# A HEURISTIC FOR OPTIMIZING THE PHYSICAL LAYOUT AND NETWORK TOPOLOGY OF INTEGRATED 3D MULTI-CHIP SYSTEMS UNDER TEMPERATURE CONSTRAINTS

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

DECEMBER 2018

By

Lambert Leong

Thesis Committee:

Henri Casanova, Chairperson
Lee Altenberg
Philip Johnson

Keywords: Integrated 3D Multi-Chip System, Simulation, Inductive Coupling, Hotspot

# ACKNOWLEDGMENTS

I would like to express my utmost thanks to Professor Henri Casanova for guiding and supporting me as an instructor, advisor, and my committee chair. Working with him has been a pleasure and intellectually rewarding. Thank you to Professor Michihiro Koibuchi and associates of the Koilab who we collaborated with on this project. I would also like to thank committee members, Professor Lee Altenberg and Professor Philip Johnson, for their input, advice, and support. Lastly, thank you to my father, Lambert, mother, Nori, and sister, Liane for their love and support.

# ABSTRACT

Many-core architectures provide large amounts of computational power and should thus be well-suited to running parallel applications. In this context, Integrated 3D multi-chip systems comprise multiple multi-core processor chips in a single tightly-coupled system. Heat dissipation is a major issue when designing integrated 3D multi-chip systems. It may not be possible to operate chips at their maximum frequency due to their being in close proximity to each other, which impedes heat dissipation. The physical layout of the chips is thus an important design consideration, also because it impacts the inter-chip network topology since a sparser physical layout can imply a less tightly-connected topology. Overall, there is a complex trade-off between between temperature, physical layout, chip operating frequencies, and inter-chip network topologies.

The ThruChip Interface (TCI) makes wireless communications between processor chips possible via inductive coupling, and thus can be used to construct inter-chip networks. Inductive coupling offers economical advantages (e.g., ability to produce/test individual chips before their are integrated in a multi-chip system), while also providing sufficient bandwidth between communicating chips with low power consumption. Furthermore, this technology makes it possible to construct integrated 3D multi-chip systems using a wide range of physical layout options: it is only necessary for two chips to overlap for them to be connected, the network bandwidth being proportional to the overlap area. In this thesis we assume the use of TCI and focus on building integrated 3D multi-chip systems using physical chip layouts that afford good heat dissipation, high chip operating frequencies, and good inter-chip network topologies. This amounts to solving a constrained multi-objective optimization problem. Previously proposed physical layouts for integrated 3D multi-chip systems include the "stack," which consist of multiple chips stacked vertically above one another, and the "checkerboard," which consists of chips connected via partial overlaps at each of the four corners so as to resemble a checkerboard when viewed from above. These baseline layouts provide poor (in the case of the stack) or likely improvable (in the case of the checkerboard) solutions to the aforementioned optimization problem.

We propose a randomized greedy heuristic to construct layouts that are superior to the baseline layouts both in terms of compute power and network topology. This heuristic relies on the Hotspot simulator to evaluate the temperature of candidate layouts. Hotspot simulations are computationally intensive, and we use various techniques that make our approach feasible in spite of the Hotspot bottleneck. We present results achieved by our

heuristic when generating 6-, 9-, and 13-chips layouts and compare these results to the checkerboard layout. Our results show that the heuristic-generated layouts are strictly superior to the baseline checkerboard layout, typically affording significantly improved compute power. Our key finding is that a randomized greedy heuristic is sufficient to generate layouts for integrated 3D multi-chip systems that afford increased performance to parallel applications when compared to previously proposed state-of-the-art layouts.

# TABLE OF CONTENTS

**Acknowledgments** . . . . . **iii**

**Abstract** . . . . . **iv**

**List of Tables** . . . . . **ix**

**List of Figures** . . . . . **x**

**1 Introduction** . . . . . **1**

1.1 Motivation . . . . . 1

1.2 Thesis Statement . . . . . 2

1.3 Thesis Road-map . . . . . 2

**2 Background and Related Work** . . . . . **4**

2.1 Heat Dissipation in Integrated 3D Multi-Chip Systems . . . . . 4

2.1.1 Liquid Cooling . . . . . 4

2.1.2 The Sparse Checkerboard Layout . . . . . 6

2.2 Wireless Inter-Chip Network Connections . . . . . 7

2.2.1 Inductive Coupling . . . . . 8

2.2.2 crosstalk Avoidance . . . . . 9

**3 Problem Statement and Approach** . . . . . **12**

3.1 Problem Statement . . . . . 12

3.1.1 Evaluation Metrics . . . . . 12

3.1.2 Simulation Infrastructure . . . . . 13

3.2 General Heuristic Approach . . . . . 13

**4 Mitigating the Hotspot Bottleneck . . . . . 15**

4.1 Hotspot API . . . . . 15

4.2 Thermodynamic Simulation Acceleration . . . . . 16

4.3 Grid Size Adjustments . . . . . 17

4.4 Conclusion . . . . . 18

**5 Layout Construction Heuristic . . . . . 19**

5.1 Heuristic . . . . . 19

5.1.1 Starting Point "Cradle" layout . . . . . 20

5.1.2 Candidate Generation . . . . . 20

5.1.3 Candidate Selection . . . . . 21

5.1.4 Challenges . . . . . 22

5.2 Parallel Hotspot Invocations . . . . . 22

5.2.1 Multithreading . . . . . 22

5.2.2 Message Passing Interface . . . . . 23

5.3 Lowering Number of Hotspot Invocations . . . . . 26

5.3.1 Multiple Chip Additions . . . . . 26

5.3.2 Static Sub-Layout Additions . . . . . 26

**6 Results . . . . . 28**

6.1 Layout Evaluation Framework . . . . . 28

6.1.1 Simulation of TCI, Wireless Communication . . . . . 29

6.1.2 Heuristic Implementation . . . . . 29
6.2 Simulation Assumptions . . . . . 30
6.3 Baseline Layouts . . . . . 31
6.4 Heuristic Tuning Experiments . . . . . 33
6.4.1 Candidate Selection Criteria . . . . . 34
6.4.2 Adaptive Selection Criterion . . . . . 36
6.4.3 Multi-Chip Additions . . . . . 38
6.5 Main Results . . . . . 43
6.5.1 6-Chip Layouts . . . . . 43
6.5.2 9-Chip Layouts . . . . . 44
6.5.3 13-Chip Layouts . . . . . 45
6.5.4 Summary of Results . . . . . 45
**7 Conclusion** . . . . . **46**
7.1 Summary of Contributions and Findings . . . . . 46
7.2 Future Work . . . . . 47
**A Heuristic Generated Layouts** . . . . . **49**
**Bibliography** . . . . . **56**

# LIST OF TABLES

4.1 SuperLU's effect on Hotspot's runtime for a 5-chip layout temperature evaluation. 16

6.1 Specification of our baseline chip. . . . . 30

6.2 Stack layout results. All layouts are above the temperature threshold even if chips are operated at their lowest possible frequencies. . . . . 32

6.3 Checkerboard Metrics . . . . . 33

6.4 Results from different candidate layout selection criterion for 6, 9, and 13 chip layouts with a .1 and .2 overlap. Note that non-integer values are the result of averaging over 10 trials. Best network and power metrics are shown in boldface. 35

6.5 "Power then temperature" results for different $\alpha$ values for 6- and 9-chip layouts with .2 overlap. Values are shown as averages over 10 trials. Best network and power metrics are shown in boldface. . . . . 37

6.6 "Network then temperature" results for different $\alpha$ values for 6- and 9-chip layouts with .2 overlap. Values are shown as averages over 10 trials. Best network and power metrics are shown in boldface. . . . . 38

6.7 Results for adding chips in multiples of 3, 2, 1, and by cradles. Best network and power metrics are shown in boldface. . . . . 40

6.8 Multi-Chip addition methods under time constraints (300 seconds for 6 chip layouts and 1200 seconds for 9 chip layouts). Best network and power metrics are shown in boldface. . . . . 42

6.9 Comparison of best heuristic-generated layouts vs checkerboard. Best network and power metrics are shown in boldface. . . . . 44

# LIST OF FIGURES


2.1 Maximum chip operating frequency vs. number of chips in a stack Integrated Multi-Chip System for different cooling options. (Courtesy of Prof. Michihiro Koibuchi, National Informatics Institute, Japan.) . . . . . 5

