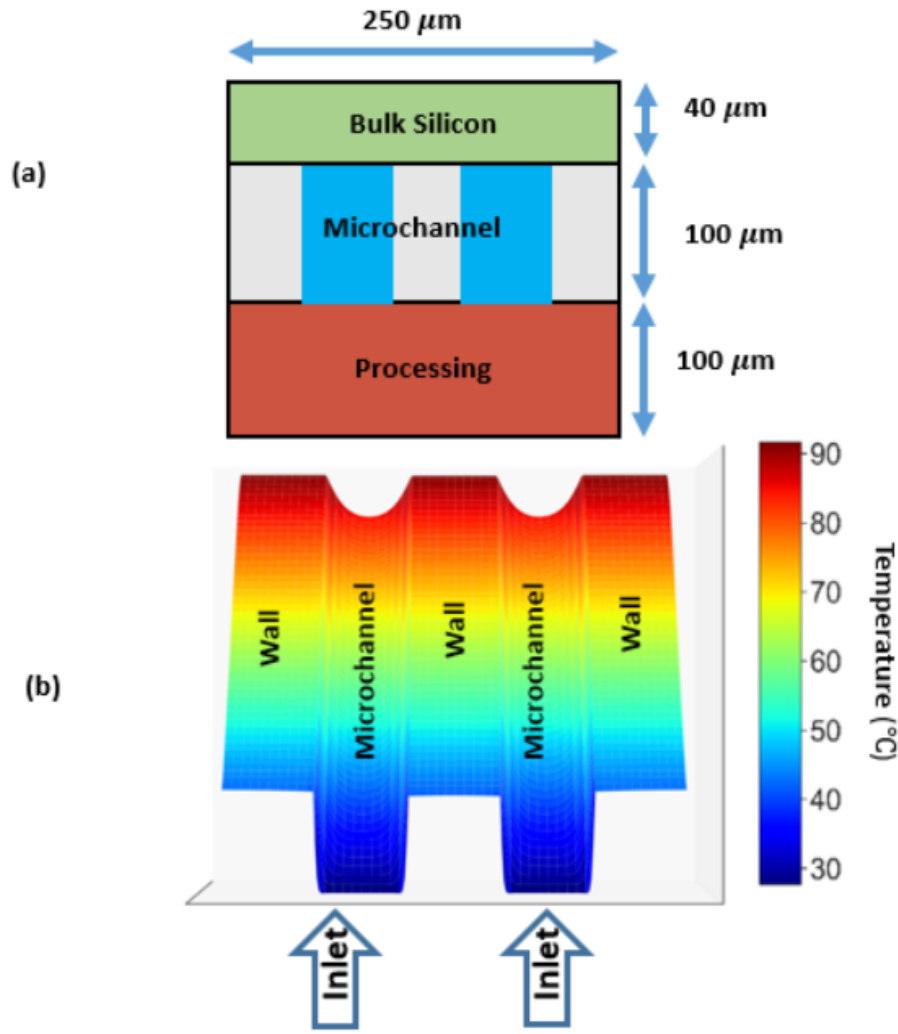


Figure 3-12: (a) The front view of the chip stack and (b) the microchannel layer thermal map (power density = 100 W/cm^2 and coolant velocity of 0.5 m/s).

for steady-state and transient simulations. We also observe that 3D-ICE does not consider the initial matrix factorization time into account when reporting the emulation time. Given the fact that the time it takes to factorize a matrix is included for PACT and is also one of the main contributing factors in PACT's runtime, PACT achieves an even higher speedup when compared to 3D-ICE. The main reason for this speedup is that PACT supports parallel thermal simulation. Figure 3-12(b) shows the microchannel layer thermal map in PACT (power density = 100 W/cm^2 and coolant flow velocity = 0.5 m/s). The temperature of the

coolant increases as the coolant flows across the chip, which results in a higher temperature at the outlet. This trend is expected since the coolant absorbs heat as it flows along the microchannel. Accuracy comparison of PACT’s liquid cooling model against another validated recent model (Kaplan et al., 2019) also shows very similar results of only up to 0.09°C maximum temperature difference.

Table 3.7: Validation setup of liquid cooling via microchannels simulations.

Chip Length	5 mm
Chip Width	$250\ \mu\text{m}$
Channel Length	5 mm
Channel Width	$50\ \mu\text{m}$
Wall Width	$50\ \mu\text{m}$
Heat Sink	fixed-air convection HTC heat sink
Air Convection HTC	$0.01\ \text{W}/\text{m}^2\text{K}$
# of Grids	1000×5
Uniform power densities	$12.5, 25, 50, 100\ \text{W}/\text{cm}^2$
Flow Velocities	$0.5, 1.0, 1.5, 2.0\ \text{m}/\text{s}$
Step Size	$3.33\ \text{ms}$
# of Steps	100
# of Cores in PACT	8
Steady-state Solver in 3D-ICE 2.2.6	SuperLU
Transient Solver in 3D-ICE 2.2.6	Backward-Euler
Steady-state Solver in PACT	AztecOO
Transient Solver in PACT	Trap

3.4.4 Standard-Cell-Level Validation of PACT against COMSOL and HotSpot

To validate the accuracy of PACT, we compare the steady-state and transient simulation results to COMSOL and HotSpot using different numbers of grids. We summarize the validation setup in Table 3.8. The detailed statistics of the MPSoCs from OpenROAD are shown in Table 3.9. To ensure standard-cell-level thermal simulation, the grid resolution should depend on the number of standard cells, standard cell size, and design complexity. Based on the MPSoCs we used in the experiments, a grid resolution of equal or higher

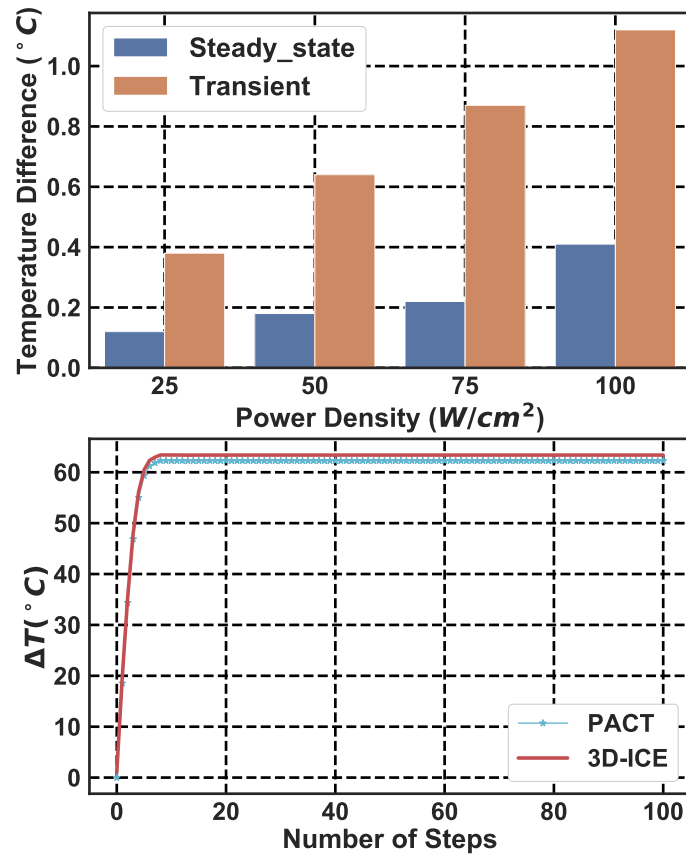


Figure 3-13: Liquid cooling via microchannels simulation results. The top image shows the maximum temperature difference for each power profile when coolant flow velocity = 0.5, 1, 1.5, and 2 m/s. The bottom image shows the transient temperature curve of PACT and 3D-ICE when power density = 100 W/cm² and liquid flow velocity = 0.5 m/s. This case shows the maximum temperature difference between PACT and 3D-ICE.

than 256×256 should be used to simulate the standard-cell designs. Utilization is defined as the ratio of the area of standard cells, macros, and the pad cells to the area of the chip minus the area of the sub floorplan. Higher utilization indicates more logic is packed into a smaller area, which in turn results in higher power density. To show the scalability of PACT, the MPSoCs in our test set have different power values and chip sizes. The steady-state thermal maps (256×256) of the MPSoCs from OpenROAD are shown in Figure 3-14. These thermal maps indicate that the maximum chip temperature across all cases is close

to 90°C, and the maximum thermal gradient is around 9°C. We also show the thermal maps and error map of PicoSoC with 95% utilization simulated using HotSpot, PACT, and COMSOL in Figure 3-15. The steady-state grid temperature validation results are shown in Figure 3-16. We observe that in comparison to COMSOL, PACT has maximum, average, and minimum grid temperature errors of 2.77%, 1.76%, and 0.89%, respectively, which demonstrates the accuracy of PACT’s steady-state simulation. The error is calculated with respect to COMSOL by dividing the grid temperature difference (°C) by the maximum on-chip temperature reported by COMSOL. Figure 3-16 also shows the accuracy results for HotSpot with respect to COMSOL. As we see in the figure, when compared to COMSOL, PACT and HotSpot have similar maximum, average, and minimum errors.

Table 3.8: Validation setup of HotSpot, COMSOL, and PACT.

Simulator	COMSOL	HotSpot 6.0	PACT
# of grids	256×256		
Solver	FEM-based solver	SuperLU, RK4	KLU, AztecOO, Trap
Heat Sink	fixed-air convection HTC heat sink		
Air Convection HTC	1e5 W/m ² K		
# of Cores in PACT	1		
Step Size	3.33 ms		
Total Step	30		

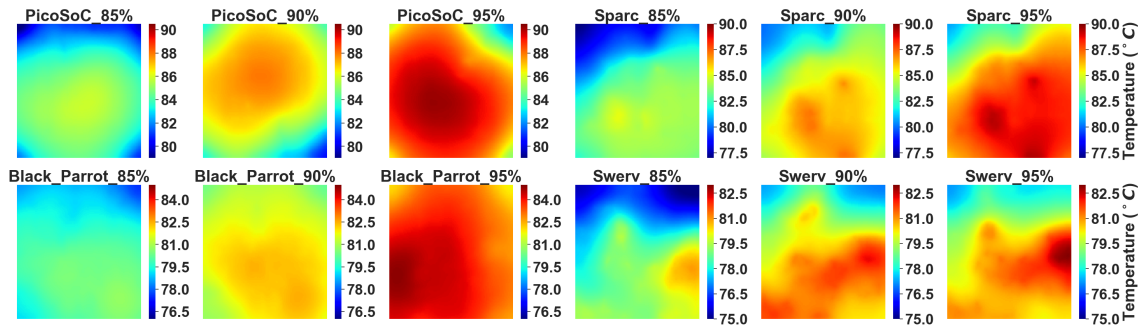
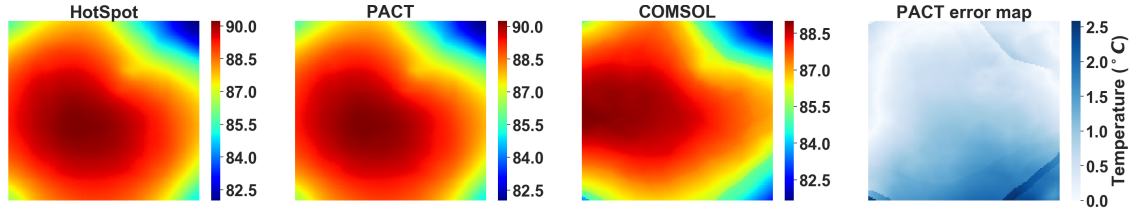


Figure 3-14: PACT’s thermal maps for the MPSoCs from OpenROAD. The number of grids used in the simulation is set to 256×256. Different utilization levels (shown next to chip names) affect floorplan, chip size, and power density.

Table 3.9: Statistics of the realistic MPSoCs from the OpenROAD benchmark set.

MPSoCs	Avg PD(W/cm^2)	Freq(GHz)	Util($\%$)	# of standard cells	Dimension (μm^2)
PicoSoC	368	3	85	254815	1567×1577
PicoSoC	387	3	90	254815	1522×1534
PicoSoC	409	3	95	254815	1483×1493
Sparc	351	3	85	192871	1225×1244
Sparc	374	3	90	192871	1194×1198
Sparc	391	3	95	192871	1162×1176
Black_Parrot	319	3	85	71285	769×779
Black_Parrot	343	3	90	71285	748×752
Black_Parrot	362	3	95	71285	728×732
Swerv	311	3	85	63423	620×622
Swerv	326	3	90	63423	602×610
Swerv	338	3	95	63423	595×600

**Figure 3-15:** Thermal maps for PicoSoC with 95% utilization simulated using HotSpot, PACT, and COMSOL. The rightmost image shows the error map of PACT when compared to HotSpot. The number of grids used in the simulation is set to 256×256 .

Next, we compare the steady-state simulation time of HotSpot and PACT using the setup as shown in Table 3.8 with various numbers of cores (8, 16, 56, and 112). We also include finer grid resolutions such as 512×512 and 1024×1024 . We show the speedup of PACT’s simulation time against HotSpot in Figure 3-17. We select KLU and AztecOO as PACT’s solvers for parallel steady-state thermal simulations with multiple cores. As we see in Figure 3-17, for steady-state simulations using 256×256 grids with a relatively small number of cores (8 and 16), HotSpot is faster than PACT by as much as $2.3 \times$. But note that the simulation time is relatively short in these cases (22-134 seconds). The reason is that since PACT is written in Python (and HotSpot is written in C), the front-end processing time of PACT is longer than HotSpot. Another possible reason is that Xyce 6.12 (PACT’s SPICE

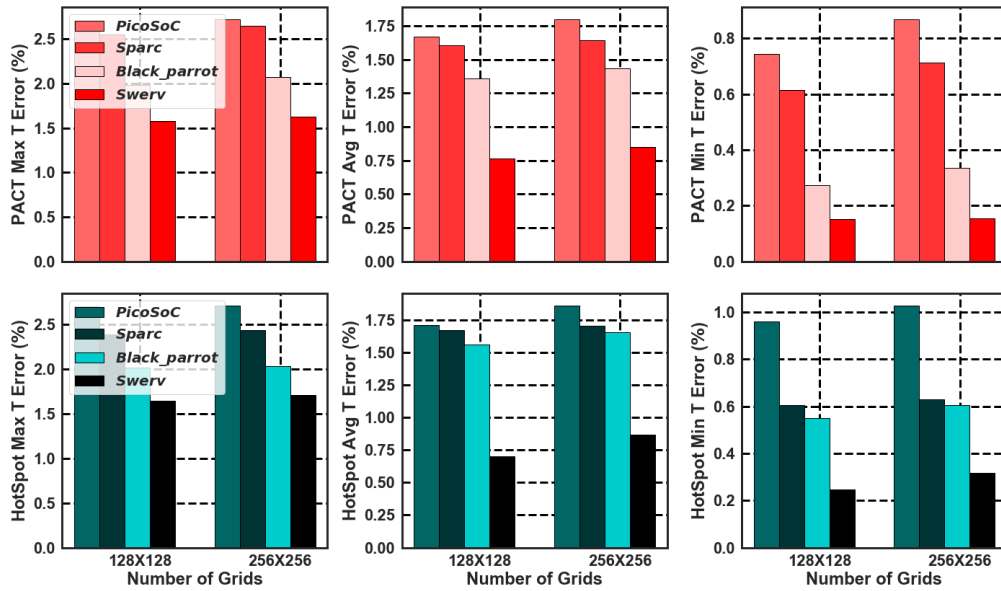


Figure 3-16: Steady-state grid temperature validation results (utilization = 95%). MP-SoCs with 95% utilization result in the highest maximum, average, and minimum grid temperature error. The error is calculated with respect to COMSOL.

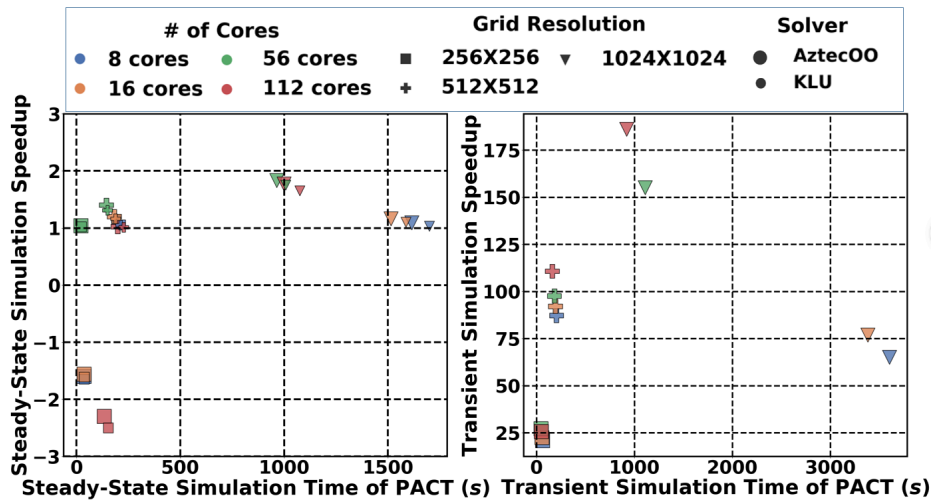


Figure 3-17: Steady-state and transient simulation times of PACT. The speedup of PACT against HotSpot is shown on the y-axis. The speedup is computed as the ratio of the simulation times of HotSpot and PACT. Negative values mean HotSpot is faster than PACT for those cases.

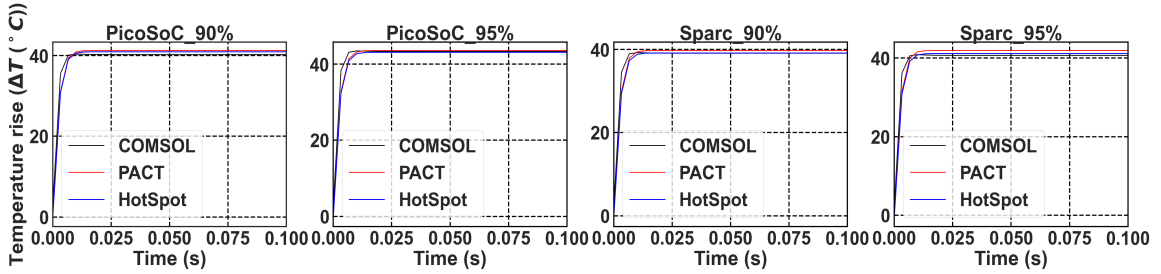


Figure 3-18: Transient validation results. The number of grids used in the simulation is set to 256×256 . Due to the space limit, we only show the results that have the highest transient temperature difference.

engine) uses a one-step DC analysis to perform operation point analysis, which slows down the steady-state simulation. When the problem size is relatively small (e.g., 256×256), using a large number of cores (e.g., 112) results in a high communication cost between cores and nodes. This communication cost is a potential timing bottleneck (Hutchinson et al., 2002) and may result in longer simulation times. For standard-cell-level problems (e.g., 512×512 and 1024×1024), PACT results in shorter simulation times than HotSpot. The maximum steady-state simulation speedup of PACT compared to HotSpot is $1.83 \times$ (1024×1024 with 56 cores). Note that, using 112 cores for problem sizes of 512×512 and 1024×1024 also has a high communication cost issue and results in longer simulation times than using 56 cores.

We also run steady-state simulations using PACT with KLU. For parallel simulation using a serial solver like KLU, the thermal netlist is evaluated and assembled using multiple processors, but only one processor is used to solve the netlist (Hutchinson et al., 2002). However, AztecOO is a parallel iterative solver which uses multiple processors to evaluate, assemble, and solve the thermal netlist. In Figure 3-17, where the thermal netlist is evaluated and assembled with the KLU solver using multiple processors, PACT still achieves a maximum speedup of $1.75 \times$ (1024×1024 with 56 cores) compared to HotSpot.

For transient validations, we create a step response for each MPSoC and compare the grid temperature results against COMSOL and HotSpot. We run each transient thermal

simulation with a step time of 3.33 ms and the total simulation time of 99.9 ms (total steps of 30). We show the average grid temperature simulation results of Sparc, Black_Parrot, Swerv, and PicoSoC in Figure 3-18. Compared to HotSpot, PACT has a maximum and average temperature difference of 0.05% and 0.01% across all the experiments. Compared to COMSOL, PACT has a maximum and average difference of 3.28% and 1.1% , respectively. ΔT is the temperature difference between the temperature of the current step and the ambient temperature. Since OpenSTA (Ajayi et al., 2019) lacks dynamic power traces, we utilize the steady-state power profiles from OpenROAD and randomly apply $\pm 15\%$ additional power values for each standard cell to create synthetic transient power traces. We simulate both PACT and HotSpot using the same setup as shown in Table 3.8. The results are shown in Figure 3-19. We see that PACT temperature traces overlap with HotSpot temperature traces. The steady-state and transient validation results indicate HotSpot and PACT are at the same accuracy level.

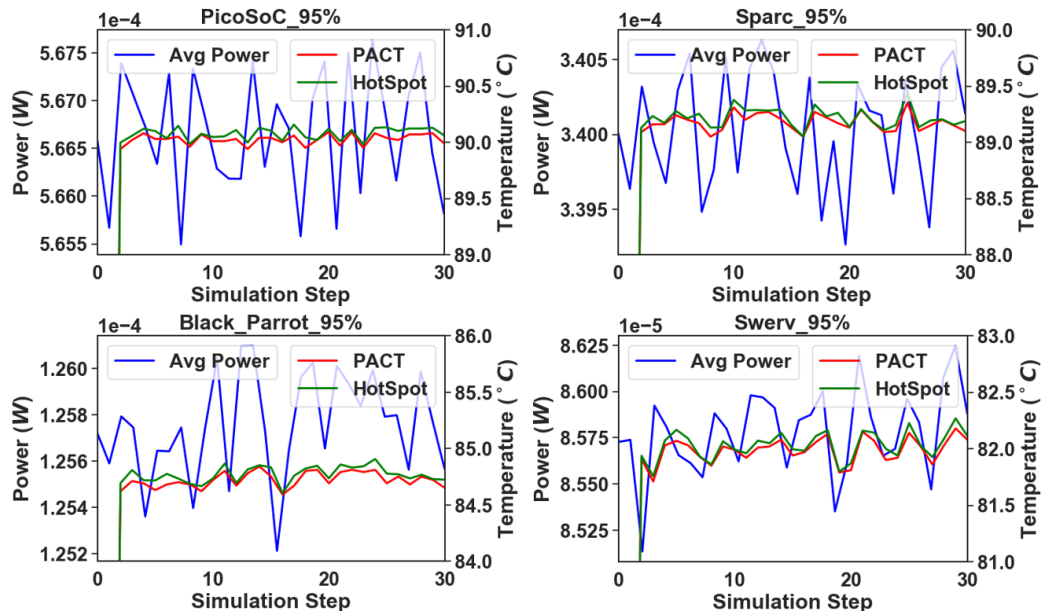


Figure 3-19: Synthetic power traces for PACT and HotSpot simulations. Due to the space limit, we only show the results that have the highest temperature difference.

We then compare the transient simulation time of HotSpot and PACT with cores = 8,

16, 56, and 112. For parallel transient thermal simulations with multiple cores, we select TRAP as the solver of PACT. Figure 3·17 demonstrates that PACT outperforms HotSpot in every test case. Since HotSpot uses the explicit adaptive RK4 method (4th order Forward Euler), to ensure the accuracy of simulation results, adaptive RK4 needs to decrease the minimum simulation step to satisfy the numerical stability constraint (Distefano, 1968). On the other hand, PACT uses the TRAP solver (2nd order Backward Euler method) that eliminates the numerical instability problem. PACT achieves a speedup of up to 186× when compared to HotSpot (1024×1024 with 112 cores). We also observe that different grid resolutions affect thermal netlist generation, hypergraph partition, and solver running time, while chip size only affects thermal netlist generation time. Across all the standard-cell-level simulations for the designs from OpenROAD, PACT’s total running time is dominated by the hypergraph partition and solver running time. The thermal netlist generation time is negligible.

3.4.5 Standard-Cell-Level Comparison of PACT against MTA

MTA (Ladenheim et al., 2016) is a thermal simulator that is able to perform standard-cell-level thermal simulations. We compare PACT’s temperature results and simulation speed for both steady-state and transient analysis to that of MTA 2.0 using full industrial designs from OpenROAD. The experimental setup is almost the same as Table 3.8. We change the transient step size to 3.33 μ s with a total number of steps of 100. We also use the same medium-cost heat sink in both PACT and MTA. We select the default mesh provided by MTA, which results in 639920 degrees of freedom. To ensure a fair comparison, we set the grid resolution in PACT to 256×256. For steady-state simulations in MTA, we use {mode 0}. Since MTA does not support adaptive mesh refinement for parallel thermal simulations, we use {mode 2} to perform transient simulations with adaptive time step size. We carry out linear heat model parallel thermal simulations with MPICH (Gropp et al., 1999). The steady-state and transient maximum temperature differences are 0.45°C and 0.83°C. We

average the simulation time for each MPSoC selected from OpenROAD as shown in Table 3.9 and present comparison in Figure 3-20. Compared to MTA, PACT achieves a maximum speedup of $1.98\times$ and $9.64\times$ for steady-state and transient simulations, respectively. Since MTA is a FEM-based thermal simulator and PACT is based on the compact thermal modeling methodology, the complexity of solving the second-order heat equation is obviously higher than solving the first-order thermal RC network. Even with an adaptive time step size, PACT still achieves better simulation time than MTA.

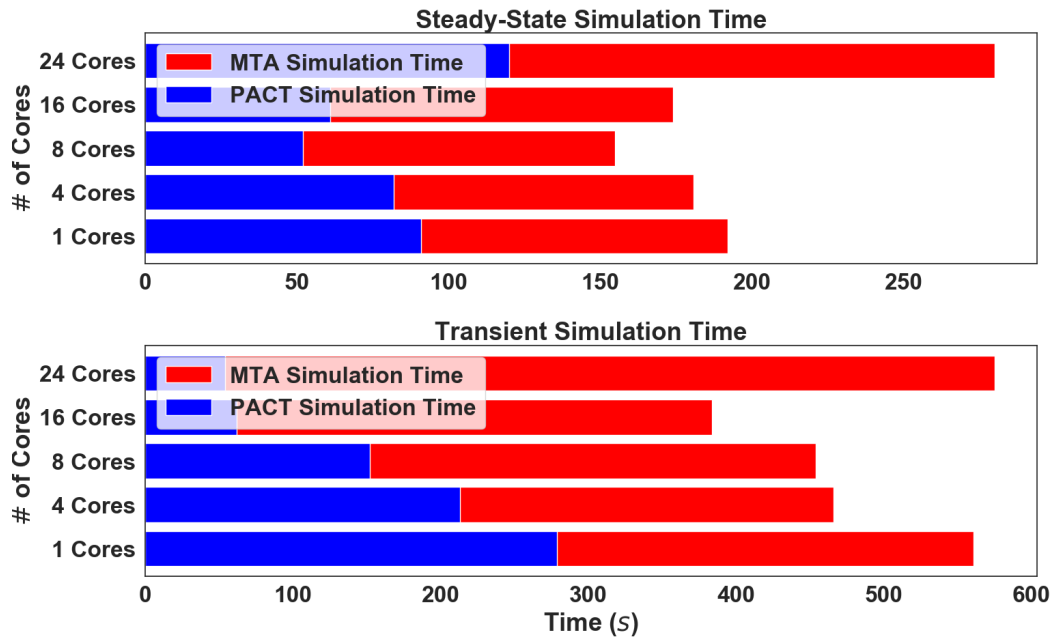


Figure 3-20: Steady-state and transient simulation time of PACT and MTA.

3.5 Case Study: Modeling Diamond Heat Spreaders Using PACT

To further demonstrate the applicability and extensibility of PACT, we perform a case study on using PACT to model and evaluate the lab-grown diamond heat spreaders.

3.5.1 Introduction

Existing cooling solutions such as forced air cooling via fans or traditional pin-fin heat sinks are often not sufficient to mitigate these high power density hot spots efficiently and lead to over/under-cooling, affecting system design cost and power. The cooling performance is relatively low for passive cooling methods such as pin-fin heat sink. However, for active cooling methods such as forced air cooling via fans and liquid cooling, the system requires additional cooling power (fan power and liquid pumping power). It's hard to optimize the existing cooling solutions at design time and runtime to achieve both high computing performance for processors and energy efficiency for the cooling methods. Lab-grown diamond heat spreaders have the potential to provide better cooling performance compared to traditional copper heat spreaders due to the high thermal conductivity, the ability to directly bond them on silicon, no additional cooling power needed, and allow for an ultra-thin silicon layer (Jagannadham, 1998; Zhou et al., 2012). However, lab-grown diamond heat spreader thermal models are usually developed and simulated using FEM-based multiphysics simulators (e.g., COMSOL and ANSYS (COM, 1998; Madenci and Guven, 2015)). Such commercial simulators are computationally expensive and experience long solution times along with large memory requirements (Yuan et al., 2019a; Yuan et al., 2019b), which results in simulation timing overhead for parametric studies and thermal evaluations for lab-grown diamond heat spreaders with real-world high-performance processors. Due to the aforementioned modeling challenges using commercial FEM-based simulators, none of the existing works have evaluated the thermal behavior of lab-grown diamond heat spreaders on real-world high-performance processors running realistic application benchmarks. We use PACT to compare the cooling performance of lab-grown diamond heat spreaders against traditional copper heat spreaders using real-world high-performance processors. To demonstrate the cooling advantages of lab-grown diamond heat spreaders, we select three different real-world high-performance processors (Intel i7

6950×, IBM Power9, and PicoSoC) and compare the cooling performance in terms of maximum temperature reductions and thermal gradient reductions between lab-grown diamond heat spreaders and traditional copper heat spreaders. We also carry out several parametric studies to demonstrate the impact of cooling performance of lab-grown diamond heat spreaders with different chip thicknesses and cooling packages.

3.5.2 Materials and Methods

In this section, we first discuss the models of processors and interconnects we build for the steady-state and transient simulations. Next, we illustrate our methodology for collecting transient power traces from realistic application benchmarks.

Processor Model

We build three different real-world processor models in PACT. Intel i7 6950× (Sima, 2018) is a desktop processor, IBM Power9 processor (Sadasivam et al., 2017) is a server processor, and PicoSoC (Ajayi et al., 2019) is a mobile processor. For Intel i7 6950× and IBM Power9, we model the processors based on the reported architecture-level floorplan and thermal design power (TDP) (220 W for IBM Power9 and 140 W for Intel i7 6950×). For PicoSoC, we directly utilized the coordinates and power values of the standard cells to generate standard-cell-level power maps using OpenROAD (Ajayi et al., 2019). We assume an extreme power case for PicoSoC with an operating frequency of 3 GHz and a total power of 9 W. For Intel i7 6950× and IBM Power9, we create 5 different chip stacks to compare the cooling performance of lab-grown diamond heat spreaders and traditional copper heat spreaders. Figure 3-21 shows the chip stacks for Intel i7 6950× and IBM Power9. Since PicoSoC is a mobile chip, using heat sinks is not possible with mobile chips due to size/volume constraints, and package temperature constraints are typically stricter compared to desktop and server processors. In this case, we build five additional chip stacks with no heat sinks and thermal interface material (TIM) (no TIM2) for PicoSoC as shown

in Figure 3-22. Besides, the TIM1 layer is relatively thinner than the desktop and server processors. We assign a fixed-air convection HTC on top of the chip stacks to represent the forced-air cooling via fans package. Chip stack 1 is used to mimic the real-world processor with a copper heat spreader, and chip stack 3 represents a real-world processor with a lab-grown diamond heat spreader ($TC_{diamond1} = 7.28(T)^{-1.42} MW/mK$). Chip stack 2 is used to directly compare the cooling performance of the copper heat spreader and diamond heat spreader. Note that, one of the advantages of lab-grown diamond heat spreaders is that they are able to be directly bonded to the silicon with an ultra-thin TIM layer (or no TIM layer is needed). In contrast, traditional copper heat spreaders require a thick TIM layer (Liang et al., 2017; Liang et al., 2018; Liang et al., 2019). Comparing chip stack 1 and 3 is more realistic than comparing chip stack 1 and chip stack 2. Chip stack 4 represents a real-world processor with a higher thermal conductivity lab-grown diamond heat spreader ($TC_{diamond2} = 10.9(T)^{-1.42} MW/mK$). Chip stack 5 mimics the processor with an ultra-thin processor layer (silicon layer). The floorplans of Intel i7 6950 \times and IBM Power9 are shown in Figure 3-23. Since PicoSoC is a standard-cell design, the floorplan of PicoSoC is very similar to a mesh.