2.2 Baseline checkerboard layout. . . . . 6

2.3 Temperature vs. chip Frequency for 5-chip systems. (Courtesy of Prof. Michihiro Koibuchi, National Informatics Institute, Japan.) . . . . . 7

2.4 Connecting partially overlapping chips using inductors. . . . . 8

2.5 Example of mitigating crosstalk. . . . . 9

2.6 Crosstalk Compliant Traditional Layouts: Black Rectangles are Chips, Red Rectangles are Induction Zones. . . . . 10

4.1 Speedup and percent error vs. `GRID_SIZE`. . . . . 18

5.1 Cradle Structure with strip overlap areas: black boxes are chips, red boxes are induction zones. . . . . 20

5.2 Cradle Structure with square overlap areas: black boxes are chips, red boxes are induction zones. . . . . 21

5.3 Speedup and parallel efficiency vs. the number of MPI worker threads invoking Hotspot. . . . . 24

5.4 Increasing number of Hotspot instances affect on runtime and L1 and LLC cache misses. . . . . 25

A.1 6 Chip, 0.2 Overlap, Top View. . . . . 50

A.2 6 Chip, 0.2 Overlap, Side View. . . . . 50

A.3 6 Chip, 0.1 Overlap, Top View. . . . . 51

A.4 6 Chip, 0.1 Overlap, Side View. . . . . 51

A.5 9 Chip, 0.2 Overlap, Top View. . . . . 52

A.6 9 Chip, 0.2 Overlap, Side View. . . . . 52

A.7 9 Chip, 0.1 Overlap, Top View. . . . . 53

A.8 9 Chip, 0.1 Overlap, Side View. . . . . 53

A.9 13 Chip, 0.2 Overlap, Top View. . . . . 54

A.10 13 Chip, 0.2 Overlap, Side View. . . . . 54

A.11 13 Chip, 0.1 Overlap, Top View. . . . . 55

A.12 13 Chip, 0.1 Overlap, Side View. . . . . 55

# CHAPTER 1
# INTRODUCTION

## 1.1 Motivation

Integrated 3D multi-chip systems consist of multiple microprocessor chips interconnected via some network topology. The goal is to use large numbers of chips operated at high frequencies with an efficient network topology so as to afford large amounts of computational power in a single system for executing parallel applications.

A key concern for integrated 3D multi-chip systems is temperature. In traditional designs chips are stacked on top of each other, but stacking many high-frequency chips in close proximity generates a considerable amount of heat which can easily overcome chips' operating temperatures. One solution is to reduce chip frequencies by adjusting the dynamic voltage and frequency scaling (DVFS) levels for each chip, so as to reduce power consumption and thus heat. However, reducing chip frequencies is at the expense of application performance. Another solution to the heat problem is to decrease the heat density by making sure that the physical layout of chips limits the proximity and amount of overlap between chips, thus promoting better heat dissipation. This second solution can result in an inter-chip network topology with poorer performance, e.g., due to increased hop counts between chips, which in turns also diminished parallel application performance. There is thus a complex trade-off between temperature, chip frequencies, inter-chip network topology, and physical layout.

Several interconnection technologies have been developed for constructing 3-D multi-chip systems, including wire-bonding, micro-bump, and through-silicon via (TSV). These technologies have made it possible to construct systems as rigid chip stacks, which suffer from poor heat dissipation. An emerging technology is inductive-coupling using the ThruChip Interface (TCI), which operates by generating a magnetic field between a pair of transmitter and receiver coils (i.e., inductors) for data transfers. TCI makes it possible to interconnect chips wireless provided they are in close proximity to one another. Importantly, chips can be overlapped in arbitrary fashion, and the communication bandwidth is proportional to the amount of overlap that exists between two connected chips. There is thus a clear trade-off between network bandwidth (higher chip overlap leads to higher bandwidth) and temperature (higher chip overlap leads to poorer heat dissipation). One of the compelling advantages of inductive coupling technology is that it makes it possible to connect individually fabricated/tested

chips using a wide range of available physical layouts. This is in contrast with traditional technologies with which systems are built monolithically using a rigid stack physical layout. In summary, TCI makes it feasible to consider promising and as of yet unexplored physical layouts for designing integrated 3D multi-chip systems. In other terms, it makes it possible to explore the complex trade-off outlines at the end of the previous paragraph.

## 1.2 Thesis Statement

*In this thesis we explore the question of how to arrange and connect chips in physical layouts so as to achieve desirable trade-offs between temperature, chip frequencies, and inter-chip network topology when designing integrated 3D multi-chip systems using TCI.*

More specifically, given a number of identical chips with known power profiles and physical dimensions, we wish to arrange them in a physical layout so as to maximize chip frequencies and minimize hop counts given temperature and network bandwidth constraints. This is a complex constrained objective optimization problem. A formalization of the problem would have to include integer variables and objective functions (e.g., to capture hop counts), making the optimization problem NP-hard. Furthermore, the temperature constraint derives from laws of physics and cannot be analytically evaluate. Instead, it must be evaluated empirically based on thermodynamics simulations. Finally, the problem is multi-objective since both chip frequencies and hop counts need to be optimized. Given the above, our goal in this work is to design a heuristic that produces non-guaranteed but good-quality solutions, i.e., chip layouts, in tractable amounts of time. Note that we develop this heuristic with the goal of designing general-purpose systems, i.e., optimizing generic metrics of performance (chip frequencies, network hop count statistics), rather than designing application-specific systems the network topology of which is optimized to certain communication patterns.

## 1.3 Thesis Road-map

This thesis is organized as follows. In Chapter 2 we discuss the background material related to integrated 3D multi-chip systems (and in particular proposed solutions to overcome heat dissipation issues for chip stacks), and to TCI technology. In Chapter 3 we formalize our optimization problem and detail our overall approach for designing a heuristic to solve the problem. This approach entails running thermodynamics simulations to estimate the temperature of candidate solutions to the problem. One challenge in this work is that

these simulations are time-consuming, which severely limits the breadth and depth of the search that can be performed if solutions are to be obtained in tractable amounts of time. Consequently, in Chapter 4 we investigate approaches for increasing the performance of these thermodynamics simulations. In Chapter 5 we detail a heuristic algorithm, which constructs solutions incrementally and select candidate layouts according to various metrics of goodness, and which also employs various techniques to reduce the number of thermodynamics simulations to perform. Chapter 6 presents our experimental methodology and results, which allow us to draw quantitative comparisons between layouts produced by our heuristics and previously proposed layouts. Finally, in Chapter 7 we provide a brief summary of our findings, discuss their significance, and discuss future directions.

# CHAPTER 2
# BACKGROUND AND RELATED WORK

The work in this thesis is motivated by previous works on systems that explored different multi-chip layouts and architectures. These works, there is mention of two standard layouts: the *stack* and the *checkerboard*. Although these layouts have proved useful in certain use cases and/or for specific applications, each comes with its own set of shortcomings. In this chapter we discuss relevant previous work on multi-chip systems, cooling methods, and inter-chip communication technology. We examine the heat issue and possible solutions that have been proposed to increase the scale of multi-chip systems in spite of this issue. We also present background material on our choice technology for inter-chip communication, TCI.

## 2.1 Heat Dissipation in Integrated 3D Multi-Chip Systems

Heat dissipation is a well known problems for integrated 3D multi-chip systems and it directly affects the performance and longevity of a system. High temperature within a system is due to chips physical positions and to chip clock frequencies: the closer the chips are to one another and the higher the DVFS level, the higher the temperature of the chips and thus of the overall system. A few strategies can be employed to mitigate the heat builds up, including use of various coolants, reducing the chip proximity, and lowering the DVFS levels. Each strategy comes with its own drawbacks in terms of cost, feasibility, and/or system performance, as discussed hereafter.

### 2.1.1 Liquid Cooling

Air has historically been used as a computer coolant. However, liquid coolants are increasingly being used in High Performance Computing (HPC) and data center platforms (e.g., Fluorinert for the Cray-2 series, Mineral oil for the Tsubame-KFC supercomputer, Fluorinert for the kukai cluster at Yahoo Japan). In comparison to liquid coolants, air has a low thermal conductivity and often require high speed fans to actively dissipate heat away from the system. Liquid coolants also benefit from natural convection effects that cool chips more efficiently than air even with high speed fans. Liquid-cooled HPC systems have reported power usage effectiveness (PUE) as low as 1.03 which further argues in favor of non-air, liquid coolants. Mineral oil and fluorinert are commonly used liquid coolants that are chemically inert which means they do not react with the computer components they come into contact

with. Mineral oil has the lower thermal conductivity of the two and is also more viscous. Fluorinert has better conductivity, but is expensive and poses potential health hazards for those working with it or on systems containing it. Water is another liquid coolant that is commonly used as is has better thermal conductivity and is cheaper than mineral oil and fluorinert. Unfortunately, since it is does not provide electrical insulation, pipes and tubing must be used, so that water is not in direct contact with the chips. By contrast, electrical insulation afforded by mineral oil and fluorinert liquid coolants allows for the full submergence of systems in the coolant. This is a big advantage over traditional water cooled systems due to a larger surface area in contact with coolants. Recently, fully submersible water cooled system prototypes were developed by coating commodity computers with a parylene film; specifically diX C Plus film provided by KISCO Ltd. [1].

Early studies of in-water computing have shown waters superior cooling abilities over air and other liquid coolants. Considering a baseline stack layout ranging from 1 to 7 chips and a max temperature constraint of 58 °C. Maximum allowed chip operating frequencies were determined using the Hotspot v6.0 [2] heat simulator. Results of different coolants on the operating frequencies of chips in a stack are presented in Figure 2.1.

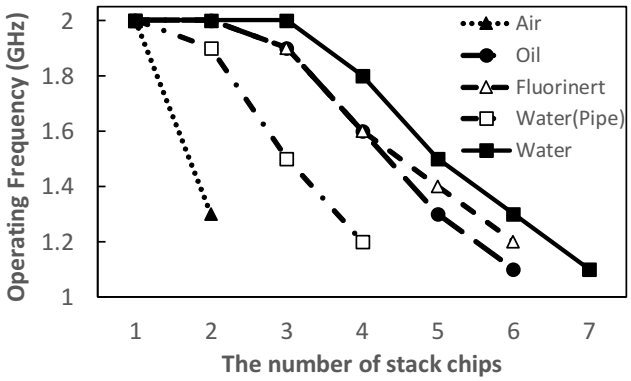



Figure 2.1: Maximum chip operating frequency vs. number of chips in a stack Integrated Multi-Chip System for different cooling options. (Courtesy of Prof. Michihiro Koibuchi, National Informatics Institute, Japan.)

Figure 2.1 highlights the potential of water cooling, which in these experiments allows a 7-chip stack to operate. While these results make a strong case for an exploration into in-water computing, it is not without challenges. Parylene coating the motherboard reduces its modularity and faulty components cannot be replaced once coated and it is unknown how long the the coating last under these use conditions. Although water makes a 7-chip stack possible, chips are clocked at a low frequency. It is thus unlikely that this layout could

scale to more chips while remaining useful. The stack is limited by its inherent inability to dissipate heat, even with the help of water. This motivates the need to explore other layouts, which are better at dissipating heat, regardless of the coolant in use.

### 2.1.2 The Sparse Checkerboard Layout

Since the stack layout still produces a lot of heat even when it is immersed in water, another layout has been proposed that provides lower chip density and thus increased cooling surface: the checkerboard layout [3]. In a checkerboard layout chips are connected via partial overlaps at each of the four corners, so as to resemble a checkerboard when viewed from above, as seen in Figure 2.2.

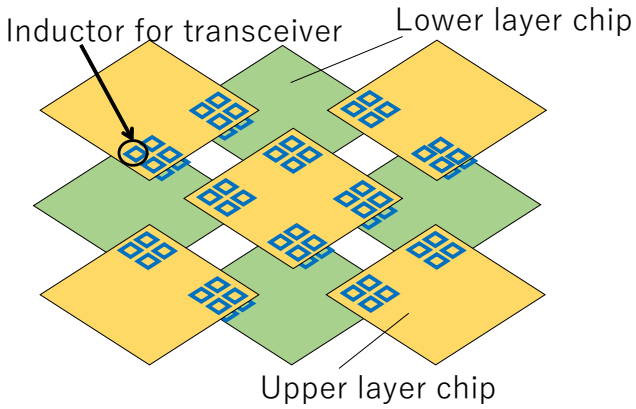



Figure 2.2: Baseline checkerboard layout.

Figure 2.3 shows temperature vs. chip operating frequencies for a stack layout with water immersion, a checkerboard layout with air cooling, and a checkerboard layout with water immersion. These results indicate that layout geometry seems to have a greater affect on system temperature than the type of coolant. There is an advantage to using water instead of air in the case of a checkerboard layout. However, a significant temperature difference exists between the stack and checkerboard layouts. Thusly, it can be concluded that layout geometry plays a large role in the temperature of the system.

In this thesis we only consider traditional air cooling. While full water immersion seems to be the best option in terms of minimizing heat generation, it is still very experimental and system longevity issues are not yet resolved (i.e., no coatings are needed and there is no potential for coating failures and system damage). Air-cooling is a low-cost approach that can be used in the consumer market, without requiring any special maintenance or

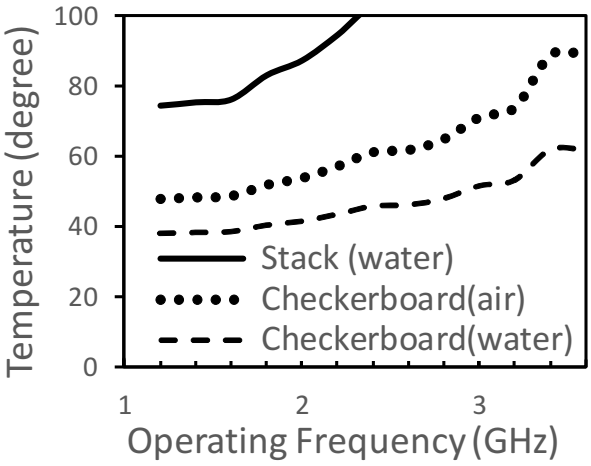



Figure 2.3: Temperature vs. chip Frequency for 5-chip systems. (Courtesy of Prof. Michihiro Koibuchi, National Informatics Institute, Japan.)

equipment. By contrast, liquid cooling approaches are for now confined to high-end HPC system and come with cost, maintenance, and even health issues. This said, we expect that layouts generated in this work assuming air cooling would also be effective in liquid cooling scenarios. But the results in Figure 2.3 show that the layout geometry, the focus of this work, plays a large role in reducing heat build up regardless of the cooling approach.

## 2.2 Wireless Inter-Chip Network Connections

Several interconnection technologies have been developed that allow for the construction of integrated 3D multi-chip systems which include, wire-bonding, micro-bump, and through silicon via (TSV). While TSV has become mainstream, wireless interconnects based on inductive coupling is an alternative attractive approach. Inductive-coupling ThruChip Interface (TCI) offers high speed (e.g., 8 Tbps [4]), low power (e.g., 0.14 pJ/b [5]) due in part to the removal of electrostatic discharge (ESD) protection devices, high integration (e.g., a 128-die NAND stack [6]), and applicability to an increasing range of computing systems including a reconfigurable processor [7]. A High-end chip multiprocessor (Integrated 3D Multi-Chip System) stacked with 4 DRAM chips for an aggregate 2 TiB/s RAM bandwidth using inductive coupling is planned for use in the custom supercomputer ranked in 4th position of Top500 as of Nov. 2017 [8–10]. This system will be the first commercially available Multi-Chip System using inductive coupling.

### 2.2.1 Inductive Coupling

Inductive coupling using TCI operates by generating a magnetic field between a pair of transmitter and receivers coils (i.e., inductors) for data transfer. One advantage of TCI technology is that it can tolerate high levels of coil misalignment. In commercial products, misalignment by 25% of the coil diameter can be tolerated, and even up to 40% misalignment is tolerable in usual operating conditions [11, 12]. Furthermore, increasing power improves coil misalignment tolerance. It has been reported that increasing energy consumption by only 4.1%, when compared to a perfectly aligned 3-D Network-On-Chip (NoC), leads to 50% misalignment tolerance [13]. High misalignment tolerance makes it possible to design 3-D layouts beyond a single 3-D rigid stack, namely layouts in which chips overlap only partially. As a result the layout design space is dramatically increased. In [14] a $675\mu\text{m} \times 675\mu\text{m}$ area is reserved for a single vertical link for 64 Gbps. From these findings we can extrapolate that high inter-chip bandwidths can be achieve with partial overlaps as long as the overlap area is large enough. Communicating inductors on overlaping chips are shown in Figure 2.4.