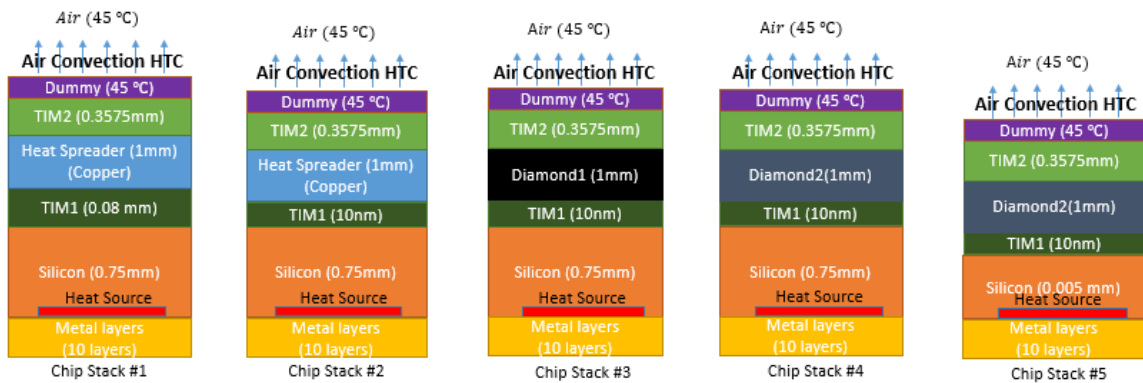


Figure 3-21: Chip stacks for IBM Power9 and Intel i7 6950 \times .

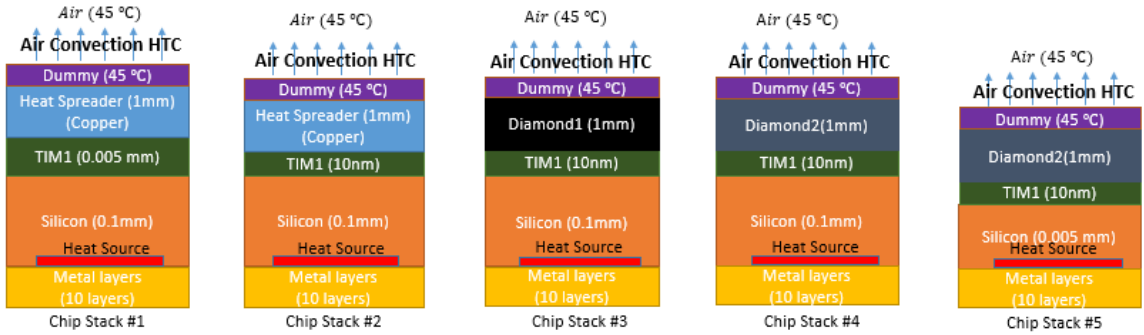


Figure 3-22: Chip stacks for PicoSoC.

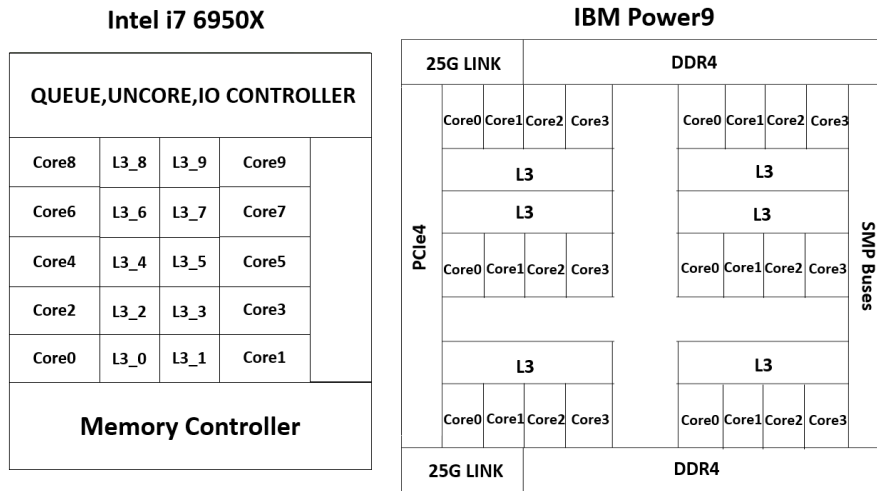


Figure 3-23: Intel i7 6950× and IBM Power9 floorplans.

Interconnect Model

To mimic the realistic interconnects in the real chips, we add additional interconnect metal layers to the chip stacks and assign additional dynamic power to represent the power delivery network’s power consumption. Figure 3-24 shows the layer stack of the interconnect model we added to the chip stack. We assume the flip-chip design and the processing layer is in between the heat spreader and interconnect model. Since metal layers 1-8 are local interconnects, we abstract metal 1-8 layers into one layer and assign a joint thermal resistivity of 75% copper and 25% silicon oxide to reduce the simulation problem size to this abstract layer. For metal 9 and 10 layers, since these metal layers are used for global

connection, we use these two layers to build a power delivery network. The floorplans for metal 9 and 10 layers are shown in Figure 3-25. The golden lines represent the metal lines, and the rest of the layer consists of silicon oxide. To further reduce the simulation problem size, for desktop processor and server processor chip stacks as shown in Figure 1, metal 9 and 10 layers metal width and pitch are set to $200\ \mu\text{m}$ and $400\ \mu\text{m}$. For mobile processors, to ensure simulation accuracy, metal 9 and 10 layers metal width and pitch are set to $20\ \mu\text{m}$ and $40\ \mu\text{m}$. The total power consumption of the interconnect layers is assumed to be 10% of the total chip power (Adhinarayanan et al., 2016). Since Metal 9 and 10 are power delivery network layers, they are assumed to consume 40% of the interconnect power. For metal 1-8, each layer is assumed to consume 7.5% of the interconnect power.



Figure 3-24: The layer stack of the interconnect model.

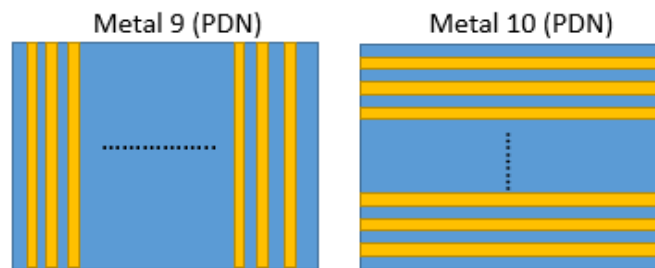


Figure 3-25: Power delivery network model.

Realistic Power Traces Collection

To carry out transient simulations for Intel i7 6950× with realistic power traces, we first use architecture-level performance simulators such as Sniper (Carlson et al., 2011) to run realistic application benchmarks and then input the program metrics to the power simulator, McPAT (Li et al., 2009), to collect the power traces. The power traces are calibrated using the reported TDP, and the collected power traces are sent to PACT to carry out transient simulations. We select the parallel applications from NAS parallel benchmarks (Bailey et al., 1991) and choose different mapping policies to map different applications to different cores to study the multi-program and multi-threaded workload scenarios. For PicoSoC, since the standard-cell design lacks dynamic power traces, we utilize the steady-state power values of PicoSoC and randomly applied -15% or +15% additional power values for each standard cell and create synthetic transient power traces. Since IBM Power9 uses Power instruction set architecture (ISA) and architecture-level performance simulators such as Sniper have better support for ×86 ISA and less support for RISC ISA such as Power ISA. We only carry out steady-state simulations for IBM Power9.

3.5.3 Results and Discussions

In this section, we first validate the accuracy of the thermal models in PACT. Then we demonstrate the steady-state and transient cooling performance comparison results of traditional copper heat spreaders and diamond heat spreaders. Last but not least, we show the parametric study results of the chip thickness and cooling packages. Note that the compact thermal modeling methodology always places the temperature node at the center of the bottom surface of the layer. When we are demonstrating and discussing the temperature of the silicon layer, we are always referring to the temperature of the heat source.

Validation of the Model

We use the following chip stacks as shown in Figure 3-26 to validate the steady-state accuracy of the thermal models in PACT. The silicon layer has a dimension of $2400 \times 2475 \mu\text{m}^2$. A $1600 \times 1650 \mu\text{m}^2$ hot spot is placed at the center of the silicon layer with a heat flux of $265 \text{ W}/\text{cm}^2$. The silicon layer consumes a total power of 7 W . And the rest of the layer consumes no power. We build all three chip stacks in ANSYS and PACT and directly compare the simulation accuracy. The simulation grid resolution in PACT is set to 100×100 . The maximum steady-state temperature difference between PACT and ANSYS is 0.51°C (chip stack 1). It takes a maximum of 6.97 seconds to run the steady-state simulations in PACT for the chip stacks shown in Figure 3-26 with a parallel configuration of 4 cores. We

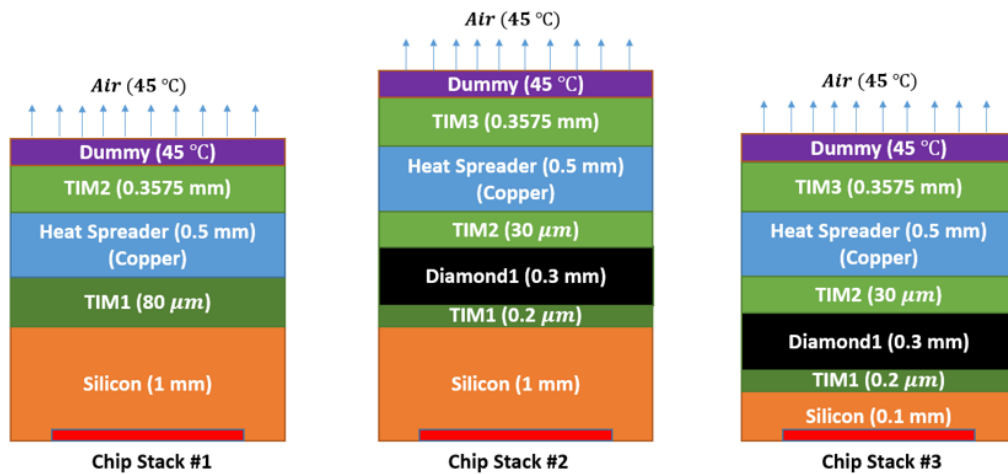


Figure 3-26: Chip stacks for steady-state validation.

use the chip stacks and die floorplan as shown in Figure 3-27 to validate the transient simulation results. The total chip area is $50 \times 50 \mu\text{m}^2$, and the die contains seven power lines colored in red. Each power line is $1 \mu\text{m}$ wide and $30 \mu\text{m}$ long. Each power line consumes a uniform power of $120 \mu\text{W}$. We switch on and off all the power lines at frequencies of 1, 10, 100, and 1000 Hz for 1 second to validate the transient simulation results accuracy of PACT against ANSYS. The simulation grid resolution is set to 100×100 , and the minimum transient solver step size is set to 0.1 ms . When compared to ANSYS, the maximum tran-

sient error is 0.35°C (chip stack 1 @ 1000 Hz). It takes a maximum of 5.93 minutes to run the transient simulation in PACT for the chip stacks shown in Figure 3-27 with a parallel configuration of 4 cores.

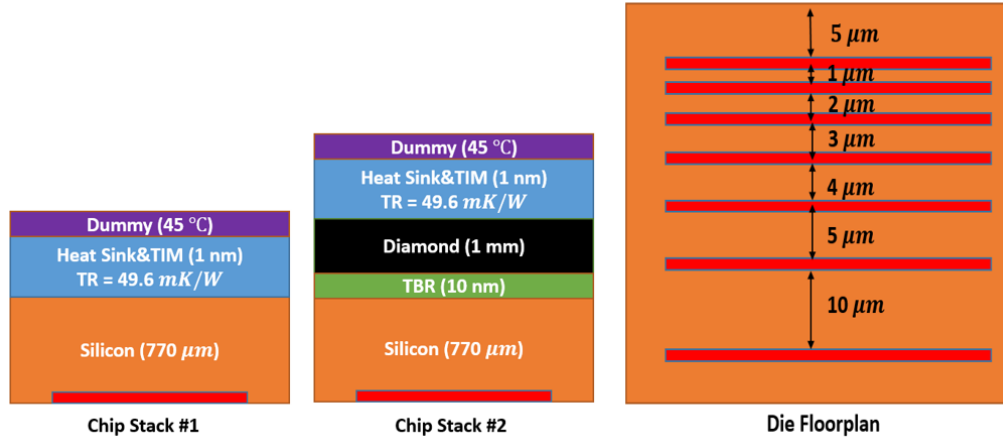


Figure 3-27: Chip stacks and die floorplan for transient validation.

Steady-State Comparisons

In this subsection, we compare the steady-state simulation results for the IBM Power9, Intel i7 6950 \times , and PicoSoC using the chip stacks as shown in Figures 3-21 and 3-22. For IBM Power9 and Intel i7 6950 \times , we use a grid resolution of 400×400 with grid sizes of $68.5 \times 63.3\ \mu\text{m}^2$ and $36.7 \times 42\ \mu\text{m}^2$, respectively. For PicoSoC, since it's a standard-cell design with a fine granularity power map and floorplan, we select to use a high grid resolution of 1024×1024 with a grid size of $1.46 \times 1.46\ \mu\text{m}^2$. The grid resolution is selected based on the size of the smallest functional unit of the processor. The selected grid size is similar to the smallest functional unit of the processor. Since Intel i7 6950 \times and IBM Power9 have architecture-level floorplans, the grid resolution is relatively coarse compared to the standard-cell-level floorplan of PicoSoC. Based on cooling packages recommended by Intel, we use a high air convection HTC of $30\ \text{KW}/\text{m}^2\text{K}$. For IBM Power9 and PicoSoC, we use air convection HTCs of $20\ \text{KW}/\text{m}^2\text{K}$ and $1\ \text{KW}/\text{m}^2\text{K}$. The convection HTC values

are adopted from previous work (Wei et al., 2012). We obtain the steady-state power map of Intel i7 6950× by running Sniper and McPAT with applications *bt*, *cg*, *dc*, *ep*, *ft*, *is*, *lu*, *mg*, *sp*, and *ua* from NAS parallel benchmarks and average the transient power traces. The steady-state power map has been calibrated to the reported TDP from Intel. For IBM Power9, we use the reported TDP and power breakdown from previous work (Sima, 2018) to calculate the power values for core, L3 cache, Nest, I/O, and DDR4 memory controller. We extract the steady-state power map of PicoSoC by running the OpenROAD project and the interface between OpenROAD and PACT. Figures 3-28, 3-29, and 3-30 show the steady-state heat map comparisons of Intel i7 6950×, IBM Power9, and PicoSoC with chip stacks 1 and 3. Chip stack 1 is the more realistic chip stack with a traditional copper heat spreader, and chip stack 3 is a realistic chip stack with a relatively lower diamond thermal conductivity lab-grown diamond heat spreader. Chip stack 2 is just for direct cooling performance comparison of the heat spreaders, assuming that the traditional copper heat spreaders are able to be directly bonded to the silicon layer. We show the Intel i7 6950×, IBM Power9, and PicoSoC steady-state layers 0-5 simulation results for all the chip stacks in Tables 3.10, 3.11, and 3.12. Layers 0, 1, and 2 are the metal 10, 9, and 1-8 layers, respectively. Layer 3 is the silicon layer, and layer 4 is the TIM layer placed above the silicon layer. Layer 5 is the heat spreader/diamond layer. The maximum steady-state simulation time of PACT for these high-performance chips is 259 seconds.

Based on our observations from Figures 3-28, 3-29, and 3-30 and Tables 3.10, 3.11, and 3.12, replacing the traditional copper heat spreaders with lab-grown diamond heat spreaders achieves at least 12.49°C (IBM Power9 chip stacks 1 and 3) and 1.89°C (PicoSoC chip stacks 1 and 3) maximum temperature and thermal gradient reductions, respectively. For Intel i7 6950× and IBM Power9 with lab-grown diamond heat spreaders, the maximum temperatures on-chip are less than 81°C. The throttling temperature for mobile processors is around 70-80°C (depends on the specific model of processor), with lab-grown dia-

mond heat spreaders, the maximum temperature of PicoSoC is less than 76°C . The above temperature reductions are mainly because the thermal conductivity of diamond is higher than copper, and the diamond heat spreaders are able to be directly bonded to the silicon layer, which results in lower vertical thermal resistance. In addition, we also observe that the temperature and thermal gradient reductions are highly correlated with the chip stack thickness. For Intel i7 6950 \times and IBM Power9, when switching the diamond thermal conductivity from $7.28(T)^{(-1.42)}\text{MW}/\text{mK}$ to $10.9(T)^{(-1.42)}\text{MW}/\text{mK}$ (chip stacks 3 and 4), the maximum temperature of the chip barely changes. The reason is that the thick TIM and silicon layers dominate the vertical thermal resistance of the chip stack. Using a high thermal conductivity diamond heat spreader cannot significantly benefit the temperature reductions. Whereas, for PicoSoC, the chip stack is much thinner than Intel i7 6950 \times and IBM Power9. When comparing PicoSoC chip stacks 3 and 4, we observe a maximum temperature reduction of 3.69°C . In addition, when we scale the silicon layer thickness to $5\ \mu\text{m}$, the maximum temperature and thermal gradient reductions increase to 19.76°C (PicoSoC chip stacks 1 and 5) and 15.69°C (IBM chip stacks 1 and 5), respectively. The hot spot locations and the number of hot spots also affect the maximum temperature and thermal gradient reductions. For Intel i7 6950 \times and PicoSoC, since the hot spots are gathering in the silicon layer, we observe maximum temperature reductions of 13.75°C and 13.21°C (chip stack 1 and chip stack 3), respectively. However, for IBM Power9, the hot spots are spread, and that's why the temperature reduction is lower than Intel i7 6950 \times and PicoSoC. In summary, compared to traditional copper heat spreaders, lab-grown diamond heat spreaders achieve maximum steady-state temperature and thermal gradient reductions of 19.76°C and 15.69°C , respectively.

Transient Comparisons

Next, we carry out transient simulations for Intel i7 6950 \times and PicoSoC. We use chip stacks 1 and 3 as shown in Figures 3-21 and 3-22. For Intel i7 6950 \times , we obtain the tran-

Table 3.10: Intel i7 6950× steady-state results. ID stands for chip stack identification number. T_{max} stands for maximum temperature ($^{\circ}\text{C}$) and ΔT stands for temperature gradient ($^{\circ}\text{C}$).

Layer	0	1	2	3	4	5
$T_{max}(ID1)$ ($^{\circ}\text{C}$)	84.64	84.64	84.60	84.54	81.13	70.43
$T_{max}(ID2)$ ($^{\circ}\text{C}$)	77.77	74.77	74.73	74.67	71.08	71.07
$T_{max}(ID3)$ ($^{\circ}\text{C}$)	70.89	70.89	70.85	70.79	67.00	66.99
$T_{max}(ID4)$ ($^{\circ}\text{C}$)	70.47	70.47	70.44	70.37	66.56	66.55
$T_{max}(ID5)$ ($^{\circ}\text{C}$)	66.76	66.76	66.72	66.66	66.63	66.62
$\Delta T(ID1)$ ($^{\circ}\text{C}$)	17.50	17.50	17.46	17.43	14.98	7.35
$\Delta T(ID2)$ ($^{\circ}\text{C}$)	11.10	11.10	11.06	11.04	8.36	8.35
$\Delta T(ID3)$ ($^{\circ}\text{C}$)	5.39	5.39	5.35	5.32	2.31	2.31
$\Delta T(ID4)$ ($^{\circ}\text{C}$)	4.72	4.72	4.69	4.65	1.59	1.59
$\Delta T(ID5)$ ($^{\circ}\text{C}$)	1.81	1.81	1.77	1.75	1.72	1.71

Table 3.11: IBM Power9 steady-state results. ID stands for chip stack identification number. T_{max} stands for maximum temperature ($^{\circ}\text{C}$) and ΔT stands for temperature gradient ($^{\circ}\text{C}$).

Layer	0	1	2	3	4	5
$T_{max}(ID1)$ ($^{\circ}\text{C}$)	92.82	92.82	92.82	92.80	89.12	77.58
$T_{max}(ID2)$ ($^{\circ}\text{C}$)	82.59	82.59	82.59	82.57	78.23	78.22
$T_{max}(ID3)$ ($^{\circ}\text{C}$)	80.33	80.33	80.33	80.31	75.70	75.69
$T_{max}(ID4)$ ($^{\circ}\text{C}$)	80.10	80.10	80.10	80.08	75.45	75.44
$T_{max}(ID5)$ ($^{\circ}\text{C}$)	75.58	75.58	75.58	75.56	75.53	75.51
$\Delta T(ID1)$ ($^{\circ}\text{C}$)	14.03	14.03	14.03	14.01	11.59	3.97
$\Delta T(ID2)$ ($^{\circ}\text{C}$)	8.32	8.32	8.32	8.33	5.05	5.04
$\Delta T(ID3)$ ($^{\circ}\text{C}$)	5.11	5.11	5.11	5.09	1.40	1.38
$\Delta T(ID4)$ ($^{\circ}\text{C}$)	4.72	4.72	4.72	4.72	0.98	0.98
$\Delta T(ID5)$ ($^{\circ}\text{C}$)	1.14	1.14	1.14	1.13	1.11	1.09

Table 3.12: PicoSoC steady-state results. ID stands for chip stack identification number. T_{max} stands for maximum temperature ($^{\circ}\text{C}$) and ΔT stands for temperature gradient ($^{\circ}\text{C}$).

Layer	0	1	2	3	4	5
$T_{max}(ID1)$ ($^{\circ}\text{C}$)	88.89	88.89	88.88	88.65	84.61	78.68
$T_{max}(ID2)$ ($^{\circ}\text{C}$)	83.10	83.10	83.09	82.86	78.78	78.69
$T_{max}(ID3)$ ($^{\circ}\text{C}$)	75.68	75.68	75.67	75.44	69.35	69.25
$T_{max}(ID4)$ ($^{\circ}\text{C}$)	71.99	71.99	71.98	71.75	68.66	68.56
$T_{max}(ID5)$ ($^{\circ}\text{C}$)	69.13	69.13	69.12	68.95	68.69	68.57
$\Delta T(ID1)$ ($^{\circ}\text{C}$)	3.92	3.93	3.94	3.87	2.43	0.34
$\Delta T(ID2)$ ($^{\circ}\text{C}$)	2.30	2.31	2.32	2.24	0.42	0.38
$\Delta T(ID3)$ ($^{\circ}\text{C}$)	2.03	2.05	2.06	1.98	0.12	0.07
$\Delta T(ID4)$ ($^{\circ}\text{C}$)	2.00	2.03	2.04	1.95	0.09	0.04
$\Delta T(ID5)$ ($^{\circ}\text{C}$)	0.40	0.41	0.43	0.41	0.17	0.06

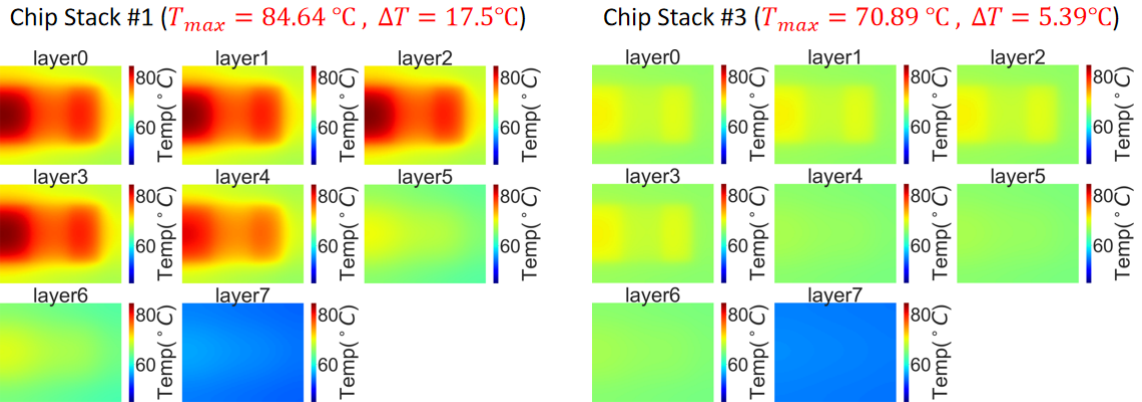


Figure 3-28: Steady-state heat map comparisons of Intel i7 6950 \times (Chip stacks 1 and 3).

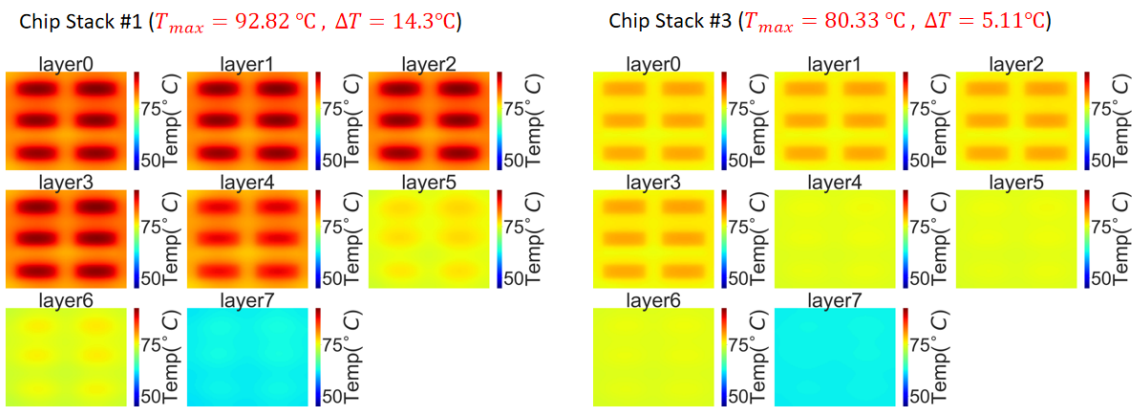


Figure 3-29: Steady-state heat map comparisons of IBM Power9 (Chip stacks 1 and 3).

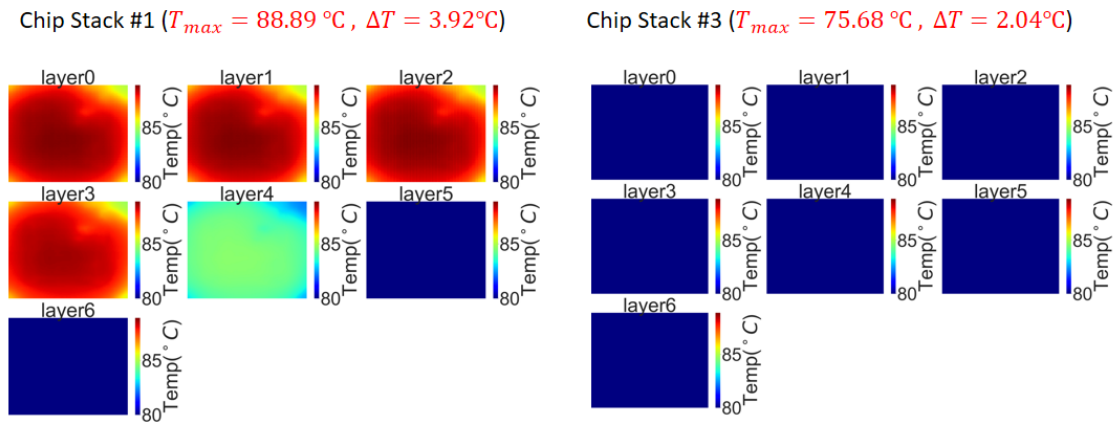


Figure 3-30: Steady-state heat map comparisons of PicoSoC (Chip stacks 1 and 3).

sient power traces by running applications *bt*, *cg*, and *ft* from NAS parallel benchmarks. The transient power traces have been calibrated to the reported TDP from Intel. We run 10 billion instructions for each application and collect power values per 10 million instructions to extract the application power traces. We select different application mapping policies to study the transient thermal behavior of traditional copper heat spreaders and lab-grown diamond heat spreaders. The selected application mapping policies are as follows: (i) we run most power-hungry applications *bt* and *ft* consecutively, and applications are mapped to all ten cores, (ii) we run application *ft* for two iterations. In the first iteration, cores 8 and 9 remain idle, and in the second iteration, cores 4 and 5 are idle, (iii) we run applications *ft* (high power) and *cg* (low power) for two iterations. In the first iteration, each application is mapped on a column of cores, and in the second iteration, applications are mapped as a checkerboard. The transient temperature plots of the Intel i7 6950× silicon layer are shown in Figure 3-31. The maximum transient simulation time of PACT for these real-world high-performance chips with realistic applications is 22 minutes. Plots (A) and (C) indicate that the maximum transient temperature reductions of lab-grown diamond heat spreaders go up to 26.73°C. In addition, the temperature reductions of the diamond heat spreaders depend on the application behavior and application mapping policy. As we see in plots (A), (B), and (C), using different application mapping policies for application *ft* results in different maximum temperatures. For plot (B), leaving cores 4 and 5 idle results in a lower maximum temperature than making cores 8 and 9 idle. As shown in Figures 3-23 and 3-28, cores 4 and 5 are placed at the center of the chip and result in the highest hot spot temperatures. Leaving cores 4 and 5 idle is similar to adding white spaces to the hot spot region to decrease the hot spot temperatures. For plot (C), since the checkerboard mapping policy help spread the lateral heat, the second iteration results in significantly less temperature than the first iteration. For PicoSoC, since the standard-cell design lacks dynamic power traces, we utilize the steady-state power values of PicoSoC and randomly applied

-15% or +15% additional power values for each standard cell and create synthetic transient power traces. The transient temperature plots of PicoSoC are shown in Figure 3-31. We still observe a maximum temperature reduction of 18°C for mobile chip setup. These transient simulation comparison results show that lab-grown diamond heat spreaders' transient cooling performance advantages over traditional copper heat spreaders are even more than the steady-state thermal simulations. For steady-state simulations, we average the power values of the applications in NAS parallel benchmarks, which results in relatively lower steady-state power values than transient power values. In addition, we haven't considered mapping policies in the steady-state simulations. As we see in the transient temperature plots, mapping policies also impact the maximum temperature reductions.

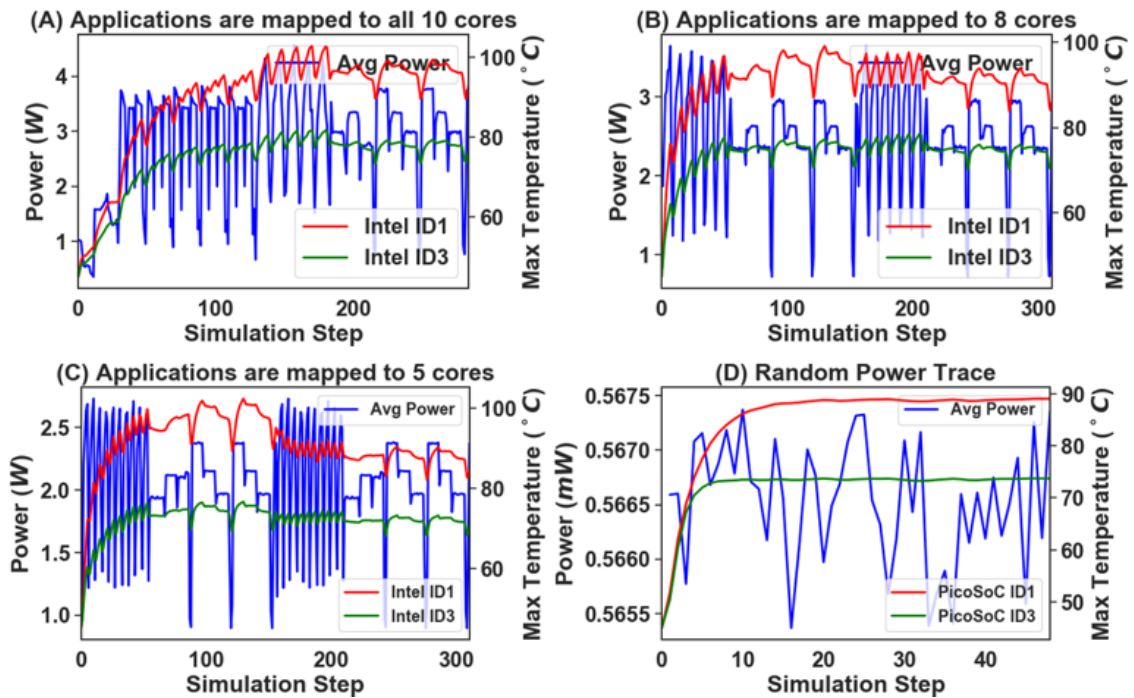


Figure 3-31: Transient temperature plots for Intel i7 6950 \times and PicoSoC.

Parametric Study of Chip Thickness

This subsection studies the cooling performance of lab-grown diamond heat spreaders with different chip thicknesses. We select Intel i7 6950× with chip stacks 1 and 3 as shown in Figure 3-21, and the chip thickness is selected to be 5, 50, 100, 250, 500, and 750 μm . The lab-grown diamond thermal conductivity is set to $7.28(T)^{(-1.42)}\text{MW}/\text{mK}$. We obtain the steady-state power map of Intel i7 6950× by running Sniper and McPAT with the most power-hungry applications *bt* and *ft* from NAS parallel benchmarks and average the transient power trace. The steady-state power map has been calibrated to the reported TDP from Intel. The steady-state maximum temperature results are shown in Figure 3-32. Decreasing the thickness of the silicon layers helps lower the chip stack’s vertical thermal resistance. However, in the meantime, it also prevents spreading the lateral heat across the silicon layer. For chip stack 1, the thick TIM layers dominate the vertical thermal resistance, and varying the silicon layer thickness does not affect the maximum temperature much. Whereas for chip stack 3, diamond heat spreaders have lower thermal resistance and are able to be directly bonded to the silicon. We observe a maximum temperature reduction of 4.62°C (thickness = 5 μm vs. 750 μm) by lowering the silicon layer thickness.

We then conduct a parametric study of silicon layer thickness for transient simulations of Intel i7 6950× chip stacks 1 and 3. We run most power-hungry applications *bt* and *ft* consecutively, and applications are mapped to all ten cores. The transient power traces have been calibrated to the reported TDP from Intel. We show the transient simulation parametric study results in Figures 3-33 and 3-34. We observe a similar trend as the steady-state thickness parametric study. For chip stack 1, decreasing the thickness of the silicon layer does not affect the maximum temperatures because of the tradeoff between vertical and lateral thermal resistance. Whereas for chip stack 3, decreasing the thickness of the silicon layer results in a maximum temperature reduction of 6.53°C (thickness = 5 μm vs. 750 μm). Based on the steady-state and transient parametric studies of the silicon

layer thickness, we show that using a thinner silicon layer achieves an even better cooling performance than a thick silicon layer for lab-grown diamond heat spreaders.

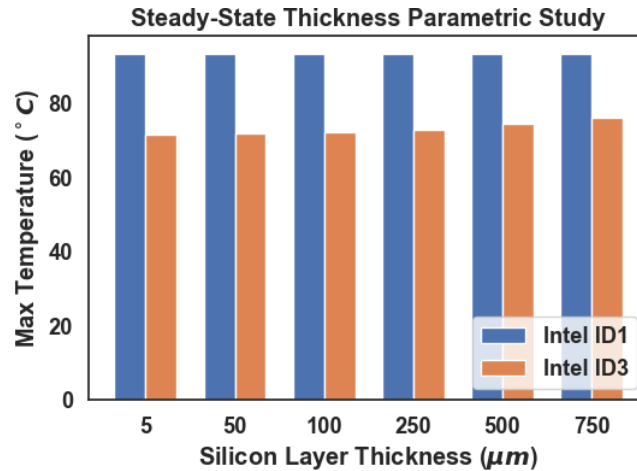


Figure 3-32: Steady-state silicon layer thickness parametric study results for Intel i7 6950×.

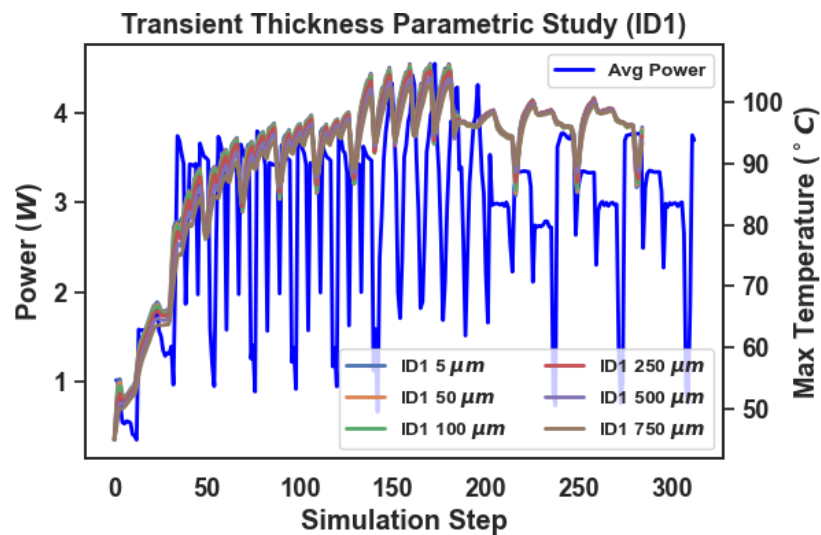


Figure 3-33: Intel i7 6950× chip stack 1 silicon layer thickness transient parametric study results.

Parametric Study of Cooling Packages

Next, we study the cooling performance of lab-grown diamond heat spreaders with different heat sinks. We select Intel i7 6950× with chip stacks 1 and 3 as shown in Figure 2, and the

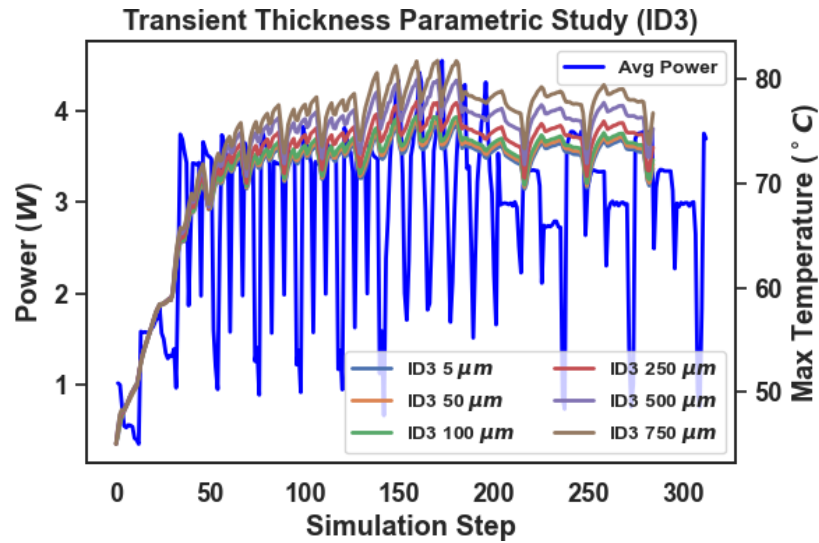


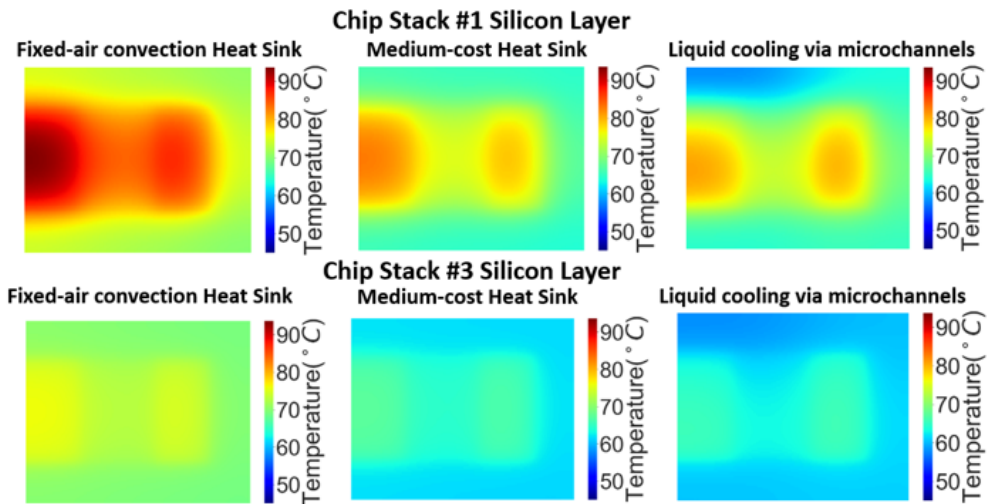
Figure 3-34: Intel i7 6950× chip stack 3 silicon layer thickness transient parametric study results.

heat sink is selected to be the fixed-air convection heat sink, single-phase liquid cooling via microchannels, and medium-cost heat sink adopted from HotSpot (Skadron et al., 2003). We set the air convection HTC to $30 \text{ KW}/\text{m}^2\text{K}$ for the fixed-air convection heat sink. We set the size and thickness of the medium-cost heat sink to $0.4 \times 0.4 \text{ mm}^2$ and 1 mm , respectively. The heat sink is made of copper. The convection resistivity and heat capacity are set to $0.21 \text{ K}/\text{W}$ and $140.4 \text{ J}/\text{K}$. We calculate the convection resistivity based on the air convection HTC of $30 \text{ KW}/\text{m}^2\text{K}$, which is the same as the fixed-air convection heat sink. For liquid cooling via microchannels, the selected material properties are shown in Table 3.13. The selected coolant velocity and the Reynolds number indicate the type of fluid flow is laminar. We obtain the steady-state power map of Intel i7 6950× by running Sniper and McPAT with the most power-hungry applications *bt* and *ft* from NAS parallel benchmarks and average the transient power trace to represent the steady-state power map. The steady-state power map has been calibrated to the reported TDP from Intel.

We show the steady-state silicon layer heat maps in Figure 3-35 and parametric study results in Figure 3-36. By replacing the fixed-air convection heat sink with a medium-cost

Table 3.13: Liquid cooling via microchannels material properties.

Coolant	Water
Thermal resistivity	1.647 mK/W
Specific heat capacity	4.181 MJ/m^3K
Inlet temperature	27°C
Fluid density	998 Kg/m^3
Dynamic viscosity	0.000889 $Pa \cdot s$
Coolant velocity	0.5, 1.0, 1.5, 2.0, 2.6 m/s
Reynolds number	37.4, 74.8, 112, 195
Number of microchannels	146
Microchannel width	50 μm
Wall width	50 μm
Wall material	Silicon
Microchannel Height	100 μm
Microchannel Hydraulic Diameter	66.67 μm

**Figure 3-35:** Steady-state heat maps for parametric study of cooling packages. Liquid flow velocity is set to 2.6 m/s .

heat sink, we observe a maximum temperature reduction of 9.66°C (ID1_Fixed_Air vs. ID1_Medium_Cost). This is mainly because the size of the heat sink is larger than the chip stack and therefore enhances the lateral heat transfer. For liquid cooling via microchannels, as the liquid flow velocity increases, liquid cooling via microchannels becomes the best

cooling package with a maximum temperature reduction of 12.71°C (ID1_Fixed_Air vs. ID1_Liquid @ 2.6 m/s). However, when considering the thermal gradients, the medium-cost heat sink performs better than liquid cooling via microchannels as shown in Figure 3-35. Since liquid absorbs heat as it flows along the channel, the temperature difference between the inlet and outlet is one of the major reasons for the high thermal gradient. Another reason is the thermal resistivity difference between the liquid and wall, which causes the high lateral thermal gradient compared to the medium-cost heat sink. We use

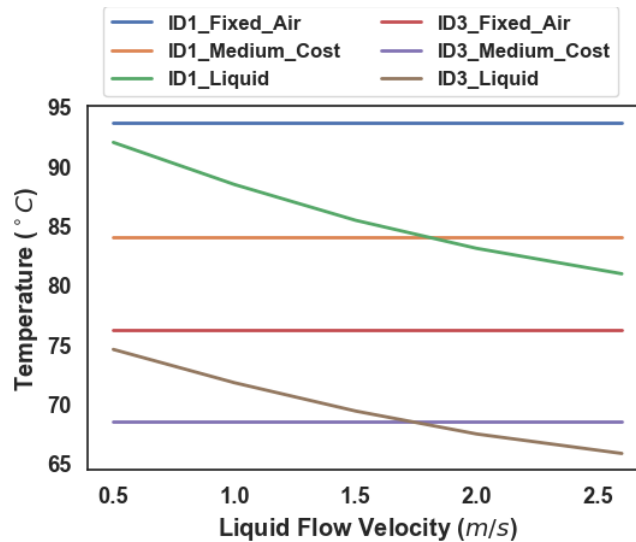


Figure 3-36: Intel i7 6950× silicon layer steady-state cooling package parametric study results (maximum temperature).

the same setup for the three types of aforementioned cooling packages for the transient parametric study. We run most power-hungry applications *bt* and *ft* consecutively, and applications are mapped to all ten cores to obtain the transient power traces. The transient power traces have been calibrated to the reported TDP from Intel. The transient temperature plots are shown in Figure 3-37. We observe a higher maximum temperature reduction of liquid cooling via microchannels against the other two heat sinks than steady-state results. The maximum temperature reduction against the fix-air convection heat sink is 14.13°C . This is due to the high specific heat capacity of the water compared to silicon and copper. In

summary, among these three types of heat sinks, liquid cooling via microchannels provides the highest cooling performance and results in the lowest maximum temperature on-chip.

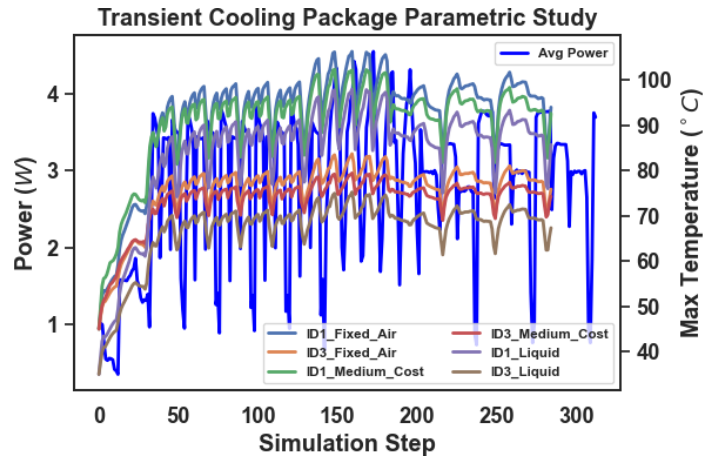


Figure 3-37: Intel i7 6950X silicon layer transient cooling package parametric study results.

Chapter 4

Modeling Emerging Cooling Methods via Machine Learning

4.1 Introduction

Two-phase cooling with VCs is attractive as it offers many advantages over the other techniques: (i) it reduces thermal gradients, (ii) the evaporator in VCs removes heat passively and saves pumping power (in contrast to liquid cooling via microchannels), and (iii) it has a higher cooling efficiency (Thome, 2010; Bulut et al., 2019). In this technique, the phase change from liquid to vapor occurs inside an enclosure called VC. The bottom surface of the VC has a porous wick that sustains thin-film evaporation supplied by passive, capillary-driven flow (Bulut et al., 2019).

Having fast and accurate thermal models is essential for processors to enable power-efficient cooling optimization. Researchers have developed fast models for various cooling methods, including liquid cooling via microchannels and hybrid cooling (of liquid cooling and TEC) (Sridhar et al., 2014; Kaplan et al., 2017). To select and optimize a cooling solution for a given chip and power profile, a fast thermal modeling approach is needed for two-phase VCs with micropillar wick evaporators. Simulations for two-phase VCs are typically carried out using CFD modules in COMSOL and ANSYS (e.g., (Bulut et al., 2019)). However, these tools are computationally expensive and experience long solution times along with large memory requirements (Yuan et al., 2019b). These limitations make CFD tools unsuitable for modeling the cooling technique and realistic processor architectures and applications.

This chapter presents modular CTMs of two-phase VCs with micropillar wick evaporators and two-phase VCs with hybrid wick evaporators to enable speedy and accurate steady-state and transient analysis of two-phase VCs cooling on realistic chip designs. We also introduce an ML-based temperature-dependent HTC simulation framework to model two-phase cooling technologies for processor cooling and discuss an ML and simulation-based methodology to predict accurate thermal maps based on readings from on-chip thermal sensors.

4.2 Two-Phase VCs with Micropillar Wick Evaporators CTM

4.2.1 Background on VCs

The schematic of a VC is shown in Figure 4-1 (a). On the bottom side of VC, there is an evaporator that consists of a thin porous wick. The evaporator is placed directly on top of the heat source (i.e., the processor). As the saturated coolant flows within the porous wick, the coolant absorbs the heat generated by the chip and evaporates. Above the VC, a condenser (e.g., a heat sink) condenses the saturated vapor back to the liquid phase. The condensed liquid is recirculated in the VC by the additional wicking structures along its sidewalls (Bulut et al., 2019). Fabrication and implementation details of VCs can be found in previous works (Hsieh et al., 2012; Bulut et al., 2019). VCs have been shown to achieve significant cooling performance on electronic devices (Bulut et al., 2019) and are already being used as cooling solutions for CPUs, and GPUs (Bulut et al., 2019). Two primary metrics determine the performance in VCs: (i) HTC, and (ii) dry-out heat flux. HTC is the rate of heat transfer per unit temperature difference between the evaporator and the environment. Dry-out heat flux refers to the thermal limit of a two-phase device, beyond which the coolant ceases to exist in two phases and instead is found in only vapor phase. Micropillar wick evaporators have improved cooling efficiency and enhanced dry-out heat flux due to their high capillary pumping budget and extended menisci evaporation area (Wei

et al., 2018; Adera et al., 2016).

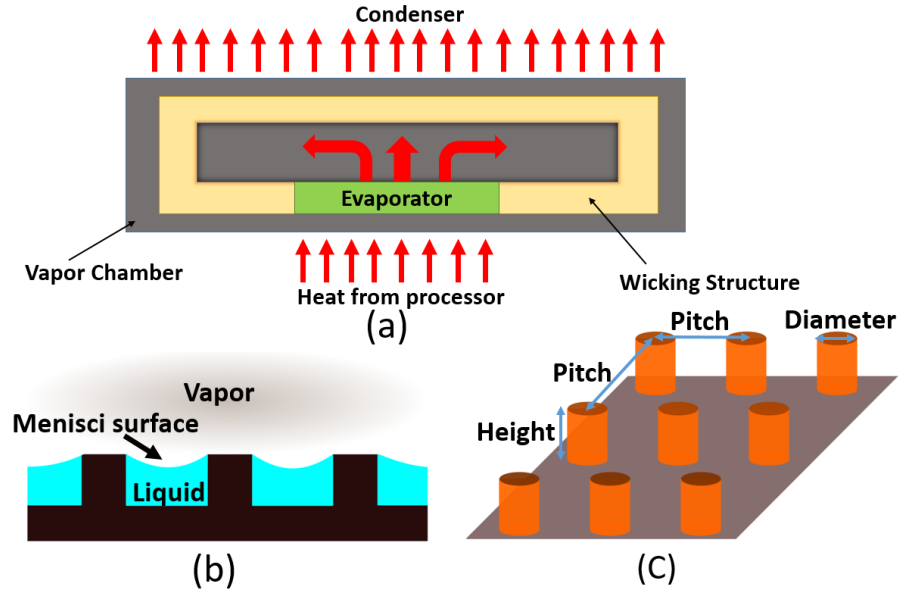


Figure 4-1: (a) VC structure view, (b) micropillar wick side view, and (c) micropillar wick side view overall view.

4.2.2 Compact Modeling Methodology

The entire system including the VC is divided into small grid cells. The default grid cell is shown in Figure 4-2 (a). This figure shows that the virtual temperature node, which represents the temperature of the grid cell, is placed on the bottom surface. The micropillar wick evaporator is modeled as a separate layer placed directly above the processing layer. In Figure 4-2 (b), we demonstrate how the grid cells of the two layers are connected in the model. We assume that inside the VC there is only saturated vapor and all the liquid is contained in the wicking structures and evaporator as shown in Figure 4-1 (b). To model the saturated vapor conditions, we place an additional virtual temperature node on top of each micropillar wick layer grid cell as shown in Figure 4-2 (b). The temperature of this node is set to the saturated temperature of the coolant, T_{sat} , and therefore, depends on coolant properties and pressure inside the VC. The micropillar wick layer along with T_{sat} nodes

represent the VC. Since we assume that T_{sat} is maintained at a constant temperature, we do not need to model a condenser.

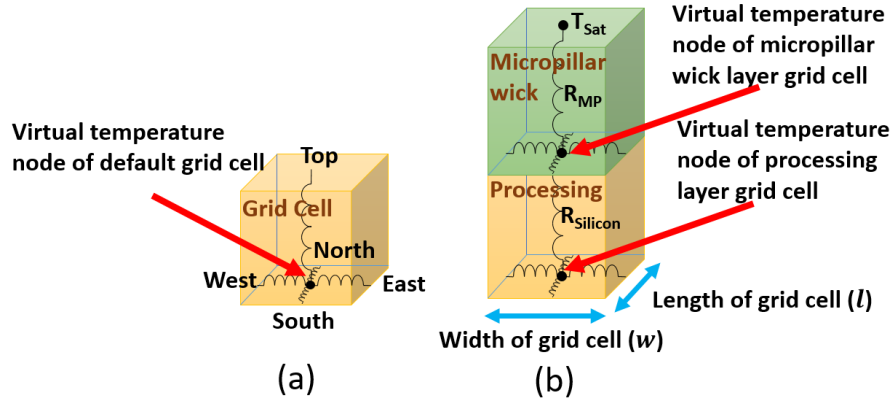


Figure 4-2: (a) Default grid cell and (b) proposed grid cells for modeling two-phase VCs with micropillar wick evaporators.

We assign silicon properties to the processing layer grid cells and represent the lateral and vertical thermal resistance, respectively, of each cell as $R_{Silicon}$ as shown in Figure 4-2 (b). In the micropillar wick layer, determining the vertical thermal resistance of each grid cell is complicated because the coolant exists in two phases. We use a previously established relationship between the HTC and the thermal resistance of a grid cell to represent the vertical thermal resistance of a micropillar grid cell, R_{MP} (Sridhar et al., 2014; Yuan et al., 2019b). Micropillar wick HTC is highly dependent on the coolant, VC pressure, and micropillar wick geometry (micropillar height, diameter, and pitch as shown in Figure 4-1 (c)) (Bulut et al., 2019). We use a COMSOL model to extract the HTCs of a wide range of micropillar geometries, coolants, and VC pressures. This COMSOL model is detailed in a prior work (Vaartstra et al., 2019). In this COMSOL model, the authors separate the CFD simulation into the fluid and heat transfer domains. By coupling these two domains, they iteratively obtain the temperature distribution. They use the Young-Laplace equation to relate the curvature of the liquid-vapor interface to local pressure. In addition, they also use Darcy's law and a volumetric loss function to model fluid flow in a uniform porous

medium and the evaporative flux, respectively. Furthermore, they parametrically derive the permeability and HTC for each micropillar wick geometry. The extracted HTCs are stored in HTC lookup tables for various coolants and geometries. For simplicity, we assume a flat evaporation surface in our model instead of a menisci evaporation surface as shown in Figure 4.1 (b). The flat evaporation surface is a conservative assumption that is widely used to define and simplify the liquid-vapor interface (Wei et al., 2018). This assumption enables us to employ a uniform HTC across the micropillar wick layer.

Compared to other state-of-the-art compact modeling methodologies (Sridhar et al., 2014; Sridhar et al., 2013b; Kaplan et al., 2017), the distinctions of our CTM are as follows: i) our proposed CTM models two-phase cooling in VCs, a passive cooling technique that requires no cooling power on the evaporator side; ii) we place the virtual temperature node at the bottom of the VC grid cells to model heat transfer happened on the evaporator and since there is no pumping power at the evaporator side, there is no need for voltage-controlled current sources; and iii) since the HTC is stored in a lookup table, our modeling methodology can be generalized to model different two-phase cooling devices.

4.2.3 Dry-out Heat Flux Analytical Model

One major concern while designing VCs is to prevent dry-out. Dry-out heat flux for square chips is defined in Equations (4.1) and (4.2) (Adera et al., 2016). The coolant and micropillar parameters are listed in Table 4.1.

$$q''_{dry-out} = (40/3)\psi M \cos\theta_{rec} \quad (4.1)$$

$$M = \frac{\sigma_{lv}\rho_l h_{lv}}{\mu_l} \quad (4.2)$$

ψ is a dimensionless function of micropillar geometry that lumps the effect of the geometry on heat transfer capacity (Adera et al., 2016). M is a figure of merit for the coolant. We use the above equations to calculate the dry-out heat flux, $q''_{dry-out}$. Since the dry-out heat

flux is sensitive to micropillar geometry and chip dimensions (Bulut et al., 2019), we next perform parametric studies on the impact of the chip dimensions and micropillar geometry on dry-out heat flux.

Table 4.1: Coolant and micropillar parameters.

h	Height of micropillar
d	Diameter of micropillar
i	Pitch of micropillar wick
P_{bg}	Background power density
P_{hs}	Hot spot power density
ρ_l	Liquid density
h_{lv}	latent heat of vaporization
μ_l	Dynamic viscosity
q''	Heat flux
$q''_{dry-out}$	Dry-out heat flux
θ_{rec}	Receding contact angle for fluid-solid pair
p	Pressure
κ	Permeability of the wick
T_{sat}	Saturated temperature of the coolant
σ_{lv}	Surface tension
ΔT	Thermal gradients across the chip
MG_{opt}	Optimal micropillar geometry
$P_{cooling}$	Cooling power
T_{hs}	Hot spot temperature
T_{limit}	User-defined temperature limit
u	Liquid flow velocity
I	TEC current

4.2.4 Parametric Study

Recall that high HTC and high dry-out heat flux deliver a better cooling performance of two-phase cooling in VCs. To understand the relationships among the HTC, dry-out heat flux, micropillar geometries, and chip sizes, we perform parametric studies using a COMSOL model (Vaartstra et al., 2019). Across all studies, we use a coolant, R134a, at T_{sat} of 50°C under 13.2 bar pressure and vary the chip size from 4 mm^2 to 100 mm^2 . In the first study, we vary the micropillar height (h) from $20\ \mu\text{m}$ to $70\ \mu\text{m}$, with diameter (d) and pitch (i) set to $10\ \mu\text{m}$ and $20\ \mu\text{m}$, respectively. Figure 4-3 (a) shows the inverse relationship between HTC and dry-out heat flux as micropillar height changes. The dry-out heat flux increases with increasing micropillar height because the wick becomes more permeable,

which lowers the viscous resistance to capillary flow. On the other hand, HTC decreases with increasing micropillar height since the liquid film becomes thicker, thus increasing conduction resistance. In the second study, we vary i from $60 \mu\text{m}$ to $150 \mu\text{m}$, with h and d set to $55 \mu\text{m}$ and $10 \mu\text{m}$, respectively. The results of this simulation are shown in Figure 4.3 (b), in which we observe that as the pitch increases, both dry-out heat flux and HTC decrease. In this particular regime, the loss of capillary pumping budget due to increasing the pitch is more significant than the increase in permeability. Therefore, increasing the pitch leads to an earlier dry-out. The HTC decreases with the increasing pitch because the solid fraction ($c = \frac{\pi i}{4} \left(\frac{d}{i}\right)^2$) diminishes, forcing more heat to conduct through the liquid film. In the third study, we fix h to $50 \mu\text{m}$ and i to $30 \mu\text{m}$, and vary d from $5 \mu\text{m}$ to $17 \mu\text{m}$. We observe that both the dry-out heat flux and HTC increase with increasing micropillar diameter. This is because a larger diameter leads to a higher capillary pumping budget that increases the dry-out heat flux. In contrast, the increase in the solid fraction is favorable for conduction, thus enhancing the HTC. Based on the above studies, we see that both the HTC and dry-out heat flux have nontrivial relationships with different geometry parameters. As a result, we need an optimal micropillar geometry to enhance the cooling performance of two-phase VCs.