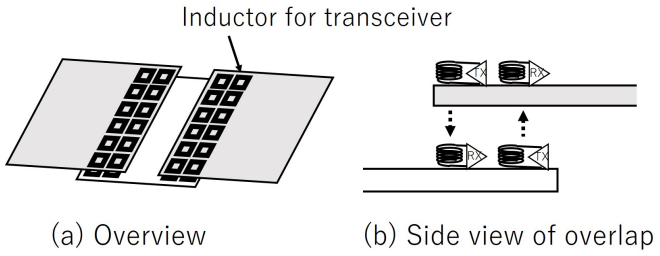



Figure 2.4: Connecting partially overlapping chips using inductors.

TCI offers an economical advantage when compared to TSV. TCI-based layouts consists of chips that are prefabricated and tested individually before assembling them into a Multi-Chip System. This is in contrast to a TSV-based design in which a chip stack is produced as a single unit and once the layout is fabricated it can not be changed without re-engineering all the TSV connections in the layout. Unlike TSV systems, TCI-based systems allow chips to be added or swapped out with more ease during system integration. TCI-based layouts thus come with a significant scalability and modularity advantage.

TCI presents it own set of challenges for constructing 3D-integrated systems. Inductive coupling requires a minimum distance for communication, which requires chips to be close to each other (but only overlapping partially) to achieve high bandwidth. As a result of these, heat dissipation is reduced. The heat problem is further exacerbated by the use of computer glue that must be inserted between chips to hold the 3D geometry of the layout. The glue

prevents the contact of the cooling medium, in our case air, with the chips. This again reduces heat dissipation. Another challenge, which is due to requiring close proximity of inter-connected chips, is that of communication interference. This phenomenon is discussed in the following section.

### 2.2.2 crosstalk Avoidance

Inductive coupling can cause communication interference, a phenomenon known as crosstalk, which must be avoided altogether when constructing a 3D-Integrated system using TCI. Crosstalk occurs between vertical links that are physically located directly above or below each other. Interference as a result of crosstalk negatively impacts inter-chip communication, which in turns can dramatically reduce application performance. Crosstalk can be avoided altogether by doubling the footprint of the induction area [15, 16]. In this thesis we avoid crosstalk by constructing 3-D layouts with the constraint that no two vertical links can be placed directly above each other. An example of crosstalk and a means to avoid it is shown in Figure 2.5

(a) Example of crosstalk on a 3-chip layout.

(b) Example of one possible reconfiguration of a 3-chip layout that avoids crosstalk.

Figure 2.5: Example of mitigating crosstalk.

The layout in Figure 2.5a consist of 3 chips on 3 different levels with 2 communication links, or induction zones. The induction zones, shown in red, are in direct vertical proximity to each other. This configuration leads to crosstalk: communication between the two chips on levels 1 and 2 may with communication between the two chips on level 2 and 3. Figure 2.5a is the classical example of crosstalk, which we completely avoid in this work. Figure 2.5b

shows an example configuration that avoids crosstalk. This new configuration maintains the same network properties as that in Figure 2.5a (i.e., diameter, number of edges, hop counts). The main difference is that the layout in Figure 2.5b experiences no crosstalk because the induction zones are no longer placed in vertical direct proximity to each other.

Avoiding crosstalk requires a re-evaluation of the baseline stack and checkerboard layouts. It is unlikely that a stack layout would be constructed using TCI in the consumer market, since TSV has been used successfully for this purpose. Nevertheless, it is possible to construct a chip stack that uses TCI for communication, and we use such a chip stack for comparison purposes and completeness in our experiments. A TCI-constructed stack layout simply requires inductors coils to be staggered for each chip level. This means that the full chip overlap cannot be utilized for communication (as seen in upcoming chapters, high bandwidth can be achieved with overlap areas that are only a small fraction of the chip surface area). Figure 2.6a presents a side view of a 5-chip stack layout where processor chips are shown in black and communication of induction zones are shown in red. The red zones are staggered on each level and only vertically above another when pairs of communicating chips chips are not in neighboring levels.

(a) Stacked Layout: Side View.

(b) Checkerboard Layout.

Figure 2.6: Crosstalk Compliant Traditional Layouts: Black Rectangles are Chips, Red Rectangles are Induction Zones.

In the checkerboard layout, if crosstalk is to be avoid, each chip is only allowed to have 4 neighboring chips (at the 4 corners of a chip connected either above or below). Without the crosstalk constraint, a chip could have up to eight neighbors in a 3-level checkerboard layout (each at the four corners of a chip connected above and each at the four corners of a chip connected below). Figure 2.6b shows an example of a 5-chip checkerboard that

avoids crosstalk. The chip on the first level has four communication links (one to each of the neighboring chips on level 2) and thus is not able to have any additional connections. Additional chips must only overlap and connect to the chips in level 2. In this thesis, when we refer to the stack and checkerboard layout, we mean the versions of these layouts that avoid crosstalk.

# CHAPTER 3
# PROBLEM STATEMENT AND APPROACH

In this chapter we detail the problem that we address in this thesis and outline a general approach for tractably computing a solution to it. This problem is a difficult multi-objective constrained optimization, and we review review the metrics that define our objective functions and the simulation infrastructure that is needed to evaluate some of these metrics. We then justify our using a simple greedy heuristic approach for solving this problem.

## 3.1 Problem Statement

Given a number of identical chips, with known physical dimensions and power/frequency profiles, given a maximum temperature threshold that cannot be overcome, and given a minimum overlap area between chips, we wish to arrange the chips in a 3D physical layout so as to maximize chip frequencies and optimize the inter-chip network topology (low diameter, low average shortest path length, high number of edges).

### 3.1.1 Evaluation Metrics

We evaluate the quality of a 3D layout based on the resulting inter-chip network and the homogeneous power output of the chips. A good network is desired because it will reduce hop counts and higher hop counts lead to less communication latency. Communication affects the overall runtime of an application and more hops will increase the percentage of time spent communicating rather than computing. Diameter, the number of edges, and average shortest path length (ASPL) are the metrics we use to characterize the goodness of a network. A good network topology would consist of a small diameter, high number of edges, and small ASPL. Chip operating frequencies and power levels are directly related, and higher operating powers and frequencies allow for faster computational time when running applications. When designing 3D layouts, we enforce a homogeneous power distribution in which all chips are set to operate at the same frequency. There are situations in which some parallel applications utilize each chip differently, in which case it may be beneficial to allow chips to operate a different frequencies. However, recall that our purpose here is to construct application-agnostic systems. Consequently, we do not consider heterogeneous chip frequencies and instead operate all chips at the maximum possible same frequency.

Temperature is the primary constraint and it relates to both the inter-chip network and the chip operating powers. Better networks, with high communication bandwidths, lead to denser layouts, less heat dissipation, and higher temperature. Additionally, higher operating frequencies require more power and more power generates more heat. Thusly, there is a trade-off between layout temperatures and inter-chip network topology as well as chip operating frequencies and layout temperatures. Ultimately, layouts must remain under a specified temperature threshold in order to be considered for comparison on the basis of its network and power metrics.

### 3.1.2 Simulation Infrastructure

Layout construction and evaluation are done in simulation. Simulation offers a means to explore and evaluate different layout designs without having to build actual physical systems. The majority of our layout construction framework is written in Python 2.7 [17]. We use the Hotspot v6.0 [2] simulation tool to compute maximum temperature (based on chip power traces generated by McPAT v1.3 [18] simulations). One of the drawbacks of Hotspot is its long execution time, which we discuss in upcoming chapters. :w

## 3.2 General Heuristic Approach

There are many strategies and techniques for handling multi-objective optimization problems. Time is often a limiting factor and the goal is to design heuristics that compute good solutions in reasonable amounts of time.

The use of meta-heuristics and other optimization techniques were considered to solve our multi-objective optimization problem. A classical approach would be to use simulated annealing, gradient descent, genetic algorithms, or other meta-heuristics. Given a putative 3D layout, evaluating its temperature (i.e., the maximum temperature at any point of any chip in the layout) entails performing thermodynamic simulations. As explained in the previous section we do this using the standard community tool Hotspot. Unfortunately, Hotspot is compute-intensive (due to solving large linear systems of equations). This precludes the use of meta-heuristics since they typically would need large number of Hotspot invocations (e.g., evaluating many individual layouts at each generation of a genetic algorithm to compute fitness functions).