4.2.5 Validation of the Proposed Model

We model a $2 \times 2 \text{ mm}^2$ chip with a thickness of $100 \mu\text{m}$ in COMSOL and HotSpot to validate the accuracy of our proposed CTM. We run two sets of simulations in COMSOL: (i) processing layer with a uniform power density and (ii) processing layer with a non-uniform power density with a $500 \times 500 \mu\text{m}^2$ hot spot placed at the center. Each simulation set has three different micropillar wick geometries and two coolants: water and R134a. In these validation experiments, since water has a better HTC than R134a, to ensure the maximum temperatures are less than 85°C , we select higher power densities for water and lower power densities for R134a. In the experiment with uniform power density, we set the power

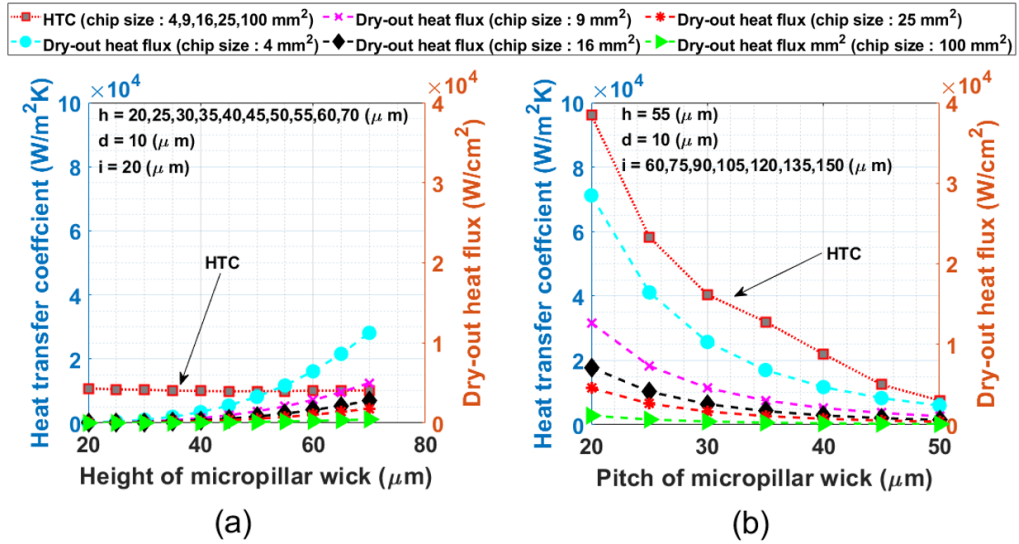


Figure 4-3: Parametric study for different micropillar wick geometries. Dry-out heat flux is shown on the right axes.

density equal $100 W/cm^2$ for water and $20 W/cm^2$ for R134a. In the non-uniform power density simulations, we set the background power density to $50 W/cm^2$ for water and $20 W/cm^2$ for R134a. The hot spot power density is set to 100, 200, and $300 W/cm^2$ for water and 25, 50, and $75 W/cm^2$ for R134a. To prevent extremely high chip temperatures, we reduce the T_{sat} of water to $50^\circ C$ by setting the pressure inside VC to 0.124 bar. As for R134a, we set its T_{sat} to $50^\circ C$ under 13.2 bar pressure. For each COMSOL simulation, we use 592 nodes to simulate the fluid domain and 1106 nodes to simulate the heat transfer domain. Among all the simulation cases, it takes a minimum of 4 iterations to converge and finish the simulation. We model the same chip in HotSpot using the grid model and use 64×64 grids to compute the steady-state temperatures. The simulation time of the COMSOL CFD model is 45 seconds, while it only takes 0.21 seconds to simulate our proposed CTM. Table 4.2 compares the temperatures obtained in the above simulations. For both uniform and non-uniform simulations, the proposed model achieves high accuracy with both the maximum and average errors less than $0.5^\circ C$ for water and $1.25^\circ C$ for R134a while achieving a speedup of $214 \times$ when compared to COMSOL CFD simulations. Typical accuracies for

CTMs of various cooling technologies range from 89.9% to 97.3% (Kaplan et al., 2017; Sridhar et al., 2014; Sridhar et al., 2013b). Our proposed CTM provides a 98.5% accuracy which is similar to the approaches mentioned above. These simulations show that our model significantly reduces simulation time with only a small tradeoff in accuracy.

Table 4.2: Comparison between proposed CTM and COMSOL.

Simulations	Coolant	P_{bg}	P_{hs}	{h,d,i}	Avg error ($^{\circ}C$)	Max error ($^{\circ}C$)
Uniform Power	Water	100	100	{30,12,36}	0.20	0.19
		100	100	{40,16,48}	0.22	0.21
		100	100	{50,20,60}	0.25	0.25
	R134a	20	20	{30,12,36}	0.65	0.71
		20	20	{40,16,48}	0.75	0.77
		20	20	{50,20,60}	0.80	0.82
Non-uniform Power	Water	50	100	{30,12,36}	0.23	0.23
		50	200	{40,16,48}	0.31	0.26
		50	300	{50,20,60}	0.49	0.45
	R134a	20	25	{30,12,36}	0.99	1.1
		20	50	{40,16,48}	1.02	1.12
		20	75	{50,20,60}	1.04	1.24

Table 4.3: Structural properties and simulation parameters.

Processing layer thickness	750 μm
Microchannel height	200 μm
Microchannel width (same as wall width)	50 μm
TEC layer thickness	100 μm
Packaging layer (bulk silicon)	40 μm
higher power density chip size	20 \times 20 mm^2
Intel SCC core size	1.129 mm^2
lower power density chip size	18 \times 14.1 mm^2

4.2.6 Cooling Performance Evaluation

To evaluate the cooling performance of two-phase VCs with micropillar wick evaporators, we run thermal simulations to compare its cooling performance and cooling power with liquid cooling via microchannels and microchannel-based two-phase cooling. The simulated chip is 20 \times 20 mm^2 large with a 500 \times 500 μm^2 hot spot placed at the center. The background power density is set to 50 W/cm^2 , and the hot spot power density varies from 100 to 2000 W/cm^2 . For a fair comparison, we select water as a coolant with a saturated temperature of 50 $^{\circ}C$ (pressure = 0.124 bar). We vary the liquid flow velocity from 0.5 to 2.6 m/s (Sridhar et al., 2014) and mass flow velocity from 100 to 560 kg/m^2s (Kandlikar,

2002). We sandwich the liquid microchannel layer between the processing and packaging layers. Structural properties of liquid cooling via microchannels and microchannel-based two-phase cooling are shown in Table 4.3. Table 4.4 shows the optimal micropillar geometries selected by the optimization flow along with the estimated pumping power for liquid-cooling via microchannels and microchannel-based two-phase cooling. Figure 4.4 shows the simulation results when liquid flow velocity is set to 2.6 m/s and mass flow velocity is set to $560 \text{ kg/m}^2\text{s}$. Two-phase VCs achieve lower hot spot temperature when compared to liquid cooling via microchannels. Since microchannel-based two-phase cooling is an active microfluidic two-phase cooling method, it provides higher hot spot temperature reductions than two-phase VCs. Most importantly, two-phase VCs provide higher reductions on thermal gradients by up to 11.78°C in comparison to liquid cooling via microchannels, and 1.5°C in comparison to microchannel-based two-phase cooling, without additional pumping power on the evaporator side.

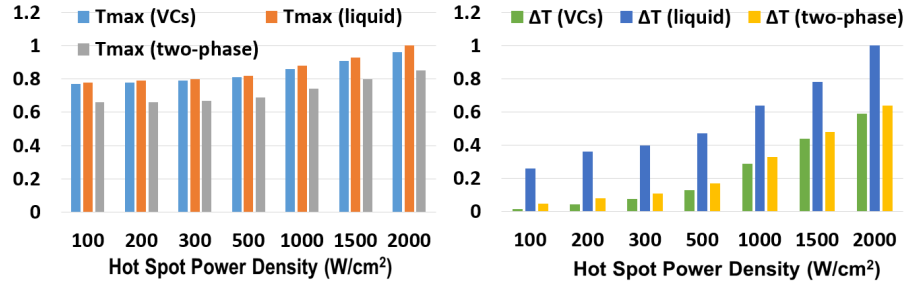


Figure 4.4: Comparison of cooling performance of two-phase VCs with micropillar evaporators (VC), liquid cooling via microchannels (liquid), and microchannel-based two-phase cooling (two-phase) when flow velocity = 2.6 m/s and mass flow velocity = $300 \text{ kg/m}^2\text{s}$. Results are normalized to liquid cooling when $P_{hs} = 2000 \text{ W/cm}^2$.

Table 4.4: Optimal geometries (h, d, i) of two-phase VCs and estimated pumping power of liquid cooling via microchannels and microchannel-based two-phase cooling ($G = 560 \text{ kg/m}^2\text{s}$).

$P_{hs} \text{ (W/cm}^2\text{)}$	100	200	300	500	1000	1500	2000	
$MG_{opt} \text{ (}\mu\text{m)}$	20,10,5	25,10,5	30,10,5	30,10,5	35,10,5	45,10,5	45,10,5	
$u \text{ (m/s)}$	0.5	1	1.5	2.6	$G \text{ (kg/m}^2\text{s)}$	100	300	560
$P_{pump} \text{ (W)}$	0.17	0.66	1.5	4.5	$P_{pump} \text{ (W)}$	0.2	0.41	1.14

4.3 Two-Phase VCs with Hybrid Wick Evaporators CTM

As we discussed in Section 4.2.4, the heat removal ability of the VC is often dominated by the evaporator (Bulut et al., 2019). An evaporator with a higher HTC is desired to reduce the thermal resistance of the VCs. However, such high-HTC evaporators often suffer from low critical dry-out heat flux (Bulut et al., 2019; Ju et al., 2013). These two metrics are typically conflicting with each other, and it is challenging to maximize HTC while enhancing dry-out heat flux (Bulut et al., 2019). In this section, we focus on a hybrid wick evaporator (of nanoporous membrane and microchannels) as shown in Figure 4-5 that improves both HTC and dry-out heat flux. The microchannel and membrane geometries can be varied independently so as to enhance the permeability of the microchannels and the heat transfer from highly conductive solids (the substrate, microchannels, and nanoporous membrane) to the liquid-vapor interface.

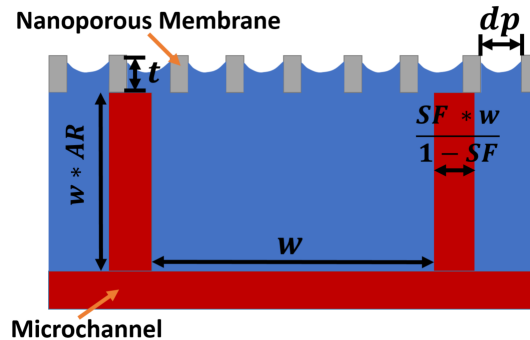


Figure 4-5: A hybrid wick evaporator cross-section view.

4.3.1 Compact Modeling Methodology

We abstract both the nanoporous membrane and the microchannel layers into a hybrid wick layer to build a CTM. The whole chip stack is shown in Figure 4-6 (a). We divide the whole chip into grids. For the processing layer, the grid cell structure is shown in Figure 4-6 (b). Thermal resistance along the north, south, east, west, and vertical directions are represented using silicon properties, i.e., $R_{silicon}$. As for the hybrid wick layer, the grid

cell is shown in Figure 4-6 (c). We represent lateral thermal resistances using $R_{silicon}$ and the vertical thermal resistance, R_{hybrid} , stands for the inverse of the heat conduction from the hybrid wick to the saturated vapor. We add an additional virtual temperature node on top of the hybrid wick grid cell to represent the saturated vapor. We consider steady-state and a predetermined VC pressure. In this way, we do not need to model the heat sink on top of the VC. Instead, we use a previously established relationship between HTC and thermal resistance to define R_{hybrid} (Sridhar et al., 2014; Yuan et al., 2019a). In addition, we assume the VC itself only contains saturated vapor at a constant temperature (Vaartstra et al., 2019; Lu et al., 2016). From the COMSOL model discussed in the next section, we extract HTC correlations for various nanoporous membranes and microchannel geometries. For a specific hybrid wick geometry, we use its corresponding HTC correlation to determine the R_{hybrid} value.

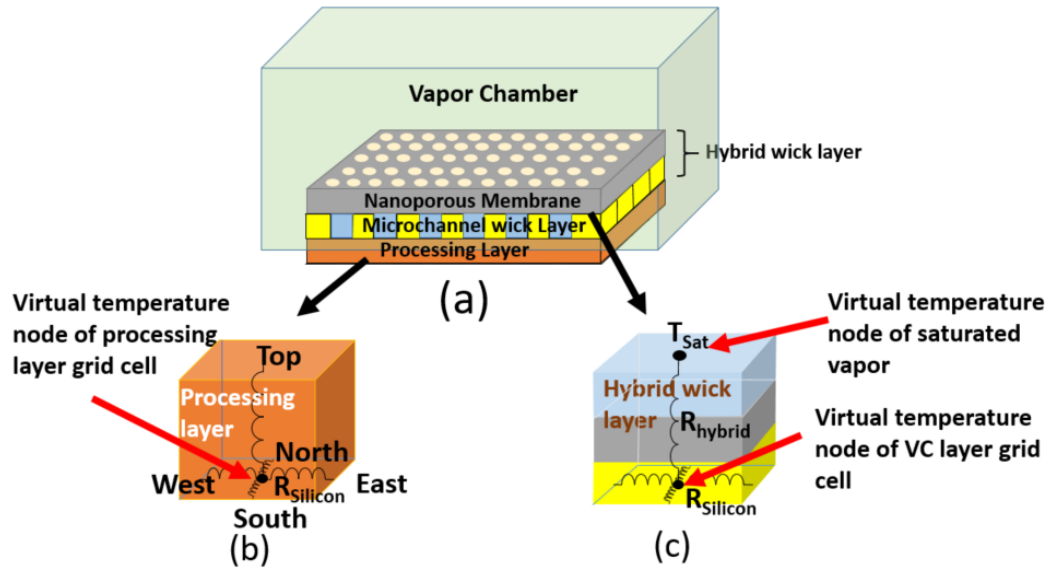


Figure 4-6: (a) The chip stack of the processing layer and two-phase VCs with hybrid wick evaporators, (b) processing layer grid cell, and (c) hybrid wick layer grid cell.

4.3.2 COMSOL Model

We determine the resistance of the hybrid wick layer by solving for its effective HTC via finite element calculations using COMSOL. The simulated domain consists of a nanoporous silicon membrane, a silicon microchannel, and the coolant. Due to device symmetry, we only need to simulate one microchannel, and we can additionally limit the domain to two dimensions by considering the channels to be infinitely long. We neglect convection in the liquid phase since the Peclet number is small (Lu et al., 2016). Constriction resistance between the substrate and the hybrid wick is accounted for by including $1 \mu\text{m}$ of the silicon substrate in the simulation domain. The thermal conductivities of the silicon and the working fluid, as well as the resistance to evaporation posed by the liquid-vapor interface, are temperature-dependent properties. Thus, the effective HTC of the hybrid wick is dependent on temperature in addition to geometry and the vapor conditions.

We calculate the HTC by imposing an inward heat flux (q'') at the bottom of the domain and an evaporative boundary condition at the top of the membrane, except where the membrane is supported by the microchannel, which is set as an insulated boundary. The evaporative boundary condition is modeled using a numerical solution to the Boltzmann transport equation, which governs the flux of vapor molecules from the liquid-vapor interface to the far-field vapor (Sone, 2000). We utilize the numerical Direct Simulation Monte Carlo (DSMC) data and prescribed boundary conditions from recent work to model the evaporative boundary condition (Lu et al., 2019). We extract the average temperature at the substrate-hybrid wick interface (T_b) from the temperature distribution determined by COMSOL, from which the effective HTC of the hybrid wick is calculated by Fourier's Law ($\text{HTC} = q'' / (T_b - T_{Sat})$), where T_{Sat} is the temperature of the far-field vapor. We obtain HTC as a function of T_b by imposing a range of heat fluxes on the hybrid wick for each fixed set of geometries and vapor conditions. This COMSOL model has been validated against the experimental results presented in a recent work (Hanks et al., 2018).

4.4 An ML-Based Thermal Simulation Framework for Emerging Two-Phase Cooling Technologies

Two-phase cooling thermal models are usually designed by calculating the temperature-dependent HTC (Sridhar et al., 2013b; Thome, 2010; Lu et al., 2016; Vaartstra et al., 2019). In these models, pre-computed temperature-dependent HTC correlations embedded in the simulation framework are functions of temperature and cooling parameters (e.g., coolant type, flow velocity, saturation temperature, and structural parameters such as microchannel width and height, micropillar height, diameter, and pitch (Sridhar et al., 2013b; Vaartstra et al., 2019; Yuan et al., 2019a)). The HTC correlations are derived either based on in-house prototypes or using CFD modules in COMSOL and ANSYS (Vaartstra et al., 2019; Bulut et al., 2019). HTC correlations based on prototypes are generally not applicable to the same cooling method with different cooling parameters or will likely result in accuracy loss (Sridhar et al., 2013b). Commercial simulation tools (e.g., COMSOL and ANSYS), on the other hand, are computationally expensive and experience long design and simulation times as well as large memory requirements. To enable cooling design exploration and optimization, there is a need for a fast and generalized temperature-dependent HTC simulation framework that applies to a wide range of cooling parameters for the same two-phase cooling technology while maintaining the desired accuracy.

We introduce an ML-based temperature-dependent HTC simulation framework to model two-phase cooling technologies for processor cooling. This framework enables fast and accurate simulations with a wide range of cooling parameters for the same two-phase cooling technology. To demonstrate our proposed framework's speedup and accuracy, we integrate the CTM for two-phase VCs with hybrid wick evaporators (of nanoporous membrane and microchannels) into our proposed simulation framework.

4.4.1 A Temperature-Dependent HTC Simulation Framework

The HTC of the hybrid wick evaporator is highly dependent on the temperature distribution (Lu et al., 2016). To precisely calculate the temperature distributions of the hybrid wick layer and the processing layer, we implement a temperature-dependent HTC simulation framework for the two-phase VCs with hybrid wick evaporators CTM. The simulation flow is shown in Figure 4.7. We divide the simulation flow into two domains: (i) the “update HTC” domain, which takes the HTC correlation from a lookup table generated by the COMSOL model for a specific nanoporous membrane and microchannel geometry, and then updates the HTC value for each hybrid wick layer grid cell, based on both the HTC correlation and temperature distribution, and (ii) the “heat conduction” domain, which takes the HTC distribution, calculates the R_{hybrid} for the corresponding hybrid wick layer grid cell, and then carries out the thermal simulation to generate a new temperature distribution to pass it to the “update HTC” domain. The simulation framework iteratively solves for the HTC and temperature distributions until the temperature distribution converges (temperature difference of $< 0.1^{\circ}\text{C}$).

4.4.2 An ML-Based Temperature-Dependent HTC Simulation Framework

In the original temperature-dependent HTC simulation framework, the HTC correlation of each hybrid wick geometry within a valid range is generated from COMSOL simulations. The naming convention and valid range of hybrid wick geometry parameters are shown in Table 4.5 (Lu et al., 2016). Figure 4.5 shows the structure of a hybrid wick. We store these HTC correlations in a lookup table. Generating a 4096-entry lookup table using COMSOL takes more than 24 hours. If we select 10 cases for each parameter (a total 1 million entries), the generation time is 786 days. In this case, the HTC correlation lookup table pre-computing time for the hybrid wick geometry is the bottleneck of this temperature-dependent HTC simulation framework. In addition, even if we have a

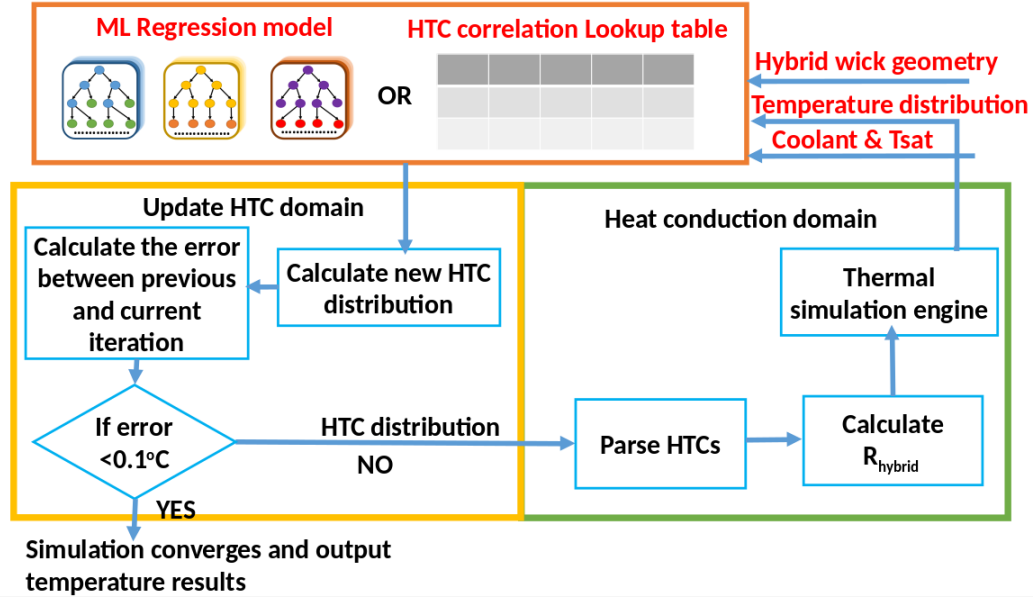


Figure 4.7: Temperature-dependent HTC simulation framework and our proposed ML-based temperature-dependent HTC simulation framework.

finer granularity HTC lookup table for one specific geometry range, we cannot run thermal simulations for a comprehensive range of the hybrid wick geometries. To enable thermal simulations for a wide range of valid hybrid wick geometries, we propose an ML-based temperature-dependent simulation framework as shown in Figure 4.7. We add additional ML regression models for different coolants to replace the HTC lookup table compared to the original temperature-dependent simulation framework. The selection of the ML regression model depends on the two-phase cooling method and the cooling parameters. We select the random forest regression model for two-phase VCs with hybrid wick evaporators. The inputs of the ML regression model are $\{t, dp, \phi, AR, SF, w, T_{Sat} - T_{Cur}, coolant\}$ (see Table 4.5). The output of the ML model is an HTC value. During each iteration, the thermal simulation engine generates a new temperature distribution and passes it to the ML regression model to predict the HTC value based on the temperature distribution, hybrid wick geometry parameters, coolant saturation temperature, and coolant type.

Table 4.5: Hybrid wick geometry parameters and valid range.

Symbol	Parameters	Valid range
t	Nanoporous membrane thickness	250-1000 nm
dp	Membrane pore diameter	50-200 nm
ϕ	Membrane porosity	0.2-0.8
AR	Microchannel aspect ratio	0.5-2
SF	Microchannel wall solid fraction	0.1-0.4
w	Microchannel width	2-8 μm
T_{Sat}	Coolant saturation temperature	50°C
T_{Cur}	Current temperature of the grid	NA

4.4.3 Validation of the ML Model

Before validating the ML-based temperature-dependent HTC simulation framework, we first perform cross-validation (CV) of the ML regression model to show that our regression model accurately predicts HTC for various hybrid wick geometries and coolants. For each hybrid wick geometry parameter range shown in Table 4.5, we select the minimum value, 25 percentile value, 75 percentile value, and the maximum value as our training and testing geometry parameters. There is a total number of 4096 geometries. We use COMSOL to generate the temperature-dependent HTC correlations for these 4096 geometries for three different coolants. For each temperature-dependent HTC correlation, we range $T_{Cur} - T_{Sat}$ from 0 to 40°C with a step of 1°C to generate the golden HTC data. The total HTC data size for all the three coolants is 163840. For each coolant, we do k-fold CV to show that our ML regression model is capable of predicting HTC for arbitrary hybrid wick geometries. We use the training data to train a random forest model with a number of trees equal to 100 for each coolant. We also test with various ML models, including support vector regression (SVR), neural network regression (NNR), decision tree, gradient boosting regression (GDR), etc. Since random forest results in the best accuracy, we only report the regression accuracy results of random forest regression. We show the worst-case scenario results from the k-fold CV for each coolant in Table 4.6. As we observe from the table, our ML model successfully predicts HTC for an extensive selection of valid hybrid wick geometries.

Table 4.6: Worst-case results from the k-fold CV tests. MAE stands for mean absolute error, RMSE stands for root mean square error. The errors are normalized with respect to the golden HTC data.

5-fold				3-fold			
Coolant	MAE	RMSE	R2	Coolant	MAE	RMSE	R2
Water	0.11%	0.16%	99.95%	Water	0.13%	0.19%	99.93%
R245fa	0.41%	0.76%	99.92%	R245fa	1.1%	2.1%	99.91%
R141b	0.17%	0.31%	99.94%	R141b	0.8%	1.2%	99.92%

4.4.4 Validation of the ML-Based Temperature-Dependent HTC Simulation Framework

Next, we perform an accuracy and speedup comparison among the ML-based HTC simulation framework, temperature-dependent HTC simulation framework, and COMSOL model using various chip power profiles and hybrid wick geometries. Both the temperature-dependent HTC simulation framework and ML-based temperature-dependent HTC simulation framework are integrated into PACT. We model a $2\text{ mm} \times 2\text{ mm}$ chip with a thickness of $100\text{ }\mu\text{m}$ in COMSOL, the ML-based framework, and the original temperature-dependent framework. We run two sets of simulations for each floorplan as shown in Figure 4-8: (i) processing layer with a uniform power density, and (ii) processing layer with a non-uniform power density with $500 \times 500\text{ }\mu\text{m}^2$ hot spots. Each simulation set uses three different hybrid wick geometries as shown in Table 4.7 and three different coolants. We train the ML regression model with 4096 geometries described in the previous section. The three validation geometries are excluded from the training set. For uniform power density tests, we use 100 W/cm^2 and 200 W/cm^2 . For non-uniform power density tests, we set the background power density to 50 W/cm^2 and hot spots power density to 100, 500, and 1000 W/cm^2 (Lu et al., 2016). The COMSOL model uses 428 nodes to compute the temperature distribution, while we use $16 \times 16 \times 3$ nodes in our modeling tool to simulate the temperature. We compare the average simulation runtime of COMSOL, temperature-dependent HTC simulation framework, and ML-based temperature-dependent HTC simulation framework. For uniform power density validation tests, the maximum and average errors of the ML-based

temperature-dependent HTC simulation framework are less than 0.46°C and 0.24°C , respectively. The accuracy results for non-uniform validation tests are shown in Figure 4.9. Compared to the COMSOL model, the temperature-dependent HTC simulation framework has a maximum error of 1.34°C with an average speedup of $22\times$. The maximum error and average error of our proposed ML-based temperature-dependent HTC simulation framework are 2.59°C and 0.98°C , respectively and the average speedup is $21\times$. Note that the accuracy results include all of the validation floorplans and geometries. Since we replace the temperature-dependent HTC correlation lookup table with ML regression models, the simulation speedup and accuracy are expected to decrease. However, our proposed model still achieves good accuracy and speedup when compared to the COMSOL model. Most importantly, our proposed ML-based temperature-dependent HTC simulation framework enables accurate thermal simulations with valid and comprehensive geometries.

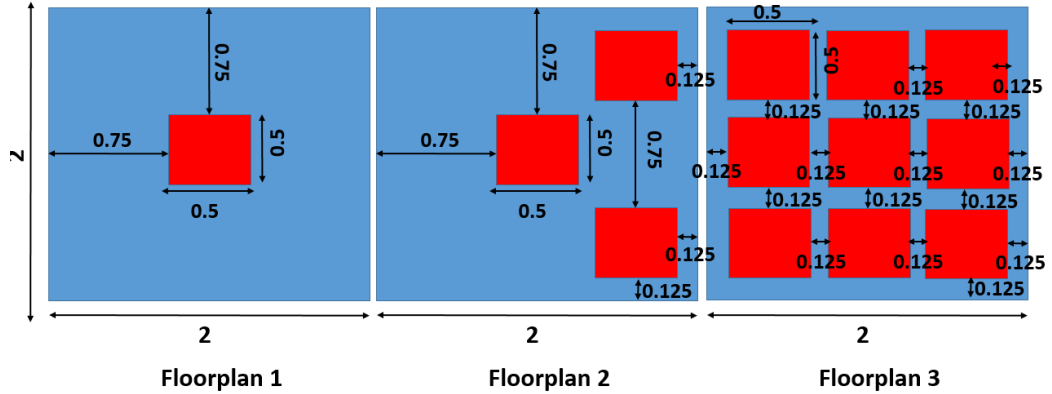


Figure 4-8: Floorplans used in validation. Dimensions are in *mm*.

Table 4.7: Hybrid wick geometries for validation tests.

Geometry	t	dp	ϕ	AR	SF	w
1	450	120	0.4	2	0.25	4
2	300	100	0.2	1	0.2	5
3	700	150	0.45	0.6	0.1	8

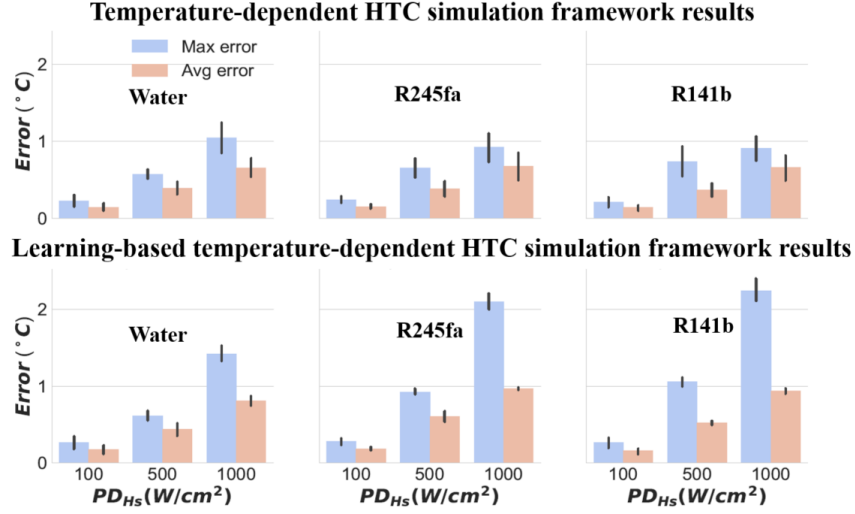


Figure 4-9: Non-uniform power profile maximum and average temperature error validation results. PD_{H_s} stands for hot spot power density.

4.5 Improved ML-Based Simulation Framework for Two-Phase VCs

In Sections 4.3 and 4.4, we propose a ML-based temperature-dependent HTC simulation framework for two-phase cooling technologies and create two-phase VCs with hybrid wick evaporators' steady-state CTM to demonstrate the speedup and accuracy compared to CFD simulation. For the two-phase VCs with hybrid wick evaporators CTM, we assume the VC itself only contains saturated vapor at a constant temperature to simplify modeling the vapor core and condenser. Therefore, the ML-based temperature-dependent HTC simulation framework has not considered the vapor core and condenser simulation. In addition, both the proposed ML-based simulation framework and CTM only support steady-state simulation, whereas dynamic thermal management policies often rely on the chip's transient temperature. In this section, we extend our two-phase VCs with hybrid wick evaporators CTM and ML-based temperature-dependent HTC thermal simulation framework to support thermal simulations for vapor core and condenser. We also introduce modeling of transient temperature behavior for two-phase VCs. The naming conventions of the parameters used in this section are listed in Table 4.8.

Table 4.8: Naming conventions of the parameters.

$R_{silicon}$	Silicon thermal resistance
R_{hybrid}	Hybrid wick vertical thermal resistance
w	Width of the grid cell
l	Length of the grid cell
h	Height of the grid cell
$R_{vapor,lat}$	Lateral thermal resistance of vapor core
$R_{vapor,vert}$	Vertical thermal resistance of vapor core
$R_{cond,lat}$	Lateral thermal resistance of condenser
$R_{cond,vert}$	Vertical thermal resistance of condenser
$T_{ambient}$	Ambient temperature
$C_{silicon}$	Thermal capacitance of silicon
C_{vapor}	Thermal capacitance of vapor core
ρ	Density
ϕ	Porosity
C_{pv}	Specific heat of vapor
T_v	Local vapor temperature
R	Gas constant
$T_{c,i}$	Temperature at the condenser/vapor interface
σ	Accommodation coefficient
h_{lv}	Latent heat of vaporization
P_v	Vapor core pressure
t_v	Thickness of vapor core
μ_v	vapor viscosity
\bar{T}_v	Average temperature in the vapor core
ϕ_{cond}	Porosity of the condenser wick
$k_{solid,wick}$	Thermal conductivity of the solid material used for the condenser wick
k_{liquid}	Thermal conductivity of liquid

4.5.1 Improved Compact Modeling of the Two-Phase VCs

The whole chip stack is shown in Fig. 4-10 (a). We divide the whole chip into grids as in prior work (Yuan et al., 2019a). For the processing layer, the grid cell structure is shown in Fig. 4-10 (b). Thermal resistance along the north, south, east, west, and vertical directions are represented using silicon properties, i.e., $R_{silicon}$. To build the hybrid wick evaporator CTM, we abstract both the nanoporous membrane and the microchannel layers into a hybrid wick layer. The hybrid wick grid cell is shown in Fig. 4-10 (c). We represent lateral thermal resistances using $R_{silicon}$ and the vertical thermal resistance, R_{hybrid} , stands for the inverse of the heat conduction from the hybrid wick to the saturated vapor. We consider a predetermined VC pressure and use a previously established relationship between HTC and thermal resistance to define R_{hybrid} (Sridhar et al., 2014; Yuan et al., 2019a). Since ther-

mal resistance is inversely proportional to the cross-section area and HTC (Sridhar et al., 2013a), the vertical thermal resistance is represented using Equation (4.3):

$$R_{hybrid} = \frac{1}{HTC \cdot w \cdot l}, \quad (4.3)$$

where w and l are the width and length of the grid cell, respectively. We extract HTC correlations for various nanoporous membranes and microchannel geometries from a COMSOL model in a previous work (Lu et al., 2016). For a specific hybrid wick geometry, we use its corresponding HTC correlation to determine the R_{hybrid} value.

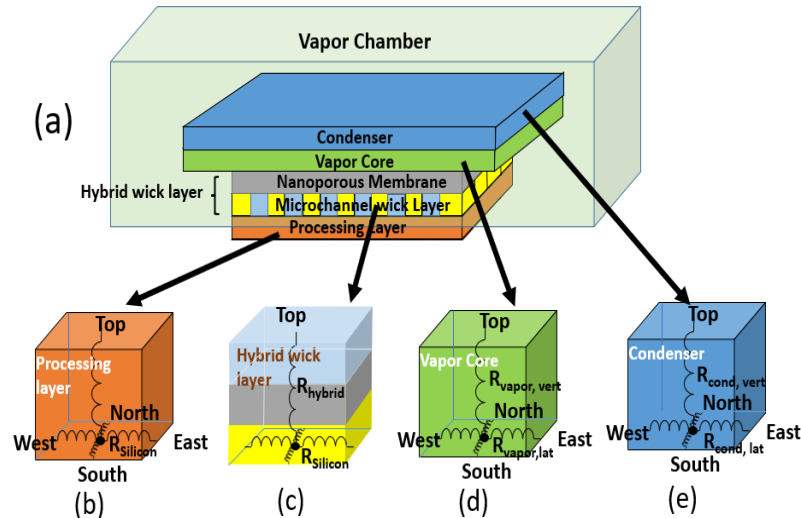


Figure 4-10: (a) The chip stack of the processing layer and two-phase VCs with hybrid wick evaporators, (b) processing layer grid cell, (c) hybrid wick layer grid cell, (d) vapor core grid cell, and (e) condenser grid cell.

Compact Modeling of the Vapor Core

On top of the hybrid wick layer, we place a vapor core layer to simulate the thermal behavior of the heated vapor inside VC as shown in Figure 4-10 (d). We adopt a compact modeling method of vapor core from a recent work (Baraya et al., 2020). The lateral ther-

mal resistance of the vapor core ($R_{vapor,lat}$) is defined as follows:

$$R_{vapor,lat,EW} = \frac{12R^2\mu_v\bar{T}_v^3}{h_{lv}^3P_v^2t_v^2l}, \quad (4.4)$$

$$R_{vapor,lat,NS} = \frac{12R^2\mu_v\bar{T}_v^3}{h_{lv}^3P_v^2t_v^2w}, \quad (4.5)$$

where $R_{vapor,lat,EW}$ is the $R_{vapor,lat}$ on the east-west direction and $R_{vapor,lat,NS}$ is the $R_{vapor,lat}$ on the north-south direction. The vertical thermal resistance of the vapor core ($R_{vapor,vert}$) is represented using Equation (4.6):

$$R_{vapor,vert} = \frac{RT_v^2\sqrt{2\pi RT_{c,i}}}{\chi h_{lv}^2 P_v}. \quad (4.6)$$

χ is calculated using the following expression:

$$\chi = \frac{\omega\sigma}{\sigma + (1 - \sigma)\omega}, \quad (4.7)$$

where ω is defined as $\frac{32\pi}{32+9\pi}$.

Compact Modeling of the Condenser

To simulate the condensing process of the heated vapor back to liquid. We add a condenser layer on top of the vapor core layer as shown in Figure 4-10 (e). The condenser layer consists of a porous wick to help cool the heated vapor (Baraya et al., 2020). $R_{cond,lat}$ and $R_{cond,vert}$ are defined using Equations (4.8), (4.9), and (4.10):

$$R_{cond,vert} = \frac{1}{k_{solid,wick}wl}, \quad (4.8)$$

$$R_{cond,lat,EW} = \frac{1}{(\phi_{cond}k_{liquid} + (1 - \phi_{cond})k_{solid,wick})t_vl}, \quad (4.9)$$

$$R_{cond,lat,EW} = \frac{1}{(\phi_{cond}k_{liquid} + (1 - \phi_{cond})k_{solid,wick})t_vw}. \quad (4.10)$$

The lateral thermal resistance of the condenser is a joint thermal resistance of the solid wick and liquid. $R_{cond,lat,EW}$ and $R_{cond,lat,NS}$ represent the $R_{cond,lat}$ on the east-west and north-south directions, respectively. $R_{cond,vert}$ is defined using the thermal resistivity of the solid material used for the condenser wick.

Transient Modeling of the VC

To model the transient behaviors of the two-phase VCs with hybrid wick evaporators, we define the thermal capacitance for each layer. For the processing layer, the thermal capacitance is $C_{Silicon}$. For the hybrid wick and condenser layer, the thermal capacitance is defined as the joint thermal capacitance of the solid and the liquid based on the porosity of the wick, ϕ . For the vapor core layer, the thermal capacitance is defined as C_{pv} . C_{pv} is a temperature-dependent parameter. We train a quadratic regression model to predict the value of C_{pv} and use that model in the improved ML-based temperature-dependent simulation framework discussed in the next section.

4.5.2 Improved ML-Based Temperature-Dependent HTC Simulation Framework

Compared to the two-phase VCs with hybrid wick evaporators CTM discussed in Section 4.3, we add additional modeling of the vapor core and condenser, which increases the simulation complexity. The vapor viscosity (μ_v) and thermal capacitance (C_{pv}) are temperature-dependent parameters and need to be simulated using CFD simulations. The original ML-based temperature-dependent simulation framework discussed in Section 4.4 only considers adding the ML regression model for HTC prediction. Whereas in this improved ML-based temperature-dependent simulation framework, we add additional linear and quadratic regression models to predict the μ_v and C_{pv} during each iteration as shown in Figure 4-11.

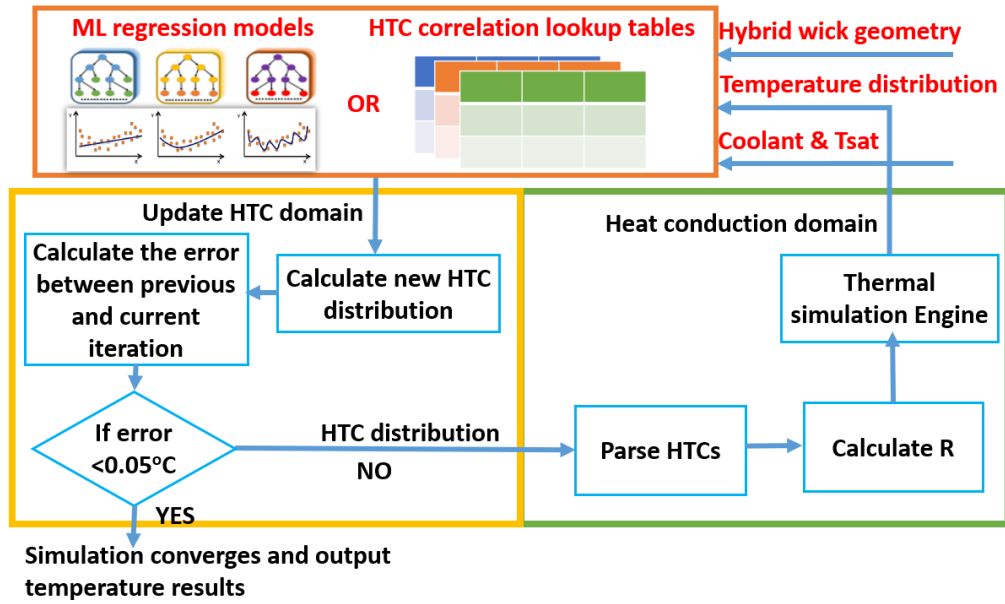


Figure 4-11: Improved ML-based temperature-dependent simulation framework.

4.5.3 Cooling Performance Evaluation of Two-Phase VCs on Realistic Mobile System

Two-phase VCs are widely used for high-performance computing systems with tight power and thermal budget, such as high-performance mobile systems (e.g., ROG Phone 3, Razer Phone 2, and Sony Xperia Pro). This section evaluates the cooling performance of the improved two-phase VCs with hybrid wick evaporators on realistic mobile systems using the improved CTM and ML-based temperature-dependent simulation framework. We use PicoSoC as a mobile processor and use OpenROAD (Ajayi et al., 2019) to generate the steady-state standard-cell-level power map. We assume an extreme power case for PicoSoC with the operating frequency of 3 GHz and total power of 9 W to mimic the gaming workload. We create two chip stacks. The first one directly places a copper heat spreader on top of the processing layer, and the second one place a VC on top of the processing layer. For both chip stacks, the processing layer has a thickness of 100 μm . For the copper heat spreader, the thickness is set to 1 mm while the thickness of the VC is 200 μm . Since the standard-cell design lacks dynamic power traces, we utilize the steady-state power values

of PicoSoC and randomly applied -15% or +15% additional power values for each standard cell and create synthetic transient power traces. We show the steady-state and transient results in Figures 4-12 and 4-13. For both steady-state and transient, the mobile chip with VC outperforms the traditional method (copper heat spreader) by at most 11%, demonstrating the cooling performance of using VC as the cooling method on the mobile system.

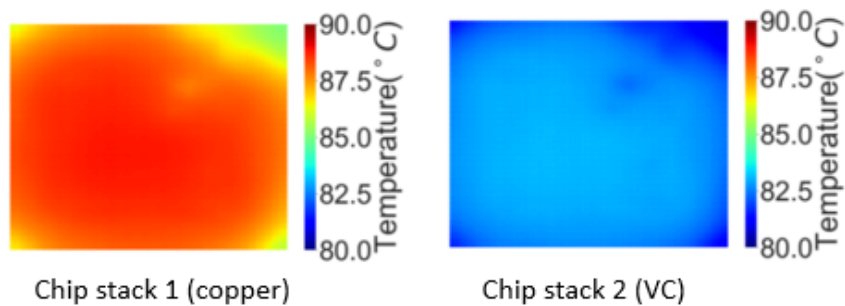


Figure 4-12: Steady-state heat maps.

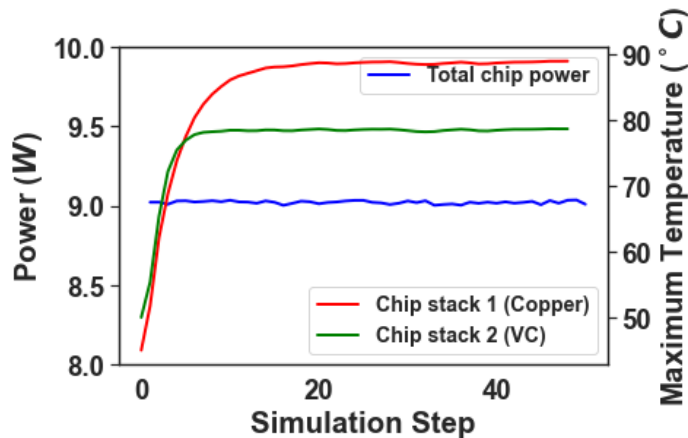


Figure 4-13: Transient comparison results.

4.6 Predicting Thermal Profiles via ML

High chip temperatures have been a primary concern for several decades. Localized hot spots resulting from these high power densities not only decrease the lifetime of processors (Srinivasan et al., 2003) but also increase transistor delays as well as leakage

power (Saini and Mehra, 2012). In addition, the heterogeneity in on-chip heat distribution incurred by these hot spots is expected to become more severe with the integration of heterogeneous architectures on a single die, such as a collection of CPUs, GPUs, accelerators, and FPGAs. To enhance reliability, researchers have proposed runtime policies that use control knobs such as dynamic voltage and frequency scaling, task scheduling, and thread migration (e.g., (Sheikh et al., 2012)). Modern processors utilize digital thermal sensors to track the processor’s temperature at various strategic locations to manage runtime temperatures. However, on-chip thermal sensors may not accurately measure the temperature profile and maximum temperature on-chip. We identify three major challenges in accurately obtaining the temperature profile and hot spot temperatures using thermal sensors. First, because of the placing and routing difficulties, thermal sensors may not be placed at the exact location of the hot spots, leading to under-estimating the hot spot temperatures and may change the dynamic thermal runtime policy decision (Reda et al., 2011). Second, the spatial and temporal fluctuations in thermal hot spots due to workload behavior make tracking the hot spot temperature on-chip particularly challenging (Sadiqbatcha et al., 2022; Reda et al., 2011). Third, on-chip thermal sensors operate within an error margin, which could under/over-estimate the temperature readings by $\pm 1^{\circ}\text{C}$ (Sharifi and Rosing, 2010; Long et al., 2008).

To reconstruct accurate on-chip thermal profiles, a recent body of work has introduced using ML models to predict chip temperatures trained with infrared (IR) camera measurements of the physical chip (Zhang et al., 2022; Sadiqbatcha et al., 2022). Other works investigate how to intelligently place the thermal sensors on-chip to perform accurate thermal profile monitoring via IR camera measurements (Reda et al., 2011; Nowroz et al., 2010). While existing methods produce accurate temperature results, the expensive IR camera setup makes these methods hard to implement broadly by the research community. In addition, additional steps of collecting and processing data from the IR camera

measurements make this method complex and time-consuming. Previous work has also introduced a simulation-based method to mitigate the inaccuracies of the on-chip thermal sensors based on analytical models (Sharifi and Rosing, 2010). However, this method targets estimating the temperature at locations of interest instead of regenerating the full thermal profile. Predicting the full thermal map is essential since being able to identify both the hot and cold spots on-chip benefits the runtime policies such as task allocation and scheduling to achieve better chip performance under temperature constraint (Chrobak et al., 2008). In addition, the locations of interest are challenging to determine, given the different behaviors of the workload.

This section proposes a simulation-based method of using a ML regression model to predict a chip’s full temperature profile based solely on the current total power usage of the chip, workload-core mappings, and measured thermal sensors temperatures. We train and validate the proposed ML model based on data generated from architectural performance, power, and thermal simulations of an Intel i7 6950× processor. We observe that using the proposed method with simulation data trains a highly accurate ML regression model. In addition, the proposed simulation-based method is generally applied to many processor designs without necessitating an expensive thermal camera setup.

4.6.1 Methodology

In this section, we first overview the methodology for generating a realistic training dataset for the ML model through architectural performance, power, and thermal simulators. Then, we discuss the proposed linear regression ML model as well as the training and validation methodologies. We use Intel i7 6950× processor (Sima, 2018) as our target processor and running ten different applications from the NAS parallel benchmarks (Bailey et al., 1991). The floorplan of the Intel i7 6950× is shown in Figure 3.23.

Training Data Preparation

In order to generate realistic temperature data for the Intel i7 6950 \times , we first use the architectural power and performance simulators, Sniper (Carlson et al., 2011) and McPAT (Li et al., 2009), to simulate power usage for a set of realistic benchmark applications. To ensure our ML model is accurate for any workload or configuration of the CPU, we must ensure that the model has seen a wide range of workloads and applications. To generate this realistic set of training data, we select ten applications from the NAS parallel benchmarks: *bt*, *cg*, *dc*, *ep*, *ft*, *is*, *lu*, *mg*, *sp*, and *ua*. We map the application to a different number of cores (1-10) with different workload-core mapping policies for each application. For example, we map application *cg* to 5 cores with a workload-core mapping policy of 1, 3, 4, 8, and 9. There are 1023 possible workload-core mappings for a ten-core CPU for each application. Note that we only consider running one application for Intel i7 6950 \times at a time. If the application hasn't been mapped to a core, we set the core to an idle state. We select a random subset containing 36 workload-core mappings to generate the training data.

We run all ten applications from the NAS parallel benchmarks in Sniper for each workload-core mapping and then run McPAT to simulate the power usage. Finally, we extract the power traces from the McPAT. We use PACT (Yuan et al., 2022; Yuan et al., 2021) as the thermal simulator. The power traces generated using Sniper and MCPAT are used directly as inputs to PACT. The original power traces generated from Sniper and McPAT are transient. To simplify the inputs of the ML model, we average the power traces over the time steps for each application and workload-core mapping to generate steady-state power profiles. The total number of the generated power profiles is 360. The power profiles are then scaled based on the TDP of Intel i7 6950 \times , which is 140 W, to ensure the model stays realistic. To scale the power profile, we first take the maximum total power of all the power profiles. We then calculate the scaling factor by dividing the TDP by the maximum total power of all the power profiles and then multiply all the function blocks

power values by this scaling factor. The resulting scaled power profiles are used as the inputs to PACT and the ML model.

In order to simulate the thermal behavior of the chip in PACT, we need to model the floorplan and power profile of the chip. The floorplan of the chip describes the dimensions, locations, and thermal material properties of the CPU’s physical functional blocks (e.g., CPU cores or cache blocks). The input power profile for each simulation run describes the power utilization of each functional block. PACT first divides the power profile into a power grid matrix using a predefined grid resolution. It then uses this information to simulate the amount of heat generated by each grid and block and the heat flow between the CPU and cooling layers. For this dataset, we create the power traces with Sniper and McPAT, average and scale the power traces into steady-state power profiles, and then run PACT simulation in steady-state grid mode with a grid resolution of 64×64 . The output of the PACT simulation for each run is a 64×64 temperature grid matrix. In this way, each power profile corresponds to a single temperature grid matrix. For the cooling method, we use PACT’s medium-cost heat sink (Yuan et al., 2022), which has a size of $40 \times 40 \text{ mm}^2$, and a heat spreader with a size of $20 \times 20 \text{ mm}^2$. The chip layer has a thickness of 0.1 mm and a physical dimension of $14.6 \times 16.8 \text{ mm}^2$. For our set of 360 power profiles, we run the same chip stack using each power profile in PACT to generate 360 corresponding temperature grid matrices to be used in our ML model.

ML Model

The goal of our ML model is to predict the full temperature grid matrix (temperature profile) of a CPU based on the power and temperature metrics available at system runtime. The model’s input values are the total power usage of the chip at the time of prediction, the temperature values reported by on-chip thermal sensors, and the CPU workload-core mapping, a binary value based on one-hot encoding for each core that represents whether that core is in use. For example, if cores 1, 2, 3 and 7 are in use, the corresponding values

would be 0, 1, 1, 1, 0, 0, 0, 1, 0, 0. The model's output data is an array of temperature values corresponding to the 64×64 temperature grids. For the temperature values reported by on-chip thermal sensors, we randomly select ten temperatures from the temperature grid matrix obtained by performing thermal simulation on each power profile and report these temperatures as the readings from thermal sensors. The flow diagram of the temperature profile prediction is shown in Figure 4-14.

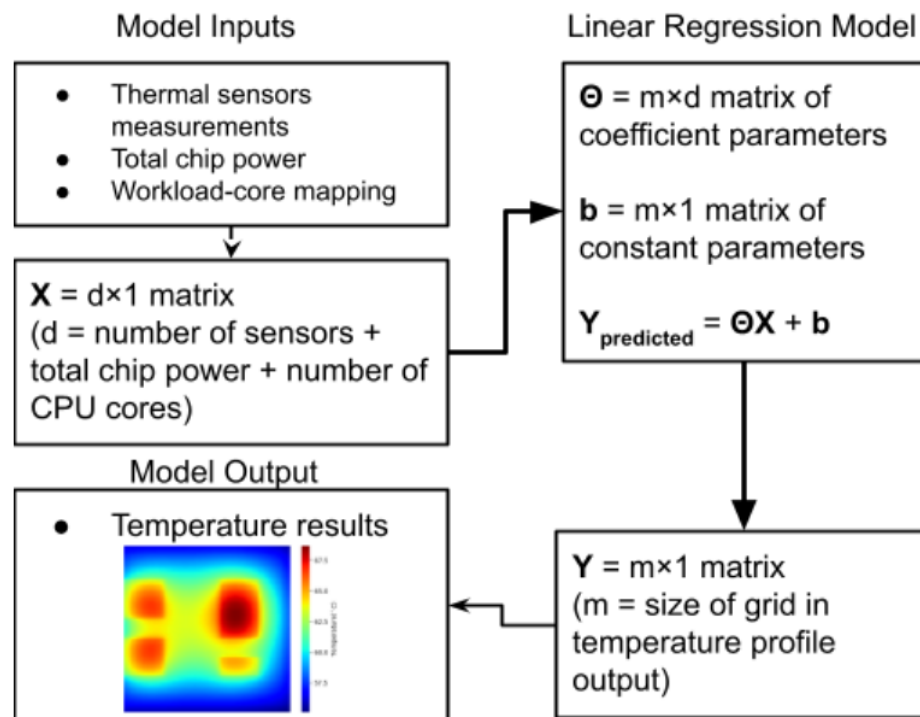


Figure 4-14: Diagram of the ML model.

In Figure 4-14, the inputs and outputs of the ML model are represented as matrices X and Y , where X is a combination of on-chip thermal sensors measurements, total chip power, and binary CPU workload-core mapping, and Y represents the output temperature grid matrix (e.g., 64×64 temperature grid matrix). Since we predict temperature values based on thermal sensors readings and the chip's total power, we select a linear regression model, where each independent predictor models a temperature grid node. Each predictor

comprises a set of coefficients corresponding to each input value and a constant. In the linear regression model, the output $Y_{predicted}$ is directly calculated using $\Theta X + b$, where Θ is the coefficient parameters matrix, and b is the matrix of constant parameters. The input on-chip thermal sensors measurements are extracted from the PACT output temperature grid matrix by using the temperatures at the grid locations of the thermal sensors. We evaluate the accuracy of the linear regression model for a range of ten randomly selected thermal sensor locations used as input in the next section.

We use the Scikit Learn library for Python and the `LinearRegression` class for our model training and evaluation. We train the model by splitting the simulation data into training and testing data and then evaluating the accuracy of the trained model with the testing data. To validate the model's accuracy, we first perform a 5-fold CV, where the data is split into five randomized buckets. For each bucket, we train the model using the rest of the data, test using that set of data, and record the accuracy for each. Lastly, we perform leave one out cross-validation (LOOCV) on our model to evaluate its accuracy by excluding applications and core allocations from the training data. We divide the dataset into buckets based on the applications or the number of enabled cores to run the application, then perform n-fold CV. This LOOCV aims to test if the model is still accurate for the applications and workload-core mappings that were not included in the training set, ensuring that the ML model can be trained without extensive applications and datasets, and be applied generally. The results of these CVs are in the next section.

4.6.2 Results and Discussions

In this section, we validate the power scaling of the simulation data used for training our model and the model itself. We evaluate the model's performance based on different CV methods and discuss how qualitative properties of the dataset, such as average power, affect the model's accuracy.

Training Data Evaluation

As detailed in the previous section, the training data for the ML model is generated through the combined use of the architectural performance and power simulators, Sniper and McPAT, and the thermal simulator PACT. With Sniper and McPAT, we are able to simulate the relative power utilization of the different CPU functional blocks. However, since McPAT lacks awareness of some of the implementation details of the CPU architecture, the outputs from McPAT may not reflect realistic power values (Lee et al., 2015). Therefore, the McPAT outputs have to be calibrated to reflect the TDP of the Intel i7 6950× chip. We show the unscaled power values directly collected from McPAT in Figure 4-15. The chip’s total power with ten cores goes to nearly 350 W, which is unrealistic for a ten-core desktop processor. To calibrate the power, we scale all of the steady-state power profiles using the same scaling factor discussed in the previous section, such that the resulting maximum total power matches the chip’s TDP, in this case, 140 W as shown in Figure 4-16.

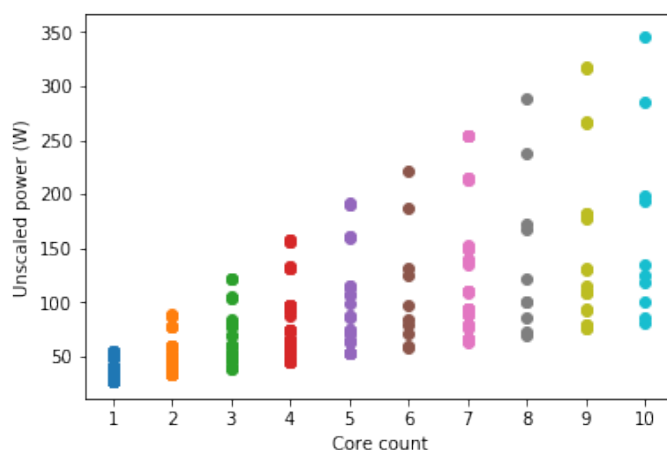


Figure 4-15: Original total chip power from McPAT, split by the number of enabled cores.

To analyze the total power of each application, we average the total powers over the number of enabled cores and split them by application as shown in Figure 4-17. We observe that most of the NAS parallel benchmarks applications result in an average power

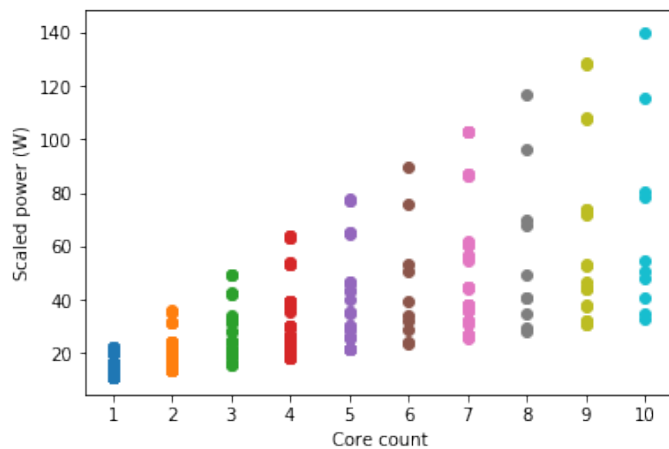


Figure 4-16: Scaled total chip power, split by the number of enabled cores.

range of 20-40 W . Applications *bt* and *ft* are high power applications and result in high average power of more than 50 W , while applications *cg* and *is* are relatively low power applications.

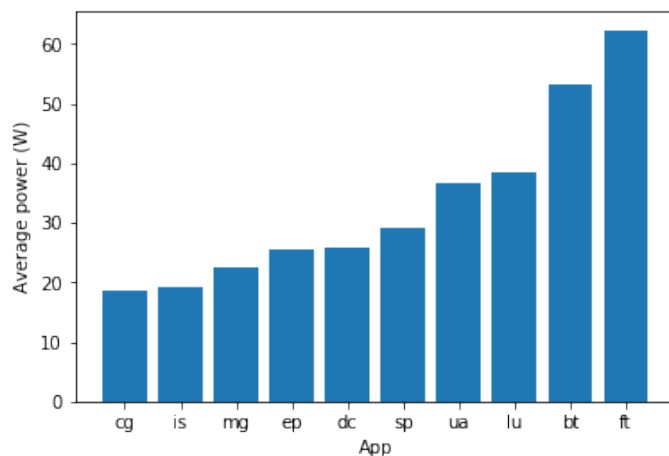


Figure 4-17: Average power of each application in the NAS parallel benchmarks.

Next, we show the temperature results obtained by running PACT with all the steady-state training power profiles. In Figure 4-18, we show the average temperatures for applications and split them by the number of enabled cores. The average temperature increases as we increase the number of cores to run the application. This is because of the total power

increase as we increase the number of enabled cores, as shown in Figure 4-16. We also show the average temperature for the number of enabled cores to run the applications in Figure 4-19. Applications *bt* and *ft* result in the highest average temperatures, and applications *cg* and *is* have the lowest average temperatures. The average temperatures generally follow the trend of their average power behaviors, as shown in Figure 4-17. However, since other factors such as the hot spots' locations and power densities also affect the average temperature, higher average powers may result in lower average temperatures (e.g., applications *dc* and *lu*).

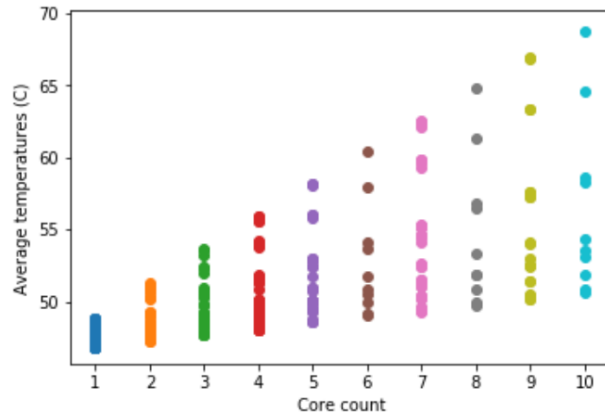


Figure 4-18: Average temperatures for applications ($^{\circ}\text{C}$), split by the number of enabled cores used.

Validation of the ML Model

Since our ML model uses a random sample of ten temperatures from the temperature grid matrix as the temperature values reported by on-chip thermal sensors (input to the model), we need to show that the model's accuracy is not affected by the randomness of the thermal sensor placements. To test the effect of the randomly selected thermal sensor locations on the model's accuracy, we evaluate the model's R2 score and RMSE across a wide range of random thermal sensor placements. We perform a 5-fold CV using the same random state for the train test split for each random sample of the ten thermal sensors readings. We show

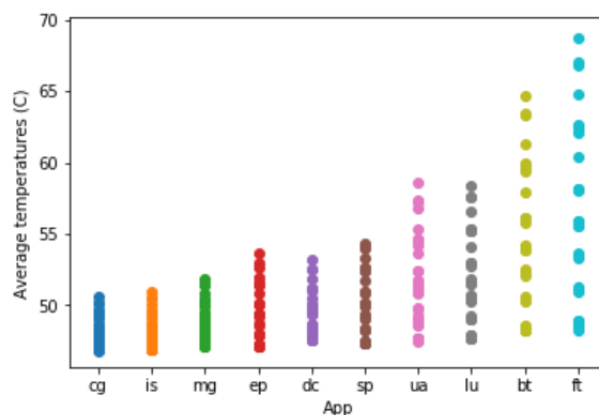


Figure 4-19: Average temperatures ($^{\circ}\text{C}$) for the number of enabled cores to run the applications, split by application and sorted in ascending order by average power.

the histograms of R2 score and RMSE in Figures 4-20 and 4-21. As a result, we observe that the locations of the thermal sensors on the chip affect the model's accuracy by at most 0.12°C . This accuracy loss indicates that our proposed ML model's accuracy is invariant to the placements of the thermal sensors. We use the thermal sensor placement corresponding to the median error for the remaining model validation results as shown in Figure 4-22.