Due to the above challenge, which we discuss more in depth in the next chapter, we instead seek to develop a simple greedy heuristic to build 3D layouts. The hope is that

our heuristic produces good 3D layouts that outperform, e.g., checkerboard layouts, and computes solution in tractable amounts of time.

# CHAPTER 4
# MITIGATING THE HOTSPOT BOTTLENECK

The Hotspot simulator is use to perform thermodynamic simulations of our constructed layouts. For every point in the layout, based on a spatial discretization, Hotspot calculates the temperature at that point. The temperature of the hottest point is then returned as as a value in °C. In order to compute these temperatures, Hotspot solves large linear systems of equation. As a result, Hotspot executions take between 100 and 150 times longer than the rest of the layout construction heuristic implementation. This problem is further exacerbated when we attempt to attempt to build bigger layouts that contain more chips, or when we attempt to perform more thorough searches of the layout design space. In this chapter we investigate this Hotspot computational bottleneck and attempt to reduce it in various ways, including exposing parallelism so as to run Hotspot efficiently on multi-core architectures.

## 4.1 Hotspot API

As previously mentioned, the runtime of Hotspot is too substantial to ignore and accounts for a significant portion of the layout construction time. An existing API, written in Python by others, was used to prepare input files and call Hotspot to perform the heat evaluation at each step of our layout construction procedure. Initial profiling results, with Python's cProfile profiler, of the Hotspot API were misleading. Profiling indicated that the majority of the execution time was spent forking processes. Unfortunately, the API was implemented monolythically as wrappers around Shell scripts (without any modularity or OOP). It was difficult to determine precisely whether the time spent in the forked processes was the same across all processes or was different. It was even difficult to determine computational bottlenecks, but we theorized that the time spent that the call to the functions implemented in the `hotspot.c` source file accounted for the majority of the execution time.

The Hotspot Python API code required refactoring to include classes and functions. In the original code, processes were forked in order to invoke Shell scripts to perform file reads and writes, sorting, string manipulations, and the calling of C programs. However, Python modules and libraries can handle the I/O, sorting, and string manipulations without using bash or the needing to fork extra OS processes. In particular, we used the `subprocess` Python module to handle the calling of the two C programs `cell.c` and `hotspot.c` and as a result, a slight speed up of <0.05x was observed. More importantly, the API code could

then be profiled accurately. Profiling results indicate that more that 98% of the execution time is spent in invoking the two C programs above, with the call to `hotspot.c` being much more time-consuming than the call to `cell.c`. From these profiling results we concluded that efforts needed to be primarily focused on `hotspot.c`.

## 4.2 Thermodynamic Simulation Acceleration

Profiling of `hotspot.c` using Gprof indicates that 74% of the execution time is spent loading data into large matrices and performing a sparse LU factorization. This factorization is performed using the efficient SuperLU numerical library. The use of SuperLU can be disabled, in which case, for a simple 5-chip layout experiment, we experience a slow down of up 2455 (see Table 4.1). While SuperLU results in tremendous speedup, the overall runtime of Hotspot is till a major computational bottleneck of our layout construction heuristic (and makes the use of meta-heuristics out of reach). A multi-threaded version of the SuperLU library exists, known as SuperLU_MT. However, it presents its own sent of challenges for integration with Hotspot. This is because SuperLUMT utilizes different data structures than that of the sequential and currently used in Hotspot SuperLU. Fitting the current data into a new data structure would require extensive reformatting of data and refactoring of Hotspot code, which is beyond the scope of this thesis.

Table 4.1: SuperLU's effect on Hotspot's runtime for a 5-chip layout temperature evaluation.

| Hotspot | Runtime(seconds) | Speedup |
|---|---|---|
| Without SuperLU | 56471 | |
| Compiled with SuperLU | 23 | 2455 |

Attempts were made to parallelize other parts of the Hotspot code that account for smaller portions of the runtime. OpenMP was used in an attempt to exploit some level of parallelization. For example, the `hotspot.c` source file contains functions that set the values in a large matrices to the same floating point number. Accessing matrix elements and setting them to a value can be done safely in parallel without race conditions. While we were able to expose some parallelism with OpenMP on some loops, the overall effect on the Hotspot runtime was mostly insignificant. Other parallelization efforts were directed at `cell.c`, which contains code responsible for producing relevant input files utilized by many classes in the Hotspot API to, in turn, produce other input files. The majority of the execution time in `cell.c` is spent resetting values to zero in many matrices and OpenMP is to help parallelize this process. But since the code in `cell.c` is not the bottleneck, the

overall speedup due to these parallelization is marginal and does not significantly decrease the overall Hotspot execution time.

## 4.3 Grid Size Adjustments

A variable called `GRID_SIZE`is defined in the `cell.c` source file to define the maximum dimensions of the Hotspot evaluation area. `GRID_SIZE`also directly correspondences to the amount of RAM code in `hotspot.c` allocates and can use to perform temperature computations. Higher `GRID_SIZE`values lead to a higher resolution and, as a result, more accurate Hotspot output temperatures. The `GRID_SIZE`is set by default to 2,048. But per request of our collaborators, due to temperature accuracy concerns, the `GRID_SIZE`was subsequently set to 8,192. As expected, experiments with Hotspot show that increasing the `GRID_SIZE`leads to more accurate temperature computations, but also leads to an increase in the Hotspot runtime.

Experiments were designed and run to evaluate the effect of `GRID_SIZE`on the overall runtime of Hotspot and to investigate the relationship between the `GRID_SIZE`and the accuracy of the temperature output. We start with a `GRID_SIZE`of 8192, invoke Hotspot, and capture the runtime and output temperature. 10 trials are performed for this `GRID_SIZE`and the average runtime is recorded. The `GRID_SIZE`is then decreased by a factor of two to 4,096 and 10 trials are performed. Trials are performed for every decreasing factor of two `GRID_SIZE`until `GRID_SIZE`is 64. A `GRID_SIZE`of 64 is the smallest value that yields any Hotspot output and smaller values result in unexplained runtime errors. Results from these experiments are shown in Figure 4.1.

Figure 4.1 shows that, as expected, as `GRID_SIZE`decreases so does the execution time. The overall speedup, however, is marginal. At most, the runtime is roughly 1.18x faster for a `GRID_SIZE`of 64 versus the recommended 8,192. When going from a `GRID_SIZE`of 8,192 down to 64, an increase in layout temperature of only .52 °C is observed. The decrease in `GRID_SIZE`leads to a less accurate temperature and the results indicates that lower `GRID_SIZE`leads to a higher resulting temperature.

The resulting temperature is only slightly less accurate for smaller `GRID_SIZE`values and the execution time of Hotspot is only 1.18 times faster. It may be a useful to utilize a smaller `GRID_SIZE`to gain small speed ups during preliminary construction of layouts. The over-estimation of the temperatures as a result of a lower `GRID_SIZE`may not be detrimental (and it seems that Hotspot temperatures with a `GRID_SIZE`of 8192 do not exceed the temperature

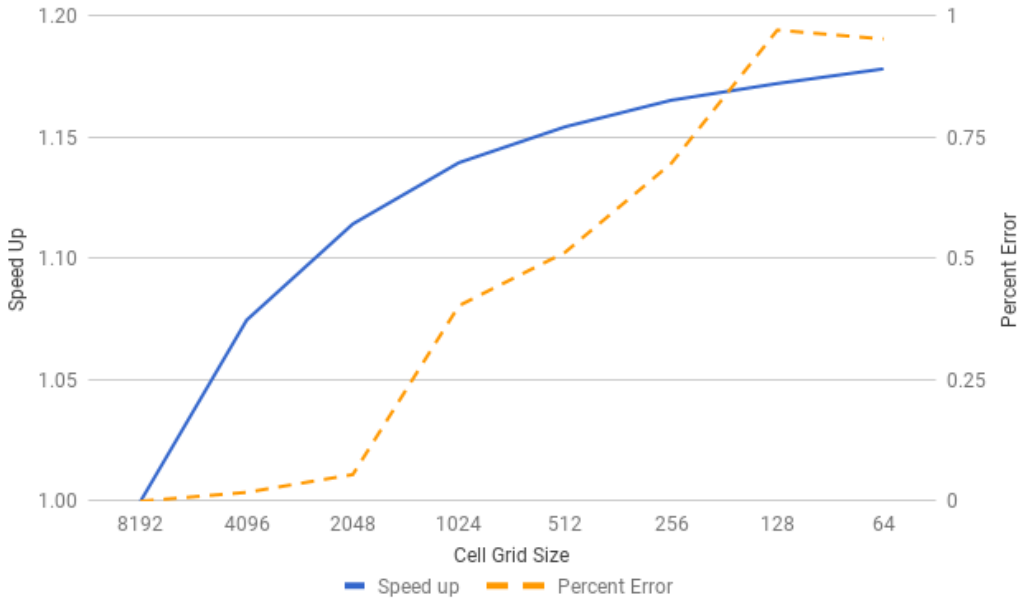



Figure 4.1: Speedup and percent error vs. `GRID_SIZE`.

calculated utilizing a lower `GRID_SIZE`value). But in the end, only marginal Hotspot runtime reductions are possible, and Hotspot thus remains a several computational bottleneck for our work.