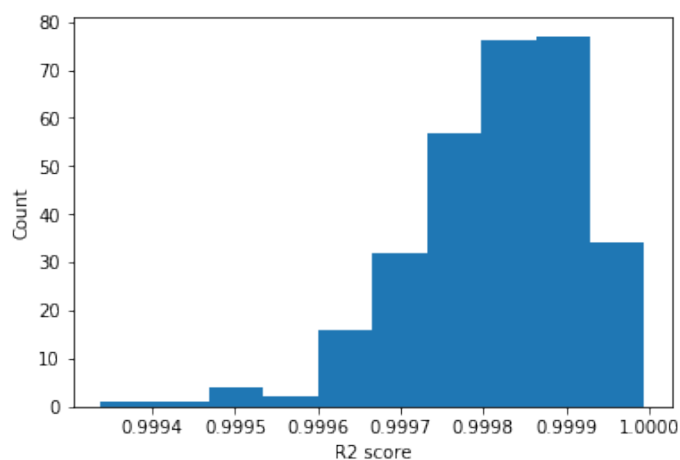


Figure 4-20: Histogram of R2 score distribution across 300 different thermal sensor placements used as input to the model.

The results of the 5-fold CV with the selected thermal sensors locations are shown in Table 4.9. The 5-fold CV results show that the proposed linear regression model has a high

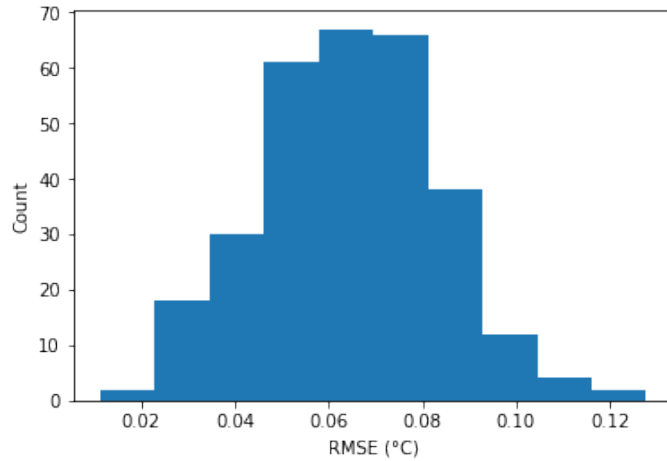


Figure 4-21: Histogram of RMSE distribution across 300 different thermal sensor placements used as input to the model.

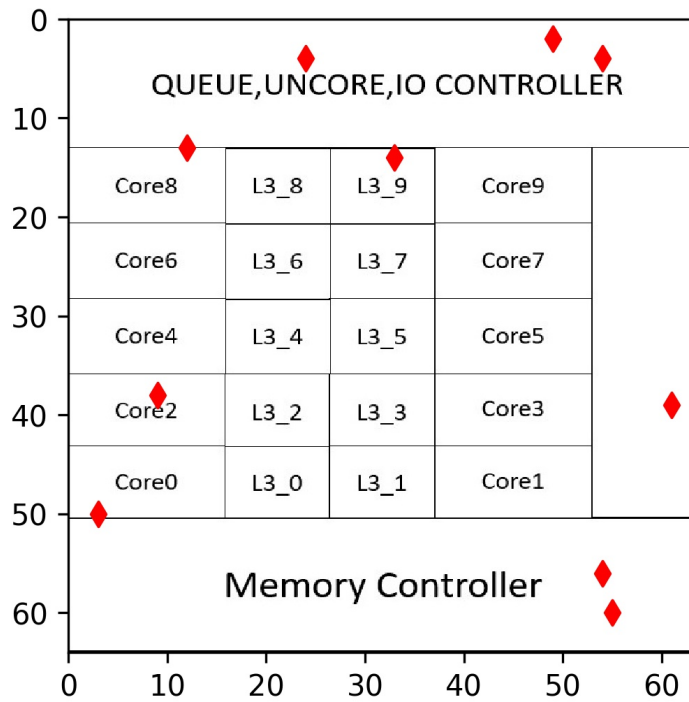


Figure 4-22: Selected thermal sensor placement for our experiments. Red markers indicate thermal sensors.

coefficient of determination with the lowest R2 score of 0.9996. In addition, the model itself is accurate with the highest RMSE of 0.0695°C. We demonstrate the comparison of the golden heat map and the predicted heat map using the ML model for the worst-case

accuracy of the 5-fold CV experiments in Figure 4-23. We observe that the hot spot on the predicted heat map has a lower temperature than the hot spot on the golden heat map. The application that results in the highest accuracy loss, in this case, is *ft*. There are two reasons behind this accuracy loss. First, application *ft* is the highest power application, with an average power of more than 60 W. The majority of the applications we used to train the ML regression model have average power within the range of 20-40 W, which means our model is more likely to learn the power and thermal trends of these medium power applications. Second, as shown in Figure 4-16, when the number of enabled cores to run the application is one, the total power of the chip is less than 30 W. Therefore, predicting the heat map for the highest power application and lowest power workload-core mappings results in the highest validation accuracy.

Table 4.9: 5-fold CV results.

5-fold CV	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
R2 Score	0.9997	0.9996	0.9997	0.9997	0.9997
RMSE	0.0678	0.0695	0.0657	0.0669	0.0688

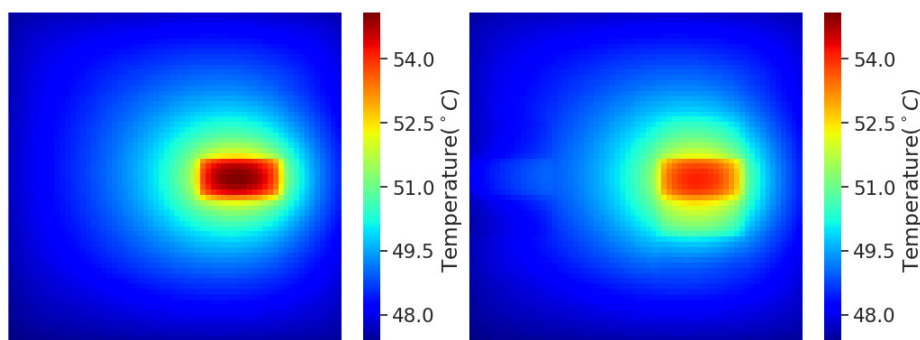


Figure 4-23: Heat map comparison for the worst case. The left heat map is the golden heat map, and the right heat map is the predicted heat map.

Lastly, to validate that the model training methodology can be applied to various use cases and configurations, we perform LOOCV on both the benchmark applications and the number of enabled cores to run the applications. For the applications, we split the data

into ten buckets based on the type of the benchmark. Then we train the model using nine buckets, leaving one out for validation. This CV is repeated for each bucket, measuring the model’s accuracy with respect to each application. The LOOCV for the number of enabled cores is similar. We split the data based on the number of enabled cores (core counts) to run the application. Then, we train the model for each core using the data containing the rest of the core counts, then test the accuracy on the left out one. We show the LOOCV results on applications in Figures 4-24 and 4-25. The comparison of the golden heat map and the predicted heat map for the number of enabled cores equal to 5 is shown in Figure 4-26. The application that causes the highest accuracy loss for five cores is *ft*. The reason is that most of the applications we used to train the ML model are medium power applications (average power within 20-40 W). However, suppose we train the model without high power applications (e.g., *bt* or *ft*). In that case, our model predicts high power applications heat maps less accurately and result in an RMSE error of less than 0.25°C .

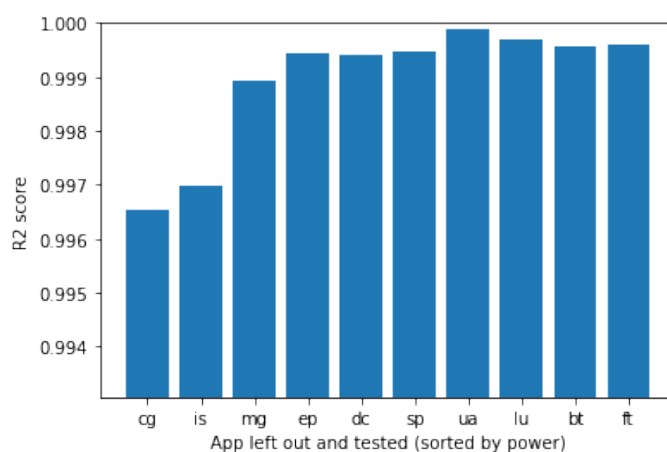


Figure 4-24: Average LOOCV R2 scores on applications.

In addition, for lower power applications *cg* and *is*, the RMSEs are also relatively high. Meanwhile, the applications *mg*, *ep*, *dc*, *sp*, *ua*, and *lu* have lower errors. This trend indicates that this model is generally able to predict CPU temperatures for application workloads not seen in the training data. However, the training data should include the upper and

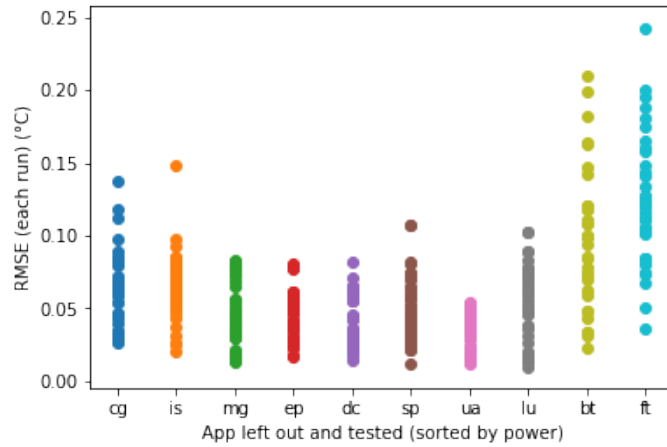


Figure 4-25: LOOCV RMSEs on applications.

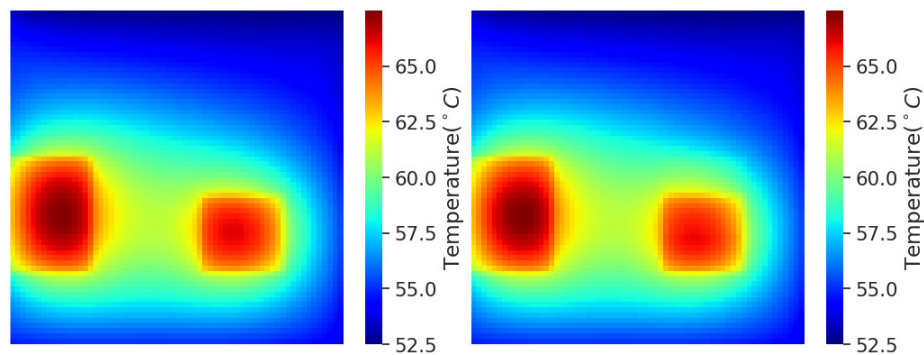


Figure 4-26: Heat map comparisons for the worst case when the number of cores used is equal to 5. The left heat map is the golden heat map, and the right heat map is the predicted heat map.

lower extreme applications in terms of power and temperature to get the best accuracy.

We perform similar LOOCV on the number of enabled cores to run the applications. This time, we split the data into buckets based on the number of enabled cores and train the model using all but one of these buckets. We demonstrate the LOOCV on the number of enabled cores in Figures 4-27 and 4-28. We also illustrate the comparison of the golden heat map and the predicted heat map for the number of enabled cores equal to ten in Figure 4-29. The application that results in the highest RMSE for ten cores case is *lu*.

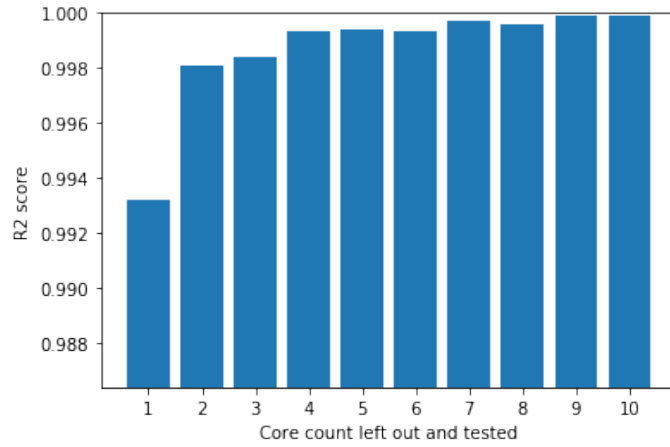


Figure 4-27: Average LOOCV R2 scores on enabled cores.

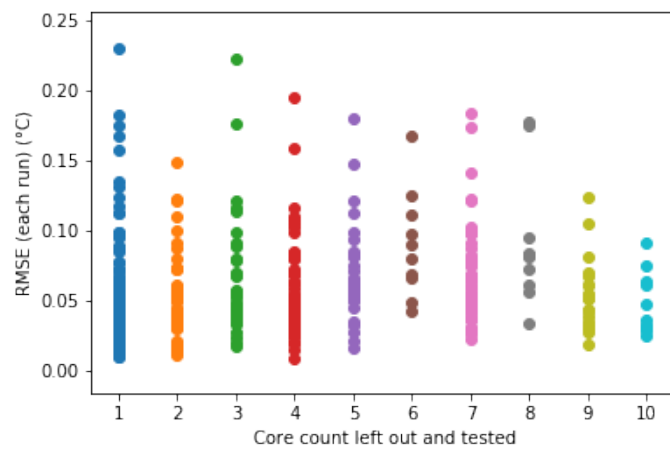


Figure 4-28: LOOCV RMSEs on enabled cores.

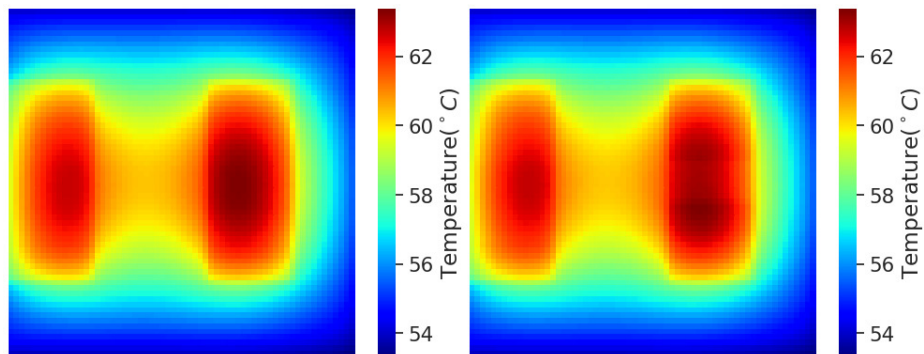


Figure 4-29: Heat map comparison for the worst case when the number of enabled cores is equal to 10. The left heat map is the golden heat map, and the right heat map is the predicted heat map.

Chapter 5

Optimizing Emerging Cooling Methods for High-Performance Processors via Deep Learning

5.1 Introduction

As discussed in Chapters 2 and 4, conventional on-chip cooling solutions such as forced air cooling via fans or pin-fin heat sink are often not sufficient to mitigate such high-power-density hot spots and result in over/under-cooling. Emerging cooling technologies such as liquid cooling via microchannels (Dang et al., 2010), TECs (Chowdhury et al., 2009), two-phase VCs (Bulut et al., 2019), and hybrid cooling options (Yazawa et al., 2012) (e.g., of liquid cooling via microchannels and TECs) have the potential to provide better cooling performance compared to the conventional cooling solutions. However, there is no obvious winner in terms of cooling efficiency among all these emerging cooling technologies. These potential solutions' cooling performance and cooling power vary significantly based on the cooling parameters (such as liquid flow velocity, evaporator design, TEC current, etc.) (Yuan et al., 2019a; Yuan et al., 2020). The selection of the cooling technologies and the cooling parameters also needs to consider the chip architecture, chip size, floorplan, and the power profiles of the applications running on the given chip. To minimize the cooling power while satisfying chip thermal constraints, there is a need for an optimization flow that enables rapid and accurate selection of the optimal cooling solution and the associated cooling parameters for a given chip and application profile.

A key enabler to such a cooling design optimization flow is a set of accurate and fast models for various cooling technologies. A common approach towards this direction is

using CTMs that model heat dissipation with an equivalent lumped circuit model (Pedram and Nazarian, 2006). However, given the vast solution space of possible cooling solutions (including possible hybrids) and cooling parameters, the optimal solution search time is still prohibitively time-consuming with CTMs (Yuan et al., 2019a). In addition to cooling design choice possibilities, the optimization flow needs to also account for the chip design and power profile changes. In this case, using a simple grid search to find the optimal cooling design for a small-sized chip floorplan and its typical power profile could take up to days (Yuan et al., 2020).

This chapter first discusses the cooling parameters optimization flows for two-phase VCs with micropillar wick evaporators via grid search and the cooling parameters optimization flows for two-phase VCs with hybrid wick evaporators via multi-start simulated annealing (MSA). Then it elaborates on using the covariance matrix adaptation evolution strategy (CMA-ES) to select the most power-efficient cooling methods. Finally, it presents cooling methods and cooling parameters optimization flow using DL models.

5.2 Two-Phase VCs Optimization Flows

5.2.1 Two-Phase VCs with Micropillar Wick Evaporators Optimization via Grid Search

This optimization flow aims to find the optimal micropillar wick evaporator geometry that results in the highest HTC while satisfying the dry-out constraint. Since the micropillar wick geometry solution space is small, we use grid search to find the optimal geometry as shown in Algorithm 1. We first calculate the dry-out heat flux for each micropillar geometry and input combination. Then, if the geometry fails to provide a dry-out heat flux greater than or equal to the maximum chip power density, we proceed to the next geometry. Finally, among all the geometries that satisfy the dry-out heat flux, we pick the one that results in the highest HTC.

Algorithm 1: Micropillar Geometry Optimization Flow

Input : $q_{dry-out}$ Dry-out heat flux
Input : Q_{max} Hot spot power density on-chip
Input : HTC Heat transfer coefficient
Input : ψ Dimensionless function of micropillar geometry
Input : M Figure of merit for the coolant
Input : θ_{rec} Contact angle
Input : MG Micropillar geometry
Output: MG_{best} Optimized micropillar geometry

```

1  $MG_{best} = None$ 
2 for each  $MG$  in lookup table do
3    $q_{dry-out} = (40/3)\psi M \cos\theta_{rec}$ 
4   if  $q_{dry-out} > Q_{max}$  then
5     if  $HTC_{MG} > HTC_{MG_{best}}$  then
6        $MG_{best} = MG$ 
7     else
8        $next\ MG$ 
9   else
10     $next\ MG$ 
  
```

5.2.2 Two-Phase VCs with Hybrid Wick Evaporators Optimization via MSA

The ultimate goal of this optimization flow is to find a hybrid wick geometry that minimizes the hot spot temperatures while satisfying the dry-out constraint. We adopt the dry-out heat flux formula from recent work and use this formula to define the dry-out limit (Lu et al., 2016). For each hybrid wick geometry parameter, there is a range of values we can select. Using the grid search to find the optimal geometry and coolant from a fine-grained geometry and coolant solution space is time-consuming and inefficient. We propose an MSA approach to speed up the searching time for the optimal hybrid wick geometry and coolant.

Our proposed MSA algorithm is shown in Algorithm 2. The algorithm randomly selects a hybrid wick geometry G and checks whether it satisfies the dry-out constraint by comparing it to the maximum power density PD_{Max} (lines 2-6). If the dry-out heat constraint

is satisfied, MSA runs the ML-based temperature-dependent HTC simulation framework to get the maximum chip temperature T_{Max} (line 7). The algorithm then randomly adds a perturbation to one of the geometry parameters and makes a new geometry G_{Nbr} (line 9). Next, MSA checks if G_{Nbr} is in the valid parameter range and also satisfies the dry-out constraint (lines 11-13). If the constraints are met, MSA runs the ML-based simulation framework for G_{Nbr} to get T_{Max_Nbr} (line 14). If the G_{Nbr} results in a lower peak temperature than G , the algorithm sets G to G_{Nbr} and T_{Max} to T_{Max_Nbr} (lines 15-17). Otherwise, the algorithm sets G to G_{Nbr} based on the probability function (lines 18-20). The algorithm terminates based on the $numstart$, energy E , and decay factor δ (lines 1, 8-9, and 21). It also saves the best G and T_{Max} into an array Opt (lines 22-23). For each type of coolant, we execute this MSA algorithm and select the best geometry and coolant that result in the minimum T_{Max} .

To evaluate the efficiency of our proposed optimization flow, we compare the optimal geometries, coolants, the corresponding peak temperatures, and the searching time of MSA and grid search. We select the solution space for grid search to be 4096 hybrid wick geometries. For each geometry, we first compare the dry-out heat flux of the geometry to the maximum power density PD_{Max} . We then collect all the geometries that satisfy the dry-out constraint and select the one that has the minimum peak temperature. We perform this grid search for each coolant and pick the optimal coolant and the best hybrid wick geometry. This grid search is coarse-grained because the solution space does not contain all valid hybrid wick geometries.

As for MSA, we set the initial energy E to 1, the decay factor δ to 0.9, and minimum energy E_{Min} to 0.01. num_start and $iter$ are set to 10 and 100, respectively. We use three different floorplans (see Figure 5.1) with a background power density PD_{Bg} of $50 W/cm^2$ and hot spot power density PD_{Hs} of $\{100, 500, 1500, 2000\} W/cm^2$. In all of the experiments, our proposed optimization flow selects combinations of coolant and geometry that

result in lower temperatures (average 0.67°C and maximum 1.78°C) compared to the selections made by grid search. Most importantly, the maximum and average searching and simulation times for 4096 solutions grid searches are 19 and 6.67 hours, respectively. However, the maximum and average simulation times of our proposed MSA are 2.05 and 1.57 hours, respectively. Our proposed MSA's maximum and average speedup are $9.4\times$ and $4\times$, respectively, which means our proposed optimization flow is more efficient than the grid search. We also observe that the dry-out limit is highly correlated with the chip size and the number of hot spots. If the chip size is larger and there are a larger number of hot spots, only water meets the dry-out constraint. R245fa and R141b generally have better HTC than water, but they suffer from low critical dry-out heat flux. R245fa and R141b can be used as coolants for small-size chips and fewer hot spots.

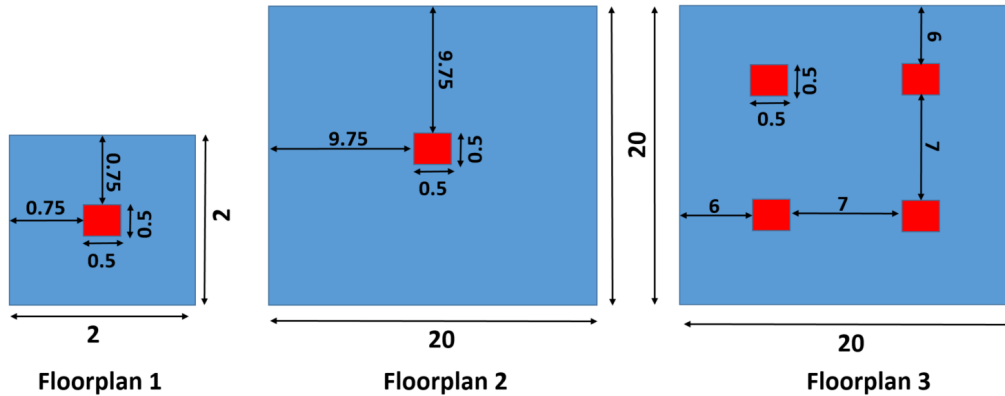


Figure 5-1: Experimental floorplans. Dimensions are in *mm*.

5.3 Emerging Cooling Methods Optimization via CMA-ES

This optimization flow aims to select the best cooling method and its cooling parameters for a target chip and its power profile while minimizing the cooling power under a temperature constraint. We incorporate liquid cooling via microchannels, hybrid cooling (of liquid cooling via microchannels and TEC), and two-phase VCs with micropillar and hybrid wick evaporators as cooling solution candidates. These cooling methods have been shown to

Algorithm 2: Multi-Start Simulated Annealing

Initialize: E , $iter$, δ , num_start , E_{Min}

```

1 while  $num\_start > 0$  do
2   randomly select hybrid wick geometry  $G$ 
3   calculate dry-out heat flux  $Q_{Dry\_G}$  (Lu et al., 2016)
4   if  $Q_{Dry} < PD_{Max}$  then
5     continue
6   else
7     run thermal simulation using  $G$  and get  $T_{Max}$ 
8     while  $E > E_{Min}$  and  $iter > 0$  do
9       randomly select a neighbor geometry  $G_{Nbr}$ 
10       $iter -= 1$ 
11      if  $G_{Nbr}$  in valid parameter range then
12        calculate dry-out heat flux  $Q_{Dry\_Nbr}$  (Lu et al., 2016)
13        if  $Q_{Dry} < PD_{Max}$  then
14          run thermal simulation using  $G_{Nbr}$  and get  $T_{Max\_Nbr}$ 
15          if  $T_{Max\_Nbr} < T_{Max}$  then
16             $G = G_{Nbr}$ 
17             $T_{Max} = T_{Max\_Nbr}$ 
18          else if  $Random(0,1) < \frac{T_{Max\_Nbr} - T_{Max}}{T_{Max} * E}$  then
19             $G = G_{Nbr}$ 
20             $T_{Max} = T_{Max\_Nbr}$ 
21           $E = E * \delta$ 
22      Save  $G$  and  $T_{Max}$  into an array  $Opt$ 
23       $num\_start -= 1$ 
24 Pick the best  $G$  from  $Opt$  based on  $T_{Max}$ 

```

achieve higher cooling performance than the traditional heat sink and forced air cooling via fan (Yazawa et al., 2012; Sridhar et al., 2014; Bulut et al., 2019). In addition, they are also compatible with processor cooling. We adopt the CTMs for liquid cooling via microchannels and hybrid cooling from recent works (Kaplan et al., 2017; Sridhar et al., 2013b).

$$\min \quad \alpha P_{cooling,norm} + \beta (\max(T_{hs} - T_{limit}, 0), norm) \quad (5.1)$$

The objective function of our optimization is to minimize the cooling power under a temperature constraint (Equation (5.1)). The cooling power for liquid cooling via mi-

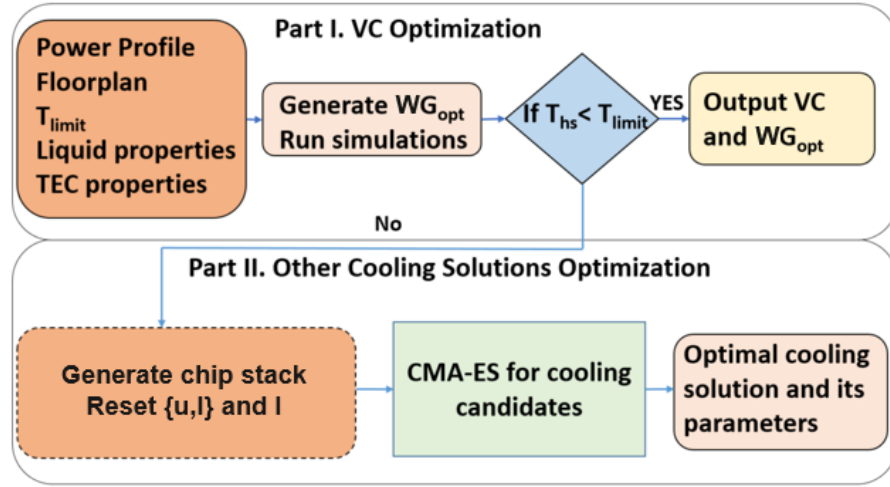


Figure 5.2: Proposed optimization flow.

crochannels (i.e., the pumping power) is calculated based on the pressure drop along the channel and the liquid volumetric flow rate (Coskun et al., 2009), while TEC cooling power is the difference between heat absorbed and rejected on the cold and hot sides. The maximum attainable liquid flow velocity and TEC current are set to 2.6 m/s and 7 A , respectively, owing to system constraints (Sridhar et al., 2014; Chowdhury et al., 2009). In Equation (5.1), the cooling cost is normalized with respect to the maximum cooling power (i.e., $u = 2.6 \text{ m/s}$ and $I = 7 \text{ A}$). The difference between hot spot temperature (T_{hs}) and temperature limit (T_{limit}) is normalized with the user-defined maximum on-chip temperature, i.e., T_{limit} . α is the user-specific weight factor with no unit, and β is the penalty weight that we set to a large value to prevent violation of the temperature constraint. We set $\alpha = 0.05$ and $\beta = 0.95$ according to our system. For two-phase VCs with micropillar wick evaporators and hybrid wick evaporators, we need to add the dry-out constraints that prevent dry-out by ensuring that the dry-out heat flux of the selected wick geometry is greater than or equal to the hot spot power density. In addition to that, we also incorporate micropillar geometry constraints ($h/i \geq 0.2$, $0.06 < d/i < 0.6$) and hybrid wick evaporator constraints as shown in Table 4.5.

Our proposed optimization flow is shown in Figure 5.2. We divide the optimization flow into two parts. The first part determines the micropillar geometry and hybrid wick geometry with the highest HTC under the dry-out constraint. We then simulate the chip using two-phase VCs with micropillar wick and hybrid wick evaporators. We select two-phase VCs as the optimal technique if a wick geometry satisfies the temperature constraint. The reason is that VC is a passive cooling device that requires no additional power on the evaporator side. Otherwise, we use CMA-ES to find the optimal $\{u, I\}$ pair for hybrid cooling and optimal I for liquid cooling via microchannels (see Table 4.1). CMA-ES is a stochastic, derivative-free sampling method that does not require a numerical objective function to converge to an optimal solution. The pseudo-code of our CMA-ES implementation for hybrid cooling is shown in Algorithm 3.

The optimization flow then enters the second part when the first part fails to find a wick geometry that satisfies the temperature constraint. In each iteration, the algorithm first samples the $\{u, I\}$ pairs based on a multivariate normal distribution and then runs thermal simulations for hybrid cooling with those sample points (lines 2-4). Next, all the $\{u, I\}$ pairs are sorted in increasing cost (line 5). In lines 6-7, the algorithm assigns a weight vector w to the $\{u, I\}$ pairs to update the mean of the sampling distribution (Auger and Hansen, 2012). This step ensures that the sampling distribution for the next iteration moves closer to the $\{u, I\}$ pair that gives the minimum cost in the current iteration (Auger and Hansen, 2012). Next, the algorithm updates the evolution paths, p_c and p_σ , which conceptually stand for the search paths in the CMA-ES algorithm (lines 8-9). It then updates the covariance matrix and step-size based on the p_c and p_σ . All the update functions ($update_{p_\sigma}$, $update_{p_c}$, $update_C$, and $update_\sigma$) used in this algorithm are adopted from a recent work (Auger and Hansen, 2012). After max_{iter} number of iterations, the algorithm generates the optimal $\{u_{opt_hybrid}, I_{opt_hybrid}\}$ pair that results in the minimum cost (line 13) along with its corresponding on-chip temperature, $T_{hs_opt_hybrid}$. Similarly, we apply CMA-ES algorithm

to liquid cooling via microchannels to get the optimal I_{liquid} , $T_{hs_opt_liquid}$. We compare $T_{hs_opt_hybrid}$ and $T_{hs_opt_liquid}$ to T_{limit} , respectively, and select the cooling solutions that satisfy the temperature constraint. Finally, we compare the cooling power of the selected solutions to output the one with the minimum cooling power as the optimal choice.