## 4.4 Conclusion

SuperLU provides a significant speed up to the runtime of Hotspot. However, further speedup is beyond the scope of this thesis. In future work, looking for an alternative to Hotspot seems a likely possibility. For now, we proceed with a `GRID_SIZE`of 2,048 because it provides a small speed up and leads to low percent error. However, in all the results in this thesis, once layouts are constructed, their temperature is re-evaluated with the maximum `GRID_SIZE`of 8,192 to ensure the most accurate temperature values. Although in this chapter our efforts to speed up Hotspot are mostly in vain, in the next chapter we explain how our heuristic design opens up opportunities for mitigating the Hotspot bottleneck via parallel Hotspot invocations.

# CHAPTER 5
# LAYOUT CONSTRUCTION HEURISTIC

In this chapter we discuss our heuristic in detail. This heuristic uses a simple randomized greedy approach. As explained in Chapter 4, the Hotspot computational bottleneck precludes the use of sophisticated meta-heuristics. Our heuristic can compute solutions in spite of this bottleneck due to its simplicity, but also exploits parallelism to mitigate this bottleneck. Our heuristic proceeds iteratively, and each step consists of random candidate generation, candidate evaluation, and greedy candidate selection.

The chapter is organized as follows. We first introduce the heuristic, the starting-point layout structure, the candidate generation process, and the candidate selection process. We then discuss candidate evaluation, and how it benefit, up to a point, from the introduction of parallel Hotspot invocations.

## 5.1 Heuristic

We wish to maximize the total computational power of the chip arrangement while keeping it within the temperature constraint. Chip layouts determine several properties: network topology characteristics (diameter, ASPL, number of edges), and maximal power within the permissible temperature range. It thus presents a multiobjective or multicriterion optimization problem. We transform this to a single objective function by using lexicographical ordering to induce a total order on the chip layouts [19]. Using this objective function, we apply a heuristic search to obtain layouts with high objective function values. The method of search we implement is what is traditionally called a (1,$\lambda$) evolutionary algorithm [20], where a new chip is randomly added to an existing configuration in one of $\lambda$ different randomly chosen locations. In practice, the value of $\lambda$ affects the overall runtime of our heuristic in that greater values lead to lengthier runtimes. Thusly, in practice, we empirically determine the value of $\lambda$ given an upper bound on the desired runtime of our heuristic. The configuration with the highest objective value of the randomly chosen locations becomes the new configuration for the next iteration. It should be noted that the maximum power value is itself obtained by a binary search to find the largest of a finite set of power values that yield temperatures under the permissible limit.

(a) Strip Cradle. (b) Strip Cradle: Side View.

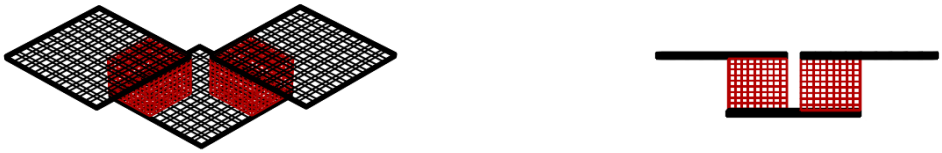
Figure 5.1: Cradle Structure with strip overlap areas: black boxes are chips, red boxes are induction zones.

### 5.1.1 Starting Point "Cradle" layout

Our layout construction algorithm starts with a base 3-chip static layout, which we call the "cradle." (Our layout construction heuristic is only used when the total number of chips in the layout is greater than 3). The cradle, shown in Figures 5.1 and 5.2, consists of 3 chips in which one chip, on level $l$, is connected to the two other chips, which are both on level $l+1$. From a chip fabrication perspective, since inductors can in principle be placed anywhere on a chip's surface, any rectangular shape is possible for the induction zones, or overlap areas. In Figure 5.1, the geometry of the overlap is a rectangular strip. We consider another possible overlap geometry, a square. With square overlap areas chips are overlapped diagonally, as shown in Figure 5.2. Our heuristic allows for overlap geometry to be strip or square in the same layout. In fact, it is possible to have a 3-chip cradle that contains both one strip overlap and one square overlap. Due to crosstalk, constraining all overlap regions to be strips results in fewer options for future chip additions, because each chip can have a maximum of 2 connections to neighboring chips. By contrast, a square overlap geometry allows for 4 potential connections from one chips to neighbor chips. Unless otherwise specified, we construct layouts that can include both square and strip overlap geometries.

### 5.1.2 Candidate Generation

After the cradle has been generated, the remaining chips are added iteratively in a random, greedy fashion. To add the next chip, the heuristic randomly generates a number of feasible



(a) Square Cradle. (b) Square Cradle: Side View.

Figure 5.2: Cradle Structure with square overlap areas: black boxes are chips, red boxes are induction zones.

candidate positions for that chip. Each such position is defined by its level and its $x$ and $y$ coordinates, and guarantees no collisions with other existing chips in the layout and no crosstalk. The candidate positions are stored in a list, and the larger that list the broader the random search of the layout design space performed by our greedy heuristic. A command-line argument to our layout construction implementation allows for the specification of the number of chip candidate positions to be generated at each iteration.

### 5.1.3 Candidate Selection

Once a list of candidate chip positions has been constructed, each candidate is evaluated for selection. For each candidate position, a new chip is temporarily added to the current layout at that position. The resulting temporary layout is evaluated in terms of network topology characteristics (diameter, ASPL, number of edges), and in terms of the largest power (i.e., chip frequencies) that can be sustained without overcoming a specified temperature threshold. The latter involves invoking Hotspot. These metrics are appended to a results list, the candidate chip is removed from the layout, and this process continues until all candidates in the candidate list have been evaluated.

The results list is then evaluated to pick the "best" candidate position. Once the best candidate position has been determined it is permanently added to the layout and the candidate generation and selection process repeats until the layout contains the desired number of chips. Candidates are picked using a lexicographical order, picking candidates based on power, diameter, number of edges, ASPL, and temperature. In other terms, ties for

one metric are broken using the next metric in the prioritization order. The prioritization of these layout characteristics is important and different prioritizations produce different results as our heuristic is more or less greedy w.r.t different metrics. We explore different prioritization options, e.g., prioritizing first by power, temperature, or network. We also experiment with an adaptive prioritization scheme that allows the heuristic to change the prioritization order as the layout grows. The idea is to have the heuristic pick for the best network until the layout temperature gets close to the maximum temperature threshold, at which point the prioritization scheme is changed to picking for the best temperature or power. Given a parameter $0 \leq \alpha \leq 1$, if the layout temperature reaches $\alpha T$, where $T$ is the maximum temperature threshold, then the heuristic shifts from greedily picking for a good network and picks for either a higher chip power or lower temperature. For example, if alpha threshold is set to .75 and the maximum allowed temperature is 50 °C, then the heuristic will greedily pick for the best network as long as the layout temperature remains below 37.5 °C.

### 5.1.4 Challenges

As expected, generating more candidates at each iteration leads to better end results, simply because the sample size from which the selection portion of the heuristic can choose from is bigger (i.e., a broader search). However each candidate evaluation involved a Hotspot invocation. Given that Hotspot is a severe computational bottleneck for our approach, the layout construction time is directly proportional to the number of candidates. As seen in the previous chapter, our attempts to speed up a single invocation of Hotspot yield at best marginal improvements. But while we are unable to speed up an individual invocation, we can place multiple invocations in parallel, which we discuss in Section 5.2. Another option is to reduce the number of Hotspot invocations placed by our heuristic by adding more than one chip at a time to the layout, which we discuss in Section 5.3

## 5.2 Parallel Hotspot Invocations

### 5.2.1 Multithreading

Using Python's threadpool abstraction as provided in the multiprocessing module, it is straightforward to launch parallel temperature evaluations for candidate chip evaluations, so as to reduce the Hotspot bottleneck by some constant factor. In theory, assuming maximum

parallel efficiency, this constant factor should be equal to the number of physical cores on the machine used to run our layout construction algorithm. More formally, if there are $N$ candidates to evaluate, each evaluation takes time $T$, and there are $p$ cores, one should be able to perform all evaluations in time $\lceil \frac{N}{p} \rceil T$.

The above approach leads to noticeable speedups but we encountered various issues, mostly having to to with logistical details of the Hotspot API implementation. When simulations were run on multi-core machines using more than 16 threads, we encountered "File Busy Error" occurrences. This is due to threads competing for the same executable files. Assigning each thread their own input and executable files with the help of their own unique process identification, PID, solved the problem for the majority of the simulations, but not entirely. Although the threadpool implementation could not scale without errors, it proved that multiple Hotspot evaluations can be performed concurrently and that some speedup can be achieved. Rather than investing time and energy in addressing the above errors with the current implementation of the Hotspot API, we instead explored other means of parallelization that would suffer from these errors.

### 5.2.2 Message Passing Interface

We turned to Message Passing Interface, MPI, as another means of performing concurrent Hotspot evaluations. A master-worker MPI model was implemented in which the master would generate candidates and assign the workers to evaluate a specific candidate layout with Hotspot. The worker would perform the evaluation and send the results back to the master thread. The master thread would aggregate all the results form the workers, pick the best candidate, and stop all the worker threads. This process repeat until all chips have been added to the layout. While MPI is typically intended for distributed-memory computation, i.e., across multiple machines, were use it here one a single multi-core machine.

The MPI version of our heuristic was implemented with Python's mpi4py module. Preliminary testing indicated observable speedups with none of the errors described in the previous section: when scaling up the number of MPI worker threads and we seem to avoid the file busy errors that was encountered when using the multiprocessing module. We perform a set of experiments to measure the layout construction time for different numbers of MPI worker threads, which allows us to measure parallel speedup and efficiency. The results for theses experiments are shown in Figure 5.3. These results are obtained on 16-core machine (2.4GHz, with hyperthreading) with a 8MiB L3 cache size and 23GiB or RAM.

The main observation from Figure 5.3 is the loss of parallel speedup and efficiency as

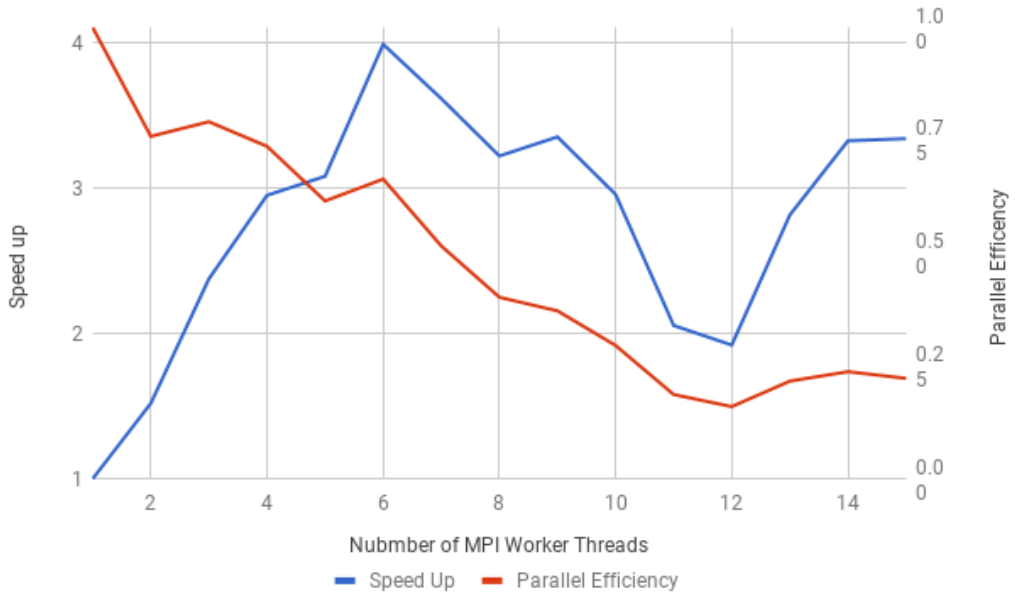



Figure 5.3: Speedup and parallel efficiency vs. the number of MPI worker threads invoking Hotspot.

the number of workers increases. While some drop in speedup and parallel efficiency as the number of threads increase is always to be expected (e.g., due to communication latency and process creation overhead), the observed drops are more than expected. The results from this experiment indicate that using 6 to 8 worker threads leads to the highest speedup on this machine. Parallel efficiency suffers greatly when more than 7 worker threads are used and drops below 50%. In summary, the fastest layout construction times are achieved without utilizing all the processing power of this machine.

Due to our suspicions that drastic decreases in parallel efficiency is cache-related, we set up and experiment to evaluate how varying the number of Hotspot instances affect the cache usage (Level 1, L1, and Lowest Level Cache, LLC). One Hotspot instance was run and the Linux `perf` tool was used to capture runtime and cache usage metrics. This process was repeated and the number of Hotspot instances was increased by powers of 2 until there was one Hotspot instance running on each core of our 16-core machine. Results from this experiment are shown in Figure 5.4.

The results in Figure 5.4 indicate a drastic increase in runtime of each Hotspot instance as the number of concurrent instances increases. There is also an increase in the number of LLC load misses as the number of concurrent Hotspot instance increases. The LLC cache is shared amongst all the cores and is only 8MiB on this machine. A high number

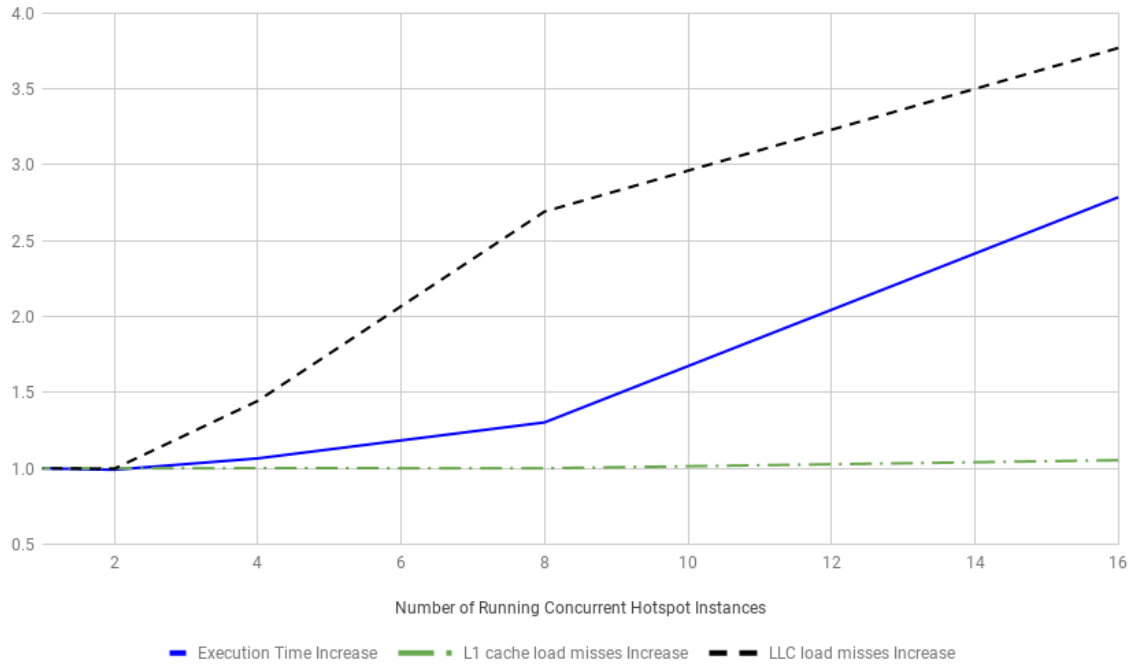



Figure 5.4: Increasing number of Hotspot instances affect on runtime and L1 and LLC cache misses.

of misses indicates collisions, which lead to longer runtimes. The L1 cache load misses remains relatively stable and low as the number of concurrent Hotspot instances increases. This is because the L1 cache is private to each individual core. It also should be noted that the slope of the execution time increases with 8 Hotspot instances which explains the drop off in speedup, seen in Figure 5.3, when more than 8 MPI worker threads are used. We conclude that the LLC cache size is the limiting factor and leads to the MPI bottleneck. One solution to this problem would be to run our simulation on another machine with bigger cache sizes. Upgrading to better hardware to run faster is expensive and not a practical solution. Another solution is to execute our layout construction procedure on a cluster of multiple machines. Regardless, to further reduce the Hotspot bottleneck, in the next section we discuss modifications to our random greedy heuristic so that it can invoke Hotspot less frequently.