Algorithm 3: CMA-ES

Input : Number of samples per iteration, λ
Input : Objective function (Equation (5.1)), $cost$
Input : Maximum number of iterations, max_{iter}
Initialize: Sampling distribution mean, μ
Initialize: Step-size, σ
Initialize: Covariance matrix, $C = Identity\ matrix$
Initialize: Cumulation for σ and C , $p_{\sigma} = 0$, $p_c = 0$
Initialize: $k = 0$

- 1 **while** $k < max_{iter}$ **do**
- 2 **for** i in $1 \dots \lambda$ **do**
- 3 $\{u, I\}_i = \mathcal{N}(\mu, \sigma^2 C)$
- 4 $cost_i = Thermal\ Simulation(\{u, I\}_i)$
- 5 Sort $\{u, I\}$ based on increasing $cost$ (objective function)
- 6 $\mu' = \mu$
- 7 $\mu = \sum_{i=1}^{\lambda/2} (w_i \{u, I\}_i)$
- 8 $p_{\sigma} = update_{p_{\sigma}}(p_{\sigma}, \sigma^{-1} C^{-1/2} (\mu = \mu'))$
- 9 $p_c = update_{p_c}(p_c, \sigma^{-1} (\mu = \mu', ||p_{\sigma}||))$
- 10 $C = update_C(C, p_c, (\{u, I\}_1 - \mu)/\sigma, \dots, (\{u, I\}_{\lambda} - \mu)/\sigma)$
- 11 $\sigma = update_{\sigma}(\sigma, ||p_{\sigma}||)$
- 12 $k++$
- 13 Generate $\{u_{opt_hybrid}, I_{opt_hybrid}\}, P_{cooling_opt_hybrid}$, and $T_{hs_opt_hybrid}$ based on the last iteration simulation results

To demonstrate the optimization results of the proposed optimization flow, We select three floorplans with a various number of hot spots and hot spot power densities as shown in Figure 5-3. For each of the floorplan, we set the background power density to $50\ W/cm^2$ with hot spot power densities of $\{100, 300, 1000, 1700, 2000\}\ W/cm^2$. We use the aforementioned emerging cooling methods as heat sinks or inter-layer cooling methods. The on-chip maximum temperature limit (temperature constraint) is set to $65^{\circ}C$. The detailed experi-

mental setup can be found in the previous work (Yuan et al., 2020; Yuan et al., 2019a). To select the optimal cooling solutions and cooling parameters, we use CMA-ES, MSA, and grid search to select the optimal cooling parameters for the aforementioned emerging cooling technologies (Yuan et al., 2020; Yuan et al., 2019a). We summarize the results in Figure 5-4.

As shown in Figure 5-4, since two-phase VCs are passive cooling methods (no additional power is needed on the evaporator side), for relatively low power density (100, 300, and 1000 W/cm^2), two-phase VCs with hybrid wick evaporators completely beat other cooling methods. Note that, compared to the hybrid wick, the micropillar wick evaporator cannot provide enough HTC on the evaporator side to cool down the chip due to the low dry-out limit. Liquid cooling via microchannels are not able to provide enough power to remove the high heat flow generated by the chip. Since hybrid cooling has finer control over the cooling power and cooling ability, hybrid cooling always results in lower cooling power compared to only using TEC. For high power density (1700 and 2000 W/cm^2), since hybrid cooling and TECs aim to remove the hot spot heat, hybrid cooling is the optimal cooling method in these cases. We also carry out experiments with a temperature constraint of 90°C . Still, since two-phase VCs with hybrid wick evaporators are passive cooling methods, they beat other emerging cooling technologies in terms of cooling efficiency. Since two-phase VCs with hybrid wick evaporators and hybrid cooling (of liquid cooling via microchannels and TECs) achieve the optimal cooling efficiency among all the aforementioned cooling technologies. Therefore, we only discuss the optimization flow for these two cooling technologies in the next section.

5.4 Emerging Cooling Methods Optimization via DL

The existing cooling optimization methods (CMA-ES and MSA) have two main issues: (i) need to run a significant number of thermal simulations, which results in large simulation

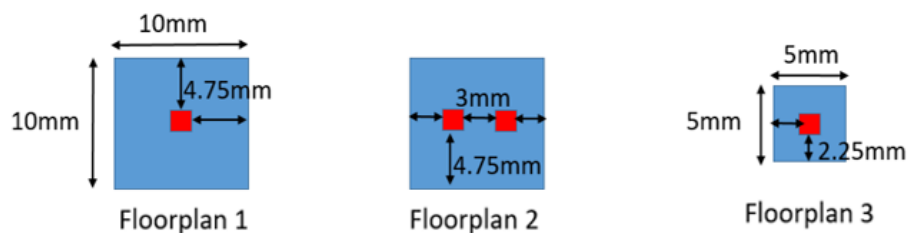


Figure 5.3: Synthetic chip floorplans.

■ Two-phase VCs with hybrid wick evaporators
 ■ Hybrid cooling

PD/Floorplan	Floorplan#1	Floorplan#2	Floorplan#3
100	(R245fa, 54.34°C, 0W)	(R141b, 54.57°C, 0W)	(R245fa, 54.47°C, 0W)
300	(R245fa, 55.74°C, 0W)	(R141b, 56.07°C, 0W)	(R245fa, 55.95°C, 0W)
1000	(R245fa, 60.64°C, 0W)	(R141b, 60.96°C, 0W)	(R245fa, 60.55°C, 0W)
1700	(1m/s 0.7A, 64.73°C, 0.137W)	(1m/s 1.0A, 64.43°C, 0.165W)	(0.9m/s 0.9A, 64.33°C, 0.12W)
2000	(1.2m/s 0.9A, 64.97°C, 0.205W)	(1.2m/s 1.0A, 64.85°C, 0.215W)	(1.1m/s 1.0A, 64.18°C, 0.19W)

Figure 5.4: Results for on-chip temperature constraint = 65°C. The format for two-phase VCs with hybrid wick evaporator is {coolant, hot spot temperature, cooling power}. The format for hybrid cooling is {liquid flow velocity, TEC current, hot spot temperature, cooling power}.

time, and (ii) there is no guarantee that the selected cooling method and its cooling parameters are optimal. The accuracy of the optimization result selected by CMA-ES and MSA is determined by the sample size and the number of iterations (Yuan et al., 2020; Yuan et al., 2019a). Using the DL model, specifically, the CNN regression model, to predict the optimal cooling solution and its cooling parameters could be the solution to these two issues. A DL regression model is able to learn the intrinsic information among the chip designs and the cooling solutions and then efficiently generate the optimal cooling solution and the cooling parameters, given a specific chip floorplan and power profile. This section demonstrates a step towards this goal by using a multi-output CNN regression model to

estimate the best cooling method and its cooling design and technology parameters. This CNN-based optimization flow requires the CNN regression model to be sufficiently modular for all chip floorplans and power profiles. If the input chip floorplan and power profile change, the predicted cooling solution and the cooling parameters should still maintain the desired accuracy.

5.4.1 Overall CNN Optimization Architecture

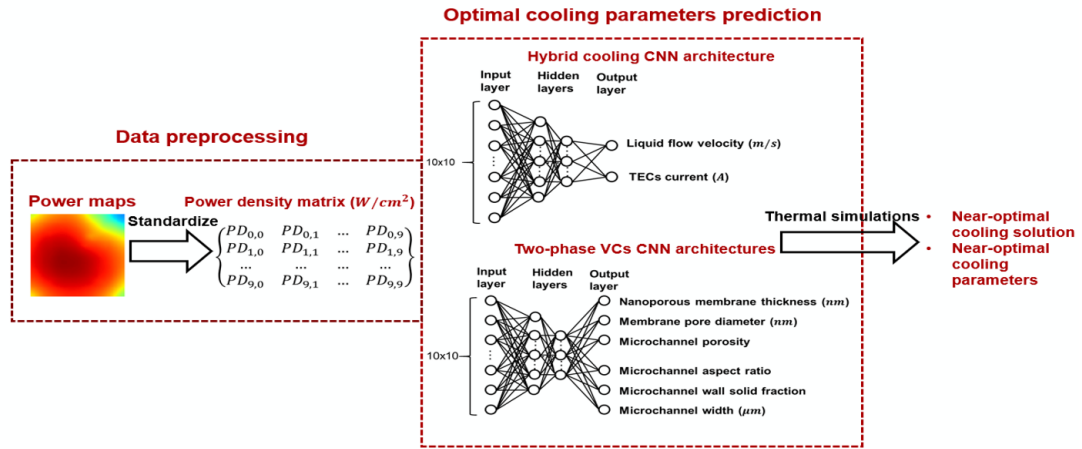


Figure 5-5: DL-based cooling optimization flow.

The overall CNN optimization architecture is shown in Figure 5-5. Given an arbitrary chip power map, the optimization flow standardizes the power map into a 10×10 power density matrix. The power density matrix is used as the input to the hybrid cooling and two-phase VCs CNN architectures to predict the optimal cooling parameters for these two cooling technologies, respectively. The optimization flow then conducts thermal simulations for the input power map using hybrid cooling and two-phase VCs with hybrid wick evaporators as the cooling method and compares the hot spot temperatures and the cooling cost to determine the optimal cooling method and its cooling parameters.

As the solution space of this cooling optimization problem is continuous instead of discrete, the optimization results found by black-box optimization methods (e.g., exhaustive

search, simulated annealing, or others) can only be near-optimal since they require discretizing the optimization inputs. Therefore, unless an accurate mathematical formula is created for such an optimization problem, the accuracy of the optimization methods will always depend on the input granularity, and the outputs are only near-optimal. It is not possible to create an accurate mathematical formula for this particular cooling optimization problem to solve it analytically; thus, we consider the output of our proposed DL-based optimization framework as optimal given the constraints. The accuracy of the proposed optimization flow depends on the granularity of the training data in the cooling parameter solution space.

5.4.2 Training Data Preparation

The CNN architectures shown in Figure 5-5 require a massive amount of data to optimize the parameters within the neural network to improve the performance of the model. One major challenge of building a CNN architecture is preparing the training data. Since real processors' power maps are hard to obtain, we generate a comprehensive training dataset using statistical distribution. We select a $5\text{ mm} \times 5\text{ mm}$ chip and divide the chip uniformly into 10×10 power density grids. To generate comprehensive and realistic power density maps, we choose to use Gamma distribution to generate power density for each power density grid randomly. The reason we use gamma distribution to generate random power density maps are as follows: (i) obtaining real processors' power maps is hard, (ii) using real processors' power maps may let the CNN architectures overfit the training power maps of the chips, (iii) training power density maps may not cover corner cases, and iv) the generated power density value should be positive and most generated values should within the background power density range of $50\text{-}200\text{ W/cm}^2$. The largest power density value that the selected Gamma distribution generates is 2000 W/cm^2 . We then apply data augmentation techniques to rotate and flip the power density maps to increase the training data size.

The total number of power density maps we generated is 90000. For each generated power density map, we need to know the optimal cooling parameters of using hybrid cooling and two-phase VCs with hybrid wick evaporators. For hybrid cooling, we apply TEC units to the power density grids that have values of more than 200 W/cm^2 . The microchannel width is set to be $50 \mu\text{m}$. We use Equation 5.1 as the optimization objective function and run grid searches to determine the optimal liquid flow velocity and TEC current for each power density map. We select water, R245fa, and R141b as the coolants for two-phase VCs with hybrid wick evaporators. We directly run grid searches with a finer granularity for each power density map to determine the optimal cooling parameters for each coolant.

5.4.3 Hybrid Cooling CNN Architecture

The hybrid cooling method combines the liquid microchannel and TEC layers into one chip stack. But since they are completely different cooling methods with different cooling performance and cooling power, the liquid flow velocity and TEC current are independent. In this case, we create two branches in this CNN architecture, and each branch is responsible for predicting the optimal values for either liquid flow velocity or TEC current. Both branches share the same input layer and have the same number of layers and parameters. However, since this is a multi-output CNN, the loss for each branch is different. To achieve the best regression accuracy, we build different multi-output CNN architectures with different kernel sizes, number of filters, number of convolutional layers, number of fully connected layers, and with or without batch normalization layer, and select the one that has the highest validation accuracy. Table 5.1 shows the details of three alternative CNN architectures of hybrid cooling. To evaluate the accuracy of the CNN alternatives, we divide the 90000 training power density maps into the training set and validation set. The total number of training power density maps is set to 72000, and the validation set is set to 18000. All the input power matrices are normalized with respect to the mean and standard derivation of the training data. We show the accuracy results of three CNN alternatives in

Table 5.2. We use the Adam optimizer to train these multi-output CNN architectures, and the loss function is selected as the mean square error (MSE). As we observe from Table 5.2. Model_1 is overfitting with the data since the validation accuracy is at least 3.5% lower than the training accuracy. To prevent overfitting, we add additional dropout layers and increase the dropout rate to 0.5. However, the model accuracy and R2 scores start to decrease below 85%. In this case, we decide to lower the complexity of the model by using fewer convolutional layers and fully connected layers, which results in model_2. The accuracy and R2 score of model_2 show that the model and the data are not closely correlated since the R2 scores are below 90%. To improve the model accuracy and R2 score, we add additional fully connected layers with additional neurons and result in model_3. As we observe from the results of model_3, the accuracy and R2 scores are higher than other alternatives, and the model itself is not overfitted. Therefore, we choose model_3 (as shown in Figure 5-6) as our hybrid cooling CNN regression model. In addition, we also experiment with different activation functions such as ReLU, Hyperbolic tangent, and Leaky ReLU. We compare the accuracy and R2 score of model_3 with ReLU, Hyperbolic tangent, and Leaky ReLU. To ensure the predicted parameter is greater than 0, we set the activation function of the last activation layer to be ReLU. We summarize the results in Table 5.3. Since ReLU achieves the highest accuracy and R2 score, so we select ReLU as our activation function for all the model's activation layers in model_3.

Table 5.1: Details parameters for hybrid cooling CNN alternatives.

	# of Conv2D	# of Batch normalization	# of Dropout	# of Dense	Kernel Sizes
Model_1	6	6	2	8	4×4
Model_2	2	2	2	2	5×5
Model_3	2	0	2	6	5×5

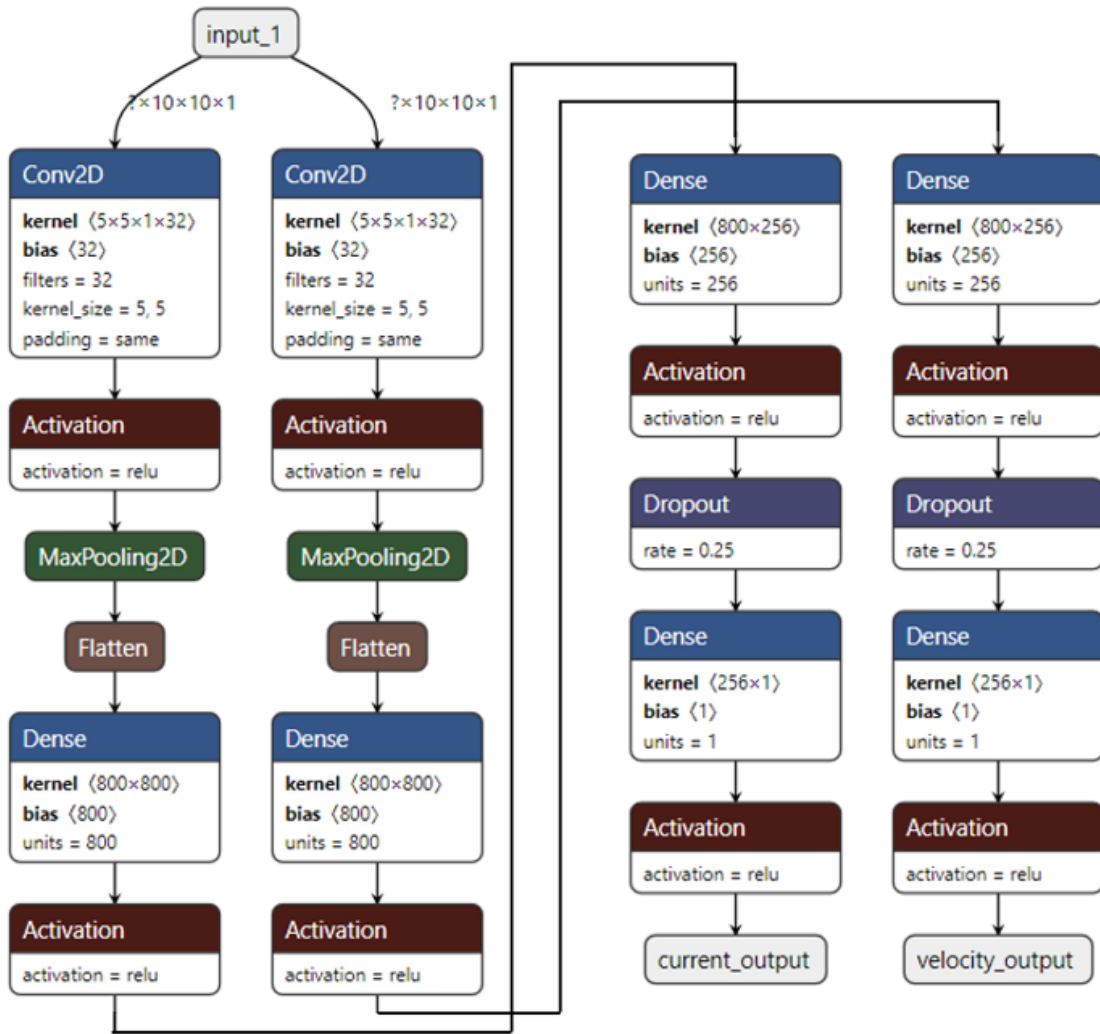


Figure 5-6: Optimal hybrid cooling CNN architecture.

Table 5.2: Accuracy results for different hybrid cooling CNN regression alternatives.

	Model_1		Model_2		Model_3	
	Velocity	Current	Velocity	Current	Velocity	Current
Training Accuracy	94%	95%	89.9%	90.5%	98.9%	98.5%
Validation Accuracy	89%	91.5%	88.8%	89.8%	97.7%	97.9%
Validation R2	90%	91%	88%	89%	93.2%	96%

5.4.4 Two-Phase VCs with Hybrid Wick Evaporators CNN Architecture

Since two-phase VCs with hybrid wick evaporators have six different cooling parameters, there are six different branches for this cooling technology. In addition, since different

Table 5.3: Accuracy results for different activation functions. Accuracy and R2 score are averaged for liquid flow velocity and current.

	ReLU	Hyperbolic Tangent	Leaky ReLU
Validation accuracy	97.8%	95.6%	96.9%
R2	94.6%	93.3%	93.7%

coolant has different cooling properties, it's not realistic to train only one CNN model to predict both the optimal cooling parameters and the coolant. To solve this problem, we train different multi-output CNNs for different coolants and conduct thermal simulations at the end to find out the optimal coolant and its cooling parameters. Compared to hybrid cooling CNN architecture, two-phase VCs with hybrid wick evaporators CNNs also need to consider the dry-out effect. To improve the prediction accuracy, we add additional convolutional layers in each branch, and the number of filters in each convolutional layer is doubled compared to hybrid cooling CNN architecture. We also build different CNN alternatives for each two-phase VCs with hybrid wick evaporators CNN with different coolants. We summarize 9 CNN alternatives' parameters in Table 5.4. For each CNN alternative, we change the Dropout layers from 6 to 36 and the dropout rate from 0.25 to 0.5 to prevent overfitting. After each Convolutional layer, we add batch normalization to stabilize the training process and improve the training time. After all the convolutional layers, we add one Max Pooling layer to decrease the problem size. We use RMSprop as the optimizer with a learning rate of 0.001, and the loss function for each branch is set to MSE. To evaluate the accuracy of the CNN alternatives, we divide the 90000 training power density maps into the training set and validation set. The total number of training power density maps is set to 72000, and the validation set is set to 18000. All the input power matrices are normalized with respect to the mean and standard derivation of the training data. We show the average accuracy results of these CNN alternatives for cooling parameters in Table 5.5. We always start with the most complex model, and we aim to simplify the CNN by using fewer convolutional layers and fully connected layers. We select model_3 as our final two-phase

VCS with hybrid wick evaporators CNN model for each of the coolants. We also select the activation functions to be ReLU, Hyperbolic tangent, and leaky ReLU. Since ReLU results in the highest accuracy, we set ReLU as our activation function for all the activation layers.

Table 5.4: Detailed parameters for two-phase VCs with hybrid wick evaporators CNN alternatives.

	Coolants	# of Conv2D	# of Batch normalization	# of Dropout	# of Dense	Kernel Sizes
Model_1	Water	36	36	36	18	4×4
Model_2	Water	6	6	6	6	5×5
Model_3	Water	12	12	6	18	5×5
Model_1	R141b	36	36	36	18	4×4
Model_2	R141b	6	6	6	6	5×5
Model_3	R141b	12	12	6	18	5×5
Model_1	R245fa	36	36	36	18	4×4
Model_2	R245fa	6	6	6	6	5×5
Model_3	R245fa	12	12	6	18	5×5

Table 5.5: Accuracy results for different two-phase VCs with hybrid wick evaporators CNN regression alternatives.

	Water			R141b			R245fa		
	Model_1	Model_2	Model_3	Model_1	Model_2	Model_3	Model_1	Model_2	Model_3
Training Accuracy	99.5%	96%	98.9%	98.5%	95.8%	99.3%	99.5%	95.3%	98.5%
Validation Accuracy	95.8%	95.7%	98.8%	96.7%	93.4%	98.83%	94.8%	92.7%	97.7%
Validation R2	95.7%	93.2%	98.6%	95.7%	92.1%	98.3%	95.7%	93.2%	98.1%

5.4.5 Results and Discussions

In this section, we first discuss the validation results of the proposed CNN architectures. Next, we demonstrate the efficiency of using our proposed DL-based cooling optimization flow against existing cooling optimization methods on realistic MPSoCs from OpenROAD (Ajayi et al., 2019) and IBM Power9 processor (Sadasivam et al., 2017).

Validation of the Proposed CNN Architectures

To validate the accuracy of the CNN architectures discussed in the previous section. We divide the 90000 training power density maps into the training and validation sets. The total number of training power density maps is set to 72000, and the validation set is set to 18000. All the input power matrices are normalized with respect to the mean and standard derivation of the training data. We summarize the validation MSE, mean absolute error (MAE), and R2 score in Table 5.6. For two-phase VCs with hybrid wick evaporators, since each coolant has its own CNN architecture, we average the error of two-phase VCs with hybrid wick evaporators' geometries for each coolant. As we observe from Table 5.6, our proposed CNN architectures can properly learn patterns to predict the optimal cooling parameters for each type of cooling technology. Compared to two-phase VCs with hybrid wick evaporator CNN, hybrid cooling CNN has higher MSEs and MAEs with lower R2 scores. The reason is that hybrid cooling is more complex because of the optimization objective function. Hybrid cooling CNN also needs to consider cooling power, making the prediction more complicated and less accurate. Whereas in two-phase VCs with hybrid wick evaporators, there is no additional cooling power at the evaporator side.

Table 5.6: Validation results of the proposed CNN architectures.

Metrics	Liquid flow velocity	TEC current	t	dp	ϕ	AR	SF	w
MSE	2.3%	2.1%	0.9%	0.6%	2.5%	0.7%	1%	2.1%
MSE	5.3%	4.7%	2%	1.5%	4%	2%	2%	4%
R2	93.2%	96%	96%	99.3%	99.2%	99.2%	99.3%	98.5%

Optimization Results for Realistic MPSoCs

To demonstrate the predicted accuracy and search time improvements on realistic chips of our proposed optimization flow against existing cooling optimization methods (Yuan et al., 2020; Yuan et al., 2019a), we select realistic high power density MPSoCs from OpenROAD (Ajayi et al., 2019) with different chip sizes, floorplans, and power profiles

to test the proposed DL-based cooling optimization framework. We compare the optimal results predicted using our proposed CNN architectures and optimization flow against MSA and CMA-ES from previous work with a temperature constraint of 90°C. The statistics of the realistic MPSoCs from the OpenROAD have listed in Table 3.9. For each MPSoC, we first map the power profiles into 10×10 power density maps. We then use Equation (5.2) to standardize the power density maps with respect to the training power density maps:

$$PD_{new} = (PD_{original} - \mu_{training}) / std_{training} + b. \quad (5.2)$$

$\mu_{training}$ is the mean power density of the training dataset, $std_{training}$ is the standard derivation of the training power density dataset, b is the bias which is defined as the ratio of the testing MPSoCs dimension over the training chip dimension.

For each cooling parameter, we calculate the average and max error for all MPSoCs and coolants and plotted the percentage error in Figure 5.7. The Avg_{Error} and Max_{error} are defined as shown in Equation (5.3):

$$Avg_{Error} = (\sum p_{pred} - p_{base}) / (\#ofMPSoCs \times \#ofcoolants),$$

$$Max_{Error} = \max(\sum p_{pred} - p_{base}), \quad (5.3)$$

where p_{pred} is the predicted parameter by our proposed CNN architectures, and p_{base} is the parameter generated using the baseline method (CMA-ES and MSA). Both CMA-ES and MSA have been validated against grid search in previous work (Yuan et al., 2020; Yuan et al., 2019a). We choose the baseline method to be CMA-ES and MSA instead of grid search because we seek to have a fast design exploration and simulation time for the baseline method, which would further show the simulation speed improvement of our proposed CNN architectures. Since the coolant is only water for hybrid cooling, $\#ofcoolants$ equals 1. For two-phase VCs, the $\#ofcoolants$ is set to 3 because there are three available coolants (water, R245fa, and R141b). As we see in Figure 5.7, our proposed CNN architectures suc-

cessfully predict optimal cooling parameters for hybrid cooling and two-phase VCs with hybrid wick evaporators with a maximum error of less than 4%. Since PicoSoC with 95% utilization has the highest power density, we also show the predicted parameters using our proposed CNN architectures and baseline parameters generated using the baseline methods of PicoSoC in Table 5.7.

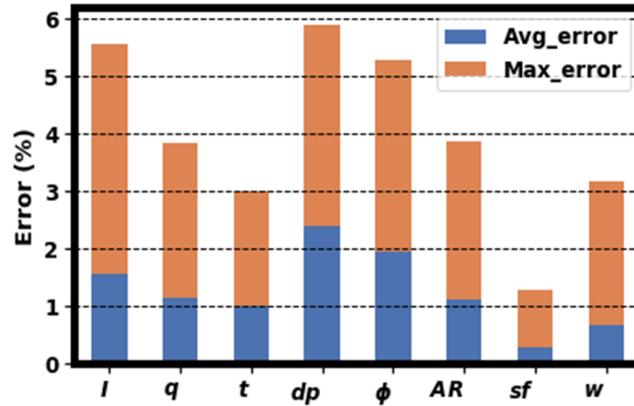


Figure 5.7: CNN architectures accuracy results.

Table 5.7: Predicted parameters using our proposed CNN architectures and baseline parameters generated using the baseline methods for PicoSoC.

Methods	Coolant	$I(A)$	$q(m/s)$	$t(nm)$	$dp(nm)$	ϕ	AR	sf	$w(\mu m)$
CNN	Water	7	2.57	0.97	0.17	0.30	1.92	0.34	7.63
Baseline	Water	6.98	2.57	0.99	0.175	0.29	1.85	0.33	7.56

Figure 5.8 shows the optimization results correlation plots for all MPSoCs. The proposed DL-based cooling optimization flow is able to find a similar optimal cooling solution and its cooling parameters with maximum temperature and cost difference of $0.7^{\circ}C$ and $0.01 W$ compared to existing methods. Note that the tested MPSoCs have different chip dimensions compared to the training chip size we are using, demonstrating that our proposed CNN architectures are able to predict the optimal cooling parameters for any given chip size and power profile. For large-size chips such as PicoSoC, Sparc, and Black_parrot, the optimal solution is always two-phase VCs with hybrid wick evaporators because it does

not consume additional power on the evaporator side. Two-phase VCs with hybrid wick evaporators cannot efficiently spread the heat across the chip for smaller chips with high power density. That is the reason for Swerv MPSoCs, the optimal cooling solution is hybrid cooling. In addition, all the predicted geometries are within the valid range, and all the two-phase VCs with hybrid wick evaporators' geometries satisfy the dry-out constraint. The average search time for the baseline method (MSA and CMA-ES) is 1.57 hours, while it only takes the proposed DL-based cooling optimization flow 50 seconds at most to predict the optimal cooling method and its cooling parameters. Our proposed DL-based cooling optimization flow achieves a maximum of $140\times$ speedup compared to existing optimization methods. In addition, the training time for hybrid cooling CNN is 13.3 minutes (21 seconds per epoch) and the maximum training time for two-phase VCs CNN is 18 minutes (56 second per epoch). The worst-case training and inference time is calculated based on Equation (5.4):

$$Time_{worst} = \max(Hybrid_{train} + Hybrid_{infer}, \max(VC_{train} + VC_{infer})), \quad (5.4)$$

where $Hybrid_{train}$ and $Hybrid_{infer}$ are the training time and inference time for hybrid cooling, respectively. VC_{train} is the training time for two-phase VCs CNN architectures with different coolants. VC_{infer} is the inference time for two-phase VCs CNN architectures with different coolants. The worst-case training and inference time for the proposed CNN architectures is 18.83 minutes, and the overall speedup compared to the baseline method is $5\times$.

Optimization Results for the IBM Power9 Processor

To further investigate the prediction accuracy of the proposed CNN optimization architectures, we model the IBM Power9 high-performance processor with a total chip power of 190 W (Sadasivam et al., 2017). The floorplan of the IBM Power9 processor is shown

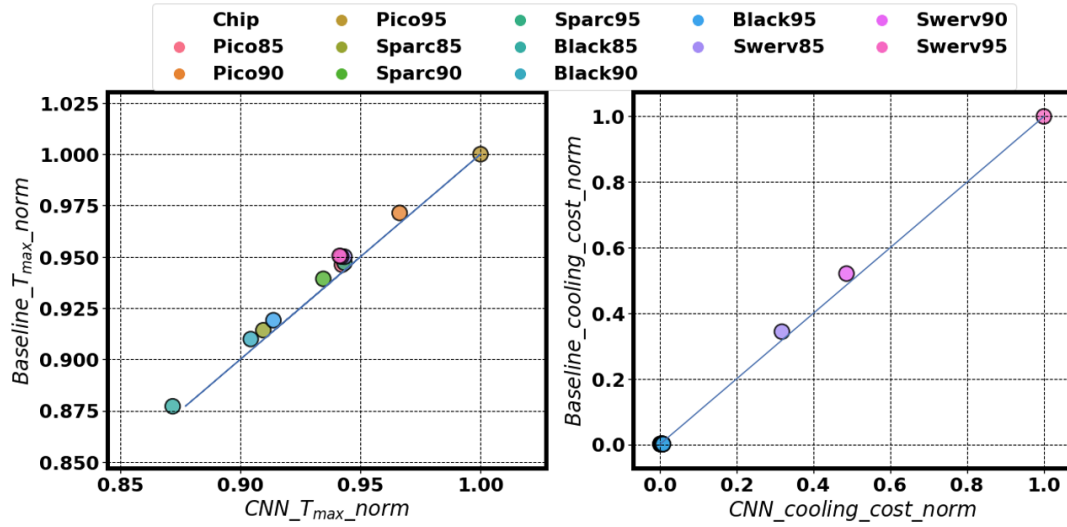


Figure 5-8: The correlation plots of the maximum temperatures and cooling costs predicted using the proposed optimization flow against the baseline methods' results. Baseline methods stand for the combination of MSA and CMA-ES. CNN stands for the proposed CNN architectures and optimization flow. All the data are normalized to the maximum value.

in Figure 3-23 and the power breakdown is shown in Table 5.8. We compare the optimal results predicted using our proposed CNN architectures and optimization flow against the baseline method (MSA and CMA-ES) with a temperature constraint of 90°C . We use Equation (5.2) to standardize the power density maps with respect to the training power density maps. The comparison results are shown in Table 5.9. The maximum cooling parameter difference is less than 3.8°C . Since the dry-out heat flux is negatively correlated with the chip size, the dry-out heat flux decreases dramatically as the chip size increases. In this case, both proposed CNN architectures and baseline methods cannot find optimal cooling parameters for two-phase VCs to optimize the maximum temperature under 90°C . Therefore, the optimal cooling solution has to be hybrid cooling. We also observe that, for a small chip with fewer hot spots, compared to the power consumption of the chip, the cooling cost is not significant. However, there will be more liquid microchannels and TEC units for high-power chips with large chip sizes and more hot spots (such as IBM Power9).

Therefore, the cooling cost starts to become significant. As shown in Table 5.9, the cooling power is around 10% of the total chip power.

Table 5.8: IBM Power9 processor power breakdown.

Components	Core(total)	Cache	Nest	I/O	DDR4
Power(W)	133	20.9	9.5	15.2	11.4

Table 5.9: IBM Power9 processor optimal cooling parameters, maximum temperature, and cooling power.

Methods	Coolant	$I(A)$	$q(m/s)$	$T_{max}(^{\circ}C)$	$Power_{cooling}(W)$
CNN	Water	7	2.59	89.9	19.50
Baseline	Water	7	2.6	89.88	19.83

Chapter 6

Conclusions and Future Work

Existing on-chip cooling solutions are not sufficient to mitigate high-power-density hot spots and can result in over/under-cooling. Emerging cooling solutions such as liquid cooling via microchannels, TECs, and two-phase cooling can achieve better cooling performance against existing cooling solutions. To facilitate design exploration and optimization for emerging cooling solutions, in this thesis, we first discuss a parallel compact thermal simulator, PACT, to enable fast and accurate parallel thermal simulations. We then demonstrate the potential of combining compact modeling methodology and ML to speed up the thermal simulation speed for two-phase cooling. Finally, we present a novel DL-based cooling solution and cooling parameters optimization flow to select the optimal cooling solution and corresponding cooling parameters for any chip floorplan and power profile.

6.1 Enabling Fast and Accurate Parallel Thermal Simulations with PACT

Thermal analysis is an essential step that enables the co-design of the computing system (i.e., ICs and computer architectures) with the cooling system (e.g., heat sink). Existing thermal simulation tools are limited by several major challenges that prevent them from providing fast solutions to large problem sizes necessary to conduct standard-cell-level thermal analysis or to evaluate new technologies or large chips. We present PACT that enables fast and accurate standard-cell-level to architecture-level steady-state and transient thermal simulations to overcome these challenges. PACT can be easily extended to support emerging integration and cooling technologies and is compatible with popular architecture-

level performance and power simulators. To demonstrate the extensibility of PACT, we integrate two types of heat sinks, a model for layers with heterogeneous materials and a CTM for liquid cooling via microchannels in PACT. We also use PACT to build a PNoC simulation framework with Sniper and McPAT to show compatibility. In addition, we also create an interface between PACT and OpenROAD that can be used to evaluate the thermal behavior of full industrial designs. Compared to COMSOL, PACT has a maximum temperature error of 2.77% for steady-state and 3.28% for transient simulation. Compared to HotSpot, PACT can achieve up to $232\times$ speedup.

The current version of PACT only supports the cuboid grid. Other grid shapes, such as circular (helpful in simulating round heat pipes), can only be approximated using several cuboid grids. However, this process can be done manually for one circular grid and automated for all the grids across the design. Also, the current version of PACT does not support an adaptive grid (non-uniform grid), and this feature can be added in the later versions of PACT.

Currently, PACT does not envision the quantum effects on the nanometer scale (40-300 *nm* (Varshney et al., 2019)). To guarantee the simulation accuracy of PACT, the minimum grid size has to be higher than $300\times 300\text{ nm}^2$. For sub-14 *nm* technology, users have to combine several normal cells into one grid node to conduct thermal simulations. Otherwise, the thermal dissipation will be dominated by the ballistic transportation of acoustical phonon, and the overall simulation accuracy will be affected (Varshney et al., 2019). An open design problem for PACT is to consider the quantum effect in the nanometer scale and use the Boltzmann transport equation to model nanometer-scale phonon effects.

Because of the fast simulation speed and the high extensibility of PACT, users can use PACT to develop a co-design and co-optimization flow for the computing system and cooling system. Depending on the design target and optimization object function, the co-design and co-optimization flow can either achieve better energy efficiency and cooling

cost-effectiveness or higher computing performance within the temperature limit.

6.2 Modeling Emerging Cooling Methods via Machine Learning

Two-phase cooling technologies are attractive because of the high heat transfer rate compared to traditional cooling methods. However, existing thermal models for two-phase cooling often include CFD simulations, which incur large memory requirements and long simulation time. In this thesis, we present ML-enabled fast and accurate compact thermal modeling methodologies for two-phase VCs. We first elaborate on CTMs for two-phase VCs with micropillar and hybrid wick evaporators. We validate our proposed model against COMSOL, and our proposed model achieves a maximum error of less than 1.25°C and a speedup of up to $214\times$ compared to COMSOL models. We next introduce an ML-based temperature-dependent HTC simulation framework for two-phase cooling. This framework can enable fast and accurate thermal simulations for two-phase cooling technology with a wide range of cooling parameters. Compared to COMSOL, our simulation framework with CTM achieves a $21\times$ speedup with an average accuracy loss of less than 0.98°C . We extend the ML-based simulation framework by modeling a real VC, supporting additional ML regression models, and adding transient modeling ability. Finally, we introduce a systematic way of predicting temperature profiles based on on-chip digital thermal sensor readings via ML. The proposed approach achieves an accuracy of 99.98% with minimal overhead.

Future directions in this area include validating the proposed ML-based simulation framework against real two-phase cooling test vehicles, evaluating the temperature profile prediction methodology using real chips, and supporting transient heat map prediction using ML.

First, the current ML-based temperature-dependent HTC simulation framework has been validated against the COMSOL model. The training data of the ML regression models in the framework are collected from the simulations of the COMSOL model. The proposed

framework needs to be validated against actual two-phase cooling test vehicles to demonstrate the practicability of the CTMs and ML-based simulation framework.

Second, the current temperature profile prediction method is simulation-based. The training data are collected from architectural performance, power, and thermal simulators. The proposed ML models need to be evaluated on an actual chip to demonstrate their applicability. In addition, some calibration methods may need to be included to obtain a more accurate temperature profile predicted using the proposed method when evaluating on real chips.

Third, the current methodology of predicting temperature profiles using ML only supports the prediction of steady-state heat maps. In contrast, in reality, the control knobs of runtime thermal control policies such as thermally-aware dynamic voltage frequency scaling often rely on the instantaneous temperature readings of the thermal sensors. Therefore, the current method needs to be extended and support transient heat map prediction.

6.3 Optimizing Emerging Cooling Methods for High-Performance Processors via Deep Learning

Various emerging cooling methods, such as liquid cooling via microchannels, TECs, two-phase VCs, and hybrid cooling options, have been designed to efficiently remove heat from high-performance processors. Selecting the optimal cooling solution for a given chip and determining the optimal cooling parameters for that solution to achieve high efficiency are open problems. These problems are, in fact, computationally expensive due to the massive space of possible solutions. We introduce a DL-based cooling optimization flow for emerging cooling technologies to address this design challenge. We demonstrate the efficiency of using DL techniques to optimize the cooling technologies against existing work. Our proposed CNN architectures and DL-based cooling optimization flow can successfully predict the optimal cooling solution and cooling parameters with a maximum error of less than 4% and a maximum speedup of $140\times$.

Future directions in this area include building a more comprehensive cost function, determining the optimal cooling solution and cooling parameters more broadly for new integration methods, using finer granularity power density maps to train more accurate CNN architectures, and designing deep runtime reinforcement learning control policies for emerging cooling solutions.

First, the cost function we used for this DL-based cooling optimization flow only considers the cooling performance and cooling power. Whereas in reality, the manufacturing cost of the cooling method is also one crucial factor. The current cost function can be improved by including the manufacturing cost of the cooling solution. In addition to that, to further improve the prediction accuracy, the CNN architectures can be modified to take other design parameters such as chip sizes and layer thickness into consideration. These parameters will be used as the input to the CNN.

Second, our proposed DL-based cooling optimization flow is not comprehensive enough to cover emerging integration technologies, such as 3D ICs with arbitrary layer configurations. Our current flow applies if the 3D IC layer configurations (i.e., which blocks are allocated on which layers) match the layer configurations available in the training data. For using the proposed CNN optimization architecture for arbitrary 3D IC designs, the CNN regression models have to be retrained as needed to maintain the desired accuracy. The layer partitioning configurations can be considered as inputs to the CNN regression models to tackle this limitation in future work.

Third, the power maps we used to train the CNN architectures are coarse granularity (10×10). To achieve a better accuracy, a more fine granularity power map can be used to train the CNN architectures. To optimize both the training accuracy and training time, there needs to be a systematic way to simultaneously select the power map's optimal granularity to satisfy the training accuracy and training time requirements.

Finally, we only consider design time cooling optimization. However, the power density

matrix may change during runtime due to the application behavior. Deep reinforcement can potentially predict the optimal cooling parameters at runtime efficiently. The future work of DL-based cooling optimization flow includes developing deep reinforcement learning-based runtime cooling parameter control policies for emerging cooling technologies such as liquid cooling via microchannels, TECs, and hybrid cooling.

References

- (1998). Introduction to comsol multiphysics®, multiphysics, comsol. *COMSOL Multiphysics, Burlington, MA*. <https://cdn.comsol.com/doc/5.5/IntroductionToCOMSOLMultiphysics.pdf>.
- Adera, S., Antao, D., Raj, R., and Wang, E. N. (2016). Design of micropillar wicks for thin-film evaporation. *International Journal of Heat and Mass Transfer*, 101:280–294.
- Adhinarayanan, V., Paul, I., Greathouse, J. L., Huang, W., Pattnaik, A., and Feng, W.-c. (2016). Measuring and modeling on-chip interconnect power on real hardware. In *2016 IEEE International Symposium on Workload Characterization (IISWC)*, pages 1–11.
- Ajayi, T., Chhabria, V. A., Fogaça, M., Hashemi, S., Hosny, A., Kahng, A. B., Kim, M., Lee, J., Mallappa, U., Neseem, M., Pradipta, G., Reda, S., Saligane, M., Sapatnekar, S. S., Sechen, C., Shalan, M., Swartz, W., Wang, L., Wang, Z., Woo, M., and Xu, B. (2019). Toward an open-source digital flow: First learnings from the openroad project. In *2019 ACM/IEEE Design Automation Conference (DAC)*, 2019, pages 1–4. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8806957>.
- Allec, N., Hassan, Z., Shang, L., Dick, R. P., and Yang, R. (2008). ThermalScope: Multi-scale thermal analysis for nanometer-scale integrated circuits. In *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, pages 603–610.
- Auger, A. and Hansen, N. (2012). Tutorial cma-es: evolution strategies and covariance matrix adaptation. In *GECCO '12: Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 827–848. <https://dl.acm.org/doi/proceedings/10.1145/2330784>.
- Bailey, D. H. et al. (1991). The NAS parallel benchmarks. *International Journal of Supercomputing Applications*, 5(3):63–73.
- Bao, M., Andrei, A., Eles, P., and Peng, Z. (2009). On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration. In *Proc. of 46th ACM/IEEE Design Automation Conference*, pages 490–495.
- Baraya, K., Weibel, J. A., and Garimella, S. V. (2020). Effective anisotropic properties-based representation of vapor chambers. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 11(1):51–56.

- Beigi, M. V. and Memik, G. (2016). Therma: Thermal-aware run-time thread migration for nanophotonic interconnects. In *Proc. of the International Symposium on Low Power Electronics and Design (ISLPED)*, page 230–235, New York, NY, USA. Association for Computing Machinery.
- Biolek, D. and Biolek, Z. (2014). Fourth fundamental circuit element: SPICE modeling and simulation. In Tetzlaff, R. (ed.) *Memristors and Memristive Systems*, pages 105–162. New York: Springer. https://link.springer.com/content/pdf/10.1007/2F978-1-4614-9068-5_4.
- Boman, E. G., Catalyurek, U. V., Chevalier, C., and Devine, K. D. (2012). The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering, and coloring. *Scientific Programming*, 20(2):129–150.
- Bulut, M., Kandlikar, S. G., and Sozbir, N. (2019). A review of vapor chambers. *Heat Transfer Engineering*, 40(19):1151–1573. <https://doi.org/10.1080/01457632.2018.1480868>.
- Campbell, D. et al. (2012). Ubiquitous high performance computing: Challenge problems specification. *Georgia Tech. Res. Inst., Atlanta, GA, USA, Tech. Rep. HR0011-10-C-0145*.
- Carlson, T. E., Heirman, W., and Eeckhout, L. (2011). Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pages 1–12, doi: 10.1145/2063384.2063454.
- Chiang, T.-Y., Banerjee, K., and Saraswat, K. (2001). Compact modeling and spice-based simulation for electrothermal analysis of multilevel ulsi interconnects. In *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers*, 2001, pages 165–172, doi: 10.1109/ICCAD.2001.968613.
- Chowdhury, I. et al. (2009). On-chip cooling by superlattice-based thin-film thermoelectrics. *Nature nanotechnology*, 4(4):235.
- Chrobak, M., Dürr, C., Hurand, M., and Robert, J. (2008). Algorithms for temperature-aware task scheduling in microprocessor systems. In Fleischer, R., Xu, J. (eds) *Algorithmic Aspects in Information and Management. AAIM 2008. Lecture Notes in Computer Science*, volume 5034. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-68880-8_13.
- Coskun, A. K., Ayala, J. L., Atienza, D., and Rosing, T. S. (2009). Modeling and dynamic management of 3D multicore systems with liquid cooling. In *17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 35–40. IEEE.

- Coskun, A. K. et al. (2010). Energy-efficient variable-flow liquid cooling in 3D stacked architectures. In *IEEE Proc. of Design, Automation and Test in Europe (DATE)*, pages 111–116.
- Coskun, A. K., Rosing, T. S., and Whisnant, K. (2007). Temperature aware task scheduling in mpsoCs. In *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*, pages 1–6. IEEE.
- Dang, B. et al. (2010). Integrated microfluidic cooling and interconnects for 2D and 3D chips. *IEEE Transactions on Advanced Packaging*, 33(1):79–87.
- Distefano, G. P. (1968). Causes of instabilities in numerical integration techniques. *International Journal of Computer Mathematics*, 2(1-4):123–142.
- Edwards, H. C., Heroux, M. A., Williams, A. B., and Crozier, P. S. (2008). Hpc application performance analysis and prediction. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).
- Eris, F. et al. (2018). Leveraging thermally-aware chiplet organization in 2.5D systems to reclaim dark silicon. In *IEEE Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1441–1446.
- Gropp, W., Thakur, R., and Lusk, E. (1999). *Using MPI-2: advanced features of the message passing interface*. MIT press.
- Hanks, D. F. et al. (2018). Nanoporous membrane device for ultra high heat flux thermal management. *Microsystems & Nanoengineering*, 4(1):1.
- Hanumaiah, V. and Vrudhula, S. (2012). Temperature-aware DVFS for hard real-time applications on multicore processors. *IEEE Transactions on Computers*, 61(10):1484–1494.
- Heroux, M. A., Bartlett, R. A., Howle, V. E., Hoekstra, R. J., Hu, J. J., Kolda, T. G., Lehoucq, R. B., Long, K. R., Pawlowski, R. P., Phipps, E. T., et al. (2005). An overview of the trilinos project. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):397–423.
- Higham, N. J. (2002). *Accuracy and stability of numerical algorithms*, volume 80. Siam.
- Howard, J. et al. (2011). A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling. *IEEE Journal of Solid-State Circuits*, 46(1):173–183.
- Hsieh, J., Huang, H., and Shen, S. (2012). Experimental study of microrectangular groove structure covered with multi mesh layers on performance of flat plate heat pipe for led lighting module. *Microelectronics Reliability*, 52(6):1071–1079.

- Hu, Y., Chen, S., Peng, L., Song, E., and Choi, J.-W. (2013). Effective thermal control techniques for liquid-cooled 3d multi-core processors. In *International Symposium on Quality Electronic Design (ISQED)*, pages 8–15. IEEE.
- Huang, W., Stan, M. R., Skadron, K., Sankaranarayanan, K., Ghosh, S., and Velusam, S. (2004). Compact thermal modeling for temperature-aware design. In *Proc. of the 41st annual Design Automation Conference*, pages 878–883.
- Hutchinson, S., Keiter, E., Hoekstra, R., Watts, H., Waters, A., Russo, T., Schells, R., Wix, S., and Bogdan, C. (2002). The Xyce™ parallel electronic simulator—an overview. In *Parallel Computing: Advances and Current Issues*, pages 165–172. World Scientific. https://doi.org/10.1142/9781860949630_0021.
- Jagannadham, K. (1998). Multilayer diamond heat spreaders for electronic power devices. *Solid-State Electronics*, 42(12):2199–2208.
- Ju, Y. S. et al. (2013). Planar vapor chamber with hybrid evaporator wicks for the thermal management of high-heat-flux and high-power optoelectronic devices. *Intl. Journal of Heat and Mass Transfer*, 60:163–169.
- Kandlikar, S. G. (2002). Fundamental issues related to flow boiling in minichannels and microchannels. *Experimental Thermal and Fluid Science*, 26(2-4):389–407.
- Kaplan, F. and Coskun, A. K. (2015). Adaptive sprinting: How to get the most out of phase change based passive cooling. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 37–42. IEEE.
- Kaplan, F. et al. (2014). Modeling and analysis of phase change materials for efficient thermal management. In *IEEE Proc. of 32nd International Conference on Computer Design (ICCD)*, pages 256–263.
- Kaplan, F., Reda, S., and Coskun, A. K. (2017). Fast thermal modeling of liquid, thermo-electric, and hybrid cooling. In *Proc. of Intersociety Conf. on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pages 726–735.
- Kaplan, F., Said, M., Reda, S., and Coskun, A. K. (2019). Locool: Fighting hot spots locally for improving system energy efficiency. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(4):895–908. doi: 10.1109/TCAD.2019.2902355.
- Ladenheim et al. (2016). Ic thermal analyzer for versatile 3-d structures using multi-grid preconditioned krylov methods. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE.
- Lee, J. S., Skadron, K., and Chung, S. W. (2010). Predictive temperature-aware dvfs. *IEEE Transactions on Computers*, 59(1):127–133.

- Lee, W., Kim, Y., Ryoo, J. H., Sunwoo, D., Gerstlauer, A., and John, L. K. (2015). Powertrain: A learning-based calibration of mcpat power models. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 189–194. IEEE.
- Li, S. et al. (2009). McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *IEEE/ACM Proc. of 42nd Annual International Symposium on Microarchitecture (MICRO)*, pages 469–480.
- Li, X. S. (2005). An overview of superlu: Algorithms, implementation, and user interface. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):302–325.
- Liang, J., Masuya, S., Kasu, M., and Shigekawa, N. (2017). Realization of direct bonding of single crystal diamond and si substrates. *Applied Physics Letters*, 110(11):111603.
- Liang, J., Masuya, S., Kim, S., Oishi, T., Kasu, M., and Shigekawa, N. (2018). Stability of diamond/si bonding interface during device fabrication process. *Applied Physics Express*, 12(1):016501.
- Liang, J., Zhou, Y., Masuya, S., Guemann, F., Singh, M., Pomeroy, J., Kim, S., Kuball, M., Kasu, M., and Shigekawa, N. (2019). Annealing effect of surface-activated bonded diamond/si interface. *Diamond and Related Materials*, 93:187–192.
- Liu, P., Li, H., Jin, L., Wu, W., Tan, S. X.-D., and Yang, J. (2006). Fast thermal simulation for runtime temperature tracking and management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(12):2882–2893.
- Liu, W., Calimera, A., Nannarelli, A., Macii, E., and Poncino, M. (2009). On-chip thermal modeling based on spice simulation. In *International Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 66–75. Springer.
- Liu, X.-X., Zhai, K., Liu, Z., He, K., Tan, S. X.-D., and Yu, W. (2014). Parallel thermal analysis of 3-D integrated circuits with liquid cooling on cpu-gpu platforms. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(3):575–579.
- Long, J., Memik, S. O., Memik, G., and Mukherjee, R. (2008). Thermal monitoring mechanisms for chip multiprocessors. *ACM Transactions on Architecture and Code Optimization*, 5(2):article 9, 33 pages. <https://doi.org/10.1145/1400112.1400114>.
- Lu, Z. et al. (2016). Design and modeling of membrane-based evaporative cooling devices for thermal management of high heat fluxes. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 6(7):1056–1065.
- Lu, Z. et al. (2019). A unified relationship for evaporation kinetics at low mach numbers. *Nature Communications*, (2019):1–8.

- Madenci, E. and Guven, I. (2015). *The finite element method and applications in engineering using ANSYS®*. Springer.
- Massobrio, G. and Antognetti, P. (1993). *Semiconductor device modeling with SPICE*, volume 21. McGraw-Hill New York.
- Meng, J., Kawakami, K., and Coskun, A. K. (2012). Optimizing energy efficiency of 3-D multicore systems with stacked dram under power and thermal constraints. In *Proc. of Design Automation Conference (DAC)*, pages 648–655.
- Narayan, A., Thonnart, Y., Vivet, P., and Coskun, A. K. (2020). Prowaves: Proactive runtime wavelength selection for energy-efficient photonic NoCs. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*.
- Narayan, A., Thonnart, Y., Vivet, P., Tortolero, C. F., and Coskun, A. K. (2019). Waves: Wavelength selection for power-efficient 2.5D-integrated photonic NoCs. In *Proc. of IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 516–521.
- Nowroz, A. N., Cochran, R., and Reda, S. (2010). Thermal monitoring of real processors: Techniques for sensor allocation and full characterization. In *Design Automation Conference*, pages 56–61. IEEE.
- OpenMPI (2014). Open MPI: Open Source High Performance Computing. <https://www.open-mpi.org/>.
- Paterna, F. and Reda, S. (2013). Mitigating dark-silicon problems using superlattice-based thermoelectric coolers. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*, pages 1391–1394. IEEE.
- Pedram, M. and Nazarian, S. (2006). Thermal modeling, analysis, and management in vlsi circuits: Principles and methods. *Proceedings of the IEEE*, 94(8):1487–1501.
- Qian, H., Chang, C.-H., and Yu, H. (2013). An efficient channel clustering and flow rate allocation algorithm for non-uniform microfluidic cooling of 3d integrated circuits. *Integration*, 46(1):57–68.
- Reda, S., Cochran, R., and Nowroz, A. N. (2011). Improved thermal tracking for processors using hard and soft sensor allocation techniques. *IEEE Transactions on Computers*, 60(6):841–851.
- Sabry, M. M., Coskun, A. K., Atienza, D., Rosing, T. Š., and Brunschwiler, T. (2011). Energy-efficient multiobjective thermal control for liquid-cooled 3-d stacked architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(12):1883–1896.

- Sabry, M. M., Sridhar, A., Meng, J., Coskun, A. K., and Atienza, D. (2013). Greencool: An energy-efficient liquid cooling design technique for 3-d mpsocs via channel width modulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(4):524–537.
- Sadasivam, S. K., Thompto, B. W., Kalla, R., and Starke, W. J. (2017). Ibm power9 processor architecture. *IEEE Micro*, 37(2):40–51.
- Sadiqbatcha, S. I., Zhang, J., Amrouch, H., and Tan, S. X.-D. (2022). Real-time full-chip thermal tracking: A post-silicon, machine learning perspective. *IEEE Transactions on Computers*, 71(6):1411–1424, doi: 10.1109/TC.2021.3086112.
- Sahu, V., Joshi, Y. K., Fedorov, A. G., Bahk, J.-H., Wang, X., and Shakouri, A. (2014). Experimental characterization of hybrid solid-state and fluidic cooling for thermal management of localized hotspots. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 5(1):57–64, doi: 10.1109/TCPMT.2014.2332516.
- Saini, P. and Mehra, R. (2012). Leakage power reduction in cmos vlsi circuits. *International Journal of Computer Applications*, 55(8):42–48. doi: 10.5120/8778–2721.
- Schultz, M. et al. (2016). Embedded two-phase cooling of large three-dimensional compatible chips with radial channels. *Journal of Electronic Packaging*, 138(2):021005.
- Sharifi, S. and Rosing, T. S. (2010). Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(10):1586–1599.
- Sharma, C. S., Tiwari, M. K., Zimmermann, S., Brunschwiler, T., Schlottig, G., Michel, B., and Poulidakos, D. (2015). Energy efficient hotspot-targeted embedded liquid cooling of electronics. *Applied Energy*, 138:414–422.
- Sheikh, H. F., Ahmad, I., Wang, Z., and Ranka, S. (2012). An overview and classification of thermal-aware scheduling techniques for multi-core processing systems. *Sustainable Computing: Informatics and Systems*, 2(3):151–169.
- Shi, B. and Srivastava, A. (2013). Optimized micro-channel design for stacked 3-d-ics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(1):90–100.
- Shukla, P., Coskun, A. K., Pavlidis, V. F., and Salman, E. (2019). An Overview of Thermal Challenges and Opportunities for Monolithic 3D ICs. In *Proc. of the Great Lakes Symposium on VLSI (GLSVLSI)*.
- Sima, D. (2018). Intel core x-series (hed lines).
- Skadron, K. et al. (2003). Temperature-aware microarchitecture. In *IEEE Proc. of International Symposium on Computer Architecture (ISCA)*, pages 2–13.

- Sone, Y. (2000). Kinetic theoretical studies of the half-space problem of evaporation and condensation. *Transport Theory and Statistical Physics*, 29(3-5):227–260.
- Sridhar, A. et al. (2010). Compact transient thermal model for 3d ics with liquid cooling via enhanced heat transfer cavity geometries. In *2010 16th International Workshop on Thermal Investigations of ICs and Systems (THERMINC)*, pages 1–6. IEEE.
- Sridhar, A. et al. (2013a). 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. In *IEEE Proc. of the International Conference on Computer-Aided Design (ICCAD)*, pages 463–470.
- Sridhar, A. et al. (2013b). STEAM: A fast compact thermal model for two-phase cooling of integrated circuits. In *IEEE Proc. of International Conference on Computer-Aided Design (ICCAD)*, pages 256–263.
- Sridhar, A., Vincenzi, A., Atienza, D., and Brunschweiler, T. (2014). 3D-ICE: A compact thermal model for early-stage design of liquid-cooled ICs. *IEEE Transactions on Computers*, 63(10):2576–2589.
- Srinivasan, J., Adve, S. V., Bose, P., Rivers, J., and Hu, C.-K. (2003). Ramp: A model for reliability aware microprocessor design. *IBM research report*. <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.748&rep=rep1&type=pdf>.
- Tan, S. X.-D., Liu, P., Jiang, L., Wu, W., and Tirumala, M. (2008). A fast architecture-level thermal analysis method for runtime thermal regulation. *Journal of Low Power Electronics*, 4(2):139–148.
- Tang, Q., Gupta, S. K. S., and Varsamopoulos, G. (2008). Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. *IEEE Transactions on Parallel and Distributed Systems*, 19(11):1458–1472.
- Taylor, R. A. and Solbrekken, G. L. (2008). Comprehensive system-level optimization of thermoelectric devices for electronic cooling applications. *IEEE Transactions on components and packaging technologies*, 31(1):23–31.
- Thome, J. (2010). Heat transfer engineering data book III. http://www.publico-online.de/details/Heat_Transfer_Engineering_Data_Book_III_extract.pdf.
- Vaartstra, G., Lu, Z., and Wang, E. N. (2019). Simultaneous prediction of dryout heat flux and local temperature for thin film evaporation in micropillar wicks. *International Journal of Heat and Mass Transfer*, 136:170–177.
- Varshney, S., Sultan, H., Jain, P., and Sarangi, S. R. (2019). Nanotherm: An analytical fourier-boltzmann framework for full chip thermal simulations. In *Proceedings of International Conference On Computer Aided Design (ICCAD)*, pages 1–8.

- Vladimirescu, A. (1994). *The SPICE book*. Wiley New York.
- Wang, T. Y. and Chen, C. C. P. (2004). SPICE-compatible thermal simulation with lumped circuit modeling for thermal reliability analysis based on modeling order reduction. *International Symposium on Signals, Circuits and Systems. Proceedings, SCS 2003*, pages 357–362, doi: 10.1109/ISQED.2004.1283700.
- Wei, H., Wu, T. F., Sekar, D., Cronquist, B., Pease, R. F., and Mitra, S. (2012). Cooling three-dimensional integrated circuits using power delivery networks. In *2012 International Electron Devices Meeting*, pages 14–2. IEEE.
- Wei, M. et al. (2018). Optimization and thermal characterization of uniform silicon micropillar based evaporators. *International Journal of Heat and Mass Transfer*, 127:51–60.
- Wong, S., El-Gamal, A., Griffin, P., Nishi, Y., Pease, F., and Plummer, J. (2007). Monolithic 3D integrated circuits. In *International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, pages 1–4.
- Yazawa, K. et al. (2012). Cooling power optimization for hybrid solid-state and liquid cooling in integrated circuit chips with hotspots. In *IEEE Proc. of 13th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pages 99–106.
- Yu, W., Zhang, T., Yuan, X., and Qian, H. (2013). Fast 3-D thermal simulation for integrated circuits with domain decomposition method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(12):2014–2018.
- Yuan, Z. et al. (2019a). Modeling and optimization of chip cooling with two-phase vapor chambers. In *Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*.
- Yuan, Z. et al. (2019b). Two-phase vapor chambers with micropillar evaporators: a new approach to remove heat from future high-performance chips. In *Proc. of Intersociety Conf. on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*.
- Yuan, Z., Shukla, P., Chetoui, S., Nemtzow, S., Reda, S., and Coskun, A. K. (2022). Pact: An extensible parallel thermal simulator for emerging integration and cooling technologies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(4):1048–1061.
- Yuan, Z., Vaartstra, G., Shukla, P., Wang, E., Reda, S., and Coskun, A. K. (2020). A learning-based thermal simulation framework for emerging two-phase cooling technologies. In *Proc. of Design, Automation and Test in Europe (DATE)*.

- Yuan, Z., Zhang, T., Van Duren, J., and Coskun, A. K. (2021). Efficient thermal analysis of lab-grown diamond heat spreaders. In *International Electronic Packaging Technical Conference and Exhibition*, volume 85505, page V001T01A002. American Society of Mechanical Engineers.
- Zhang, J., Sadiqbatcha, S., O’Dea, M., Amrouch, H., and Tan, S. X.-D. (2022). 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*, 41(5):1453–1466. doi: 10.1109/TCAD.2021.3088081.
- Zhou, L., Tian, Y., Huang, H., Sato, H., and Shimizu, J. (2012). A study on the diamond grinding of ultra-thin silicon wafers. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 226(1):66–75.
- Zhou, T. Y., Zhou, D., Zhang, H., and Niu, X. (2008). Foundational-circuit-based spice simulation. In *Proc. of IEEE International Symposium on Circuits and Systems*, pages 876–879.
- Zhou, X., Yang, J., Xu, Y., Zhang, Y., and Zhao, J. (2010). Thermal-aware task scheduling for 3D multicore processors. *IEEE Transactions on Parallel and Distributed Systems*, 21(1):60–71.
- Ziabari, A., Park, J. H., Ardestani, E. K., Renau, J., Kang, S. M., and Shakouri, A. (2014). Power blurring: Fast static and transient thermal analysis method for packaged integrated circuits and power devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(11):2366–2379.

CURRICULUM VITAE