## 5.3 Lowering Number of Hotspot Invocations

The Hotspot simulator is invoked for each candidate chip position at each step of our heuristic. The addition of the next chip is dependent on the resulting layout temperature from the addition of the chip before it. Therefore, if our heuristic uses $c$ candidates at each steps and is used to construct a layout with $p \geq 3$ chips, then it will invoke Hotspot $c(p-3)$ times. The execution time is thus linear in both the number of candidates and the number of chips in the layout. As seen in the previous section, using a multi-core machine to parallelize candidate evaluations is only moderately successful due to LLC cache competition among processes. While throwing more hardware at this problem can help (larger LLC cache, multiple machines in a cluster), in this section we instead explore possible modifications to our greedy heuristic that will reduce the number of Hotspot invocations.

### 5.3.1 Multiple Chip Additions

Since layout construction time is driven by Hotspot invocation time, reducing the number of Hotspot invocations should reduce layout construction time. One way to accomplish this is to add multiple chips at a time before performing a heat evaluation on the layout being constructed. For instance, we can add chips 2-by-2 or 3-by-3 rather than 1-by-1, as done with the current implementation. However, adding multiple chips at a time runs the risk of producing less desirable layouts as it makes the search performed by the heuristic less local.

### 5.3.2 Static Sub-Layout Additions

Along a similar line of logic for adding multiple chips at a time we explore adding multiple chips in a multi-chip structure to the layout at a time. Our heuristic starts with the 3-chip cradle structure and adds chips accordingly until it reaches the desired number of chips. We use the cradle as a starting point to start because it offers many options for adding new chips and does not suffer from crosstalk. The cradle also consist of 2 levels which helps to enforce a small degree of sparseness in the layout, which may be good for heat dissipation. As a result, we explore a cradle-by-cradle chip addition approach for our layout construction, which is special case of a 3-by-3 chip addition approach. The cradle-by-cradle approach reduces the randomness of our heuristic when compared to a normal 3-by-3 addition. When adding 3-by-3, the heuristic finds random positions of all 3 chips to be added. The position of the 3 chips in a normal 3-by-3 addition are irrespective of each other which is not the case in a cradle-by-cradle addition. During a cradle-by-cradle addition, only one random search for

a new chip position is required to place one chip in the cradle. Once this chip is added at the randomly found position, the remaining chips of the cradle are added with respect to that first chip's position. A small degree of randomness is introduced when determining if the first chip is one of the edge chips or the middle chips of the cradle. Lastly, the heuristic iterates through the proper checks to ensure that the position of the remaining cradle chips are feasible and connects them if they happen to also have proper overlaps with other chip without crosstalk. The random search process has potential to be time consuming and thus reducing the number of random searches by adding a static structure, like the cradle, may also lead to speed up when compared to a 3-by-3 addition approach.

# CHAPTER 6
# RESULTS

In this chapter we evaluate the efficacy of our heuristic, as described in Chapter 5 and as implemented in our Python framework, using the Hotspot simulator to evaluate the temperature of layouts as they are being constructed. We refine certain aspects of our heuristic experimentally by running different heuristic variants over various trials. The results give us insight into which variant tends to yield the best layout metrics and we proceed, in future simulations, with this variant. Layouts generated by our heuristic are then compared against baseline stack and checkerboard layouts.

We discuss the simulation framework used for building layouts in Section 6.1. In Section 6.2 we define our simulation assumptions (i.e, chip specification, temperature thresholds, desired bandwidth). We evaluate the baseline layouts in Section 6.3. In Section 6.4 we experimentally evaluate variants of our heuristic. Finally, in Section 6.5 we present results obtained with our heuristic and compare them to the baseline layouts.

## 6.1 Layout Evaluation Framework

As mentioned in previous chapters, the majority of our simulation software is written in Python 2.7. When evaluating the quality of a constructed layout we have two main points of concern: the layout temperature and the network topology. The temperature is evaluated using the Hotspot thermodynamic simulator discussed in Chapter 4 and network evaluations are performed by the Python networkx module. An object-oriented approach is used to construct layouts, which helps with the evaluation of layouts and the storage of layout metrics. A Chip class contains the properties of microprocessor chips in the layouts we construct, and a Layout class contains layout properties in various list data structures. When a layout is ready for Hotspot evaluation, a file is generated that contains the chip's dimensions, the $(x, y, z)$ coordinates and rotation (0 or 90 degrees) of each chip. This file is passed to Hotspot, which then computes the temperature at evenly spaced physical grid points throughout the layout. Our evaluation framework is only concerned with the highest such temperature, which must remain below a prescribed temperature threshold.

### 6.1.1 Simulation of TCI, Wireless Communication

The use of TCI allows for wireless communication and our simulation frameworks accounts for the inductors and their influence on layout construction. Inductors are placed at locations where two chips overlap sufficiently for communication, and such locations are designated as induction zones. An induction zone contains a large number of individual inductors, but in all that follows we simply use the term "inductor" to mean "a set of inductors in the same induction zone that together connect one chip to another chip". An inductor position list within the Layout class stores the location (i.e., level and $(x, y)$ coordinates) of each inductor. The level of an inductor is defined as the level of the lowest-level chip it helps to connect. In other words, if there is an inductor connecting a chip on level 1 to a chip on level 2 then the inductor is considered to be on level 1.

In our simulation framework we account for the location of inductors, and chips are only deemed connected if they overlap sufficiently and an inductor is located between them in that overlap region. As a result, it is possible to have two chips with sufficient overlap but no inter-connection because of the absence of an inductor. This scenario is unfavorable because it implies that chips are in close proximity for no beneficial reason: the network is not improved but temperature is likely increased due to chips being in close proximity. This situation, however, may occur due to crosstalk (see Section 2.2.2).

A method is implemented to detect the presence of crosstalk within the layout. This is accomplished by iterating through all inductors and looking if any inductors are one level apart and overlap each other in any way. Recall that crosstalk only creates interference with inductors that are one level away. The above method is always called before adding a new chip to a layout. If crosstalk is detected, then the chip is not be added and a new attempt must be made.

### 6.1.2 Heuristic Implementation

As mentioned in Chapter 5, our heuristic starts with a *cradle* structure (3 chips, 2 inductors, no crosstalk). Our heuristic relies on thermodynamic heat simulation using Hotspot as well as network evaluations. As mentioned in Chapter 4, Hotspot simulations are time consuming. Yet, the efficacy of our greedy approach is dependent on the number of random, prospective layout that can be generated and evaluated. The use of MPI (see Section 5.2.2) allows for more heat simulations in a given amount of time, which allows our heuristic to produce more competitive layouts.

Table 6.1: Specification of our baseline chip.

| | |
|---|---|
| Processor family | x86-64 |
| Number of cores | 4 |
| L1 I/D cache size | 32/128 KB (line:64B) |
| L1 cache latency | 1 cycle |
| L2 cache bank size | 16 MB (assoc:4) |
| L2 cache latency | 6 cycles |
| Memory size | 4 GB |
| Memory latency | 160 cycles |
| Area | 169 $mm^2$ (square) |
| Minimum Power | 14.01 Watts @ 1.2 GHz |
| Maximum Power | 56.72 Watts @ 3.6 GHz |

Early simulation designs bundled the candidate generation, evaluation, and selection process into one bulky function which is now split up into their own methods. Having separate functions for each step of the heuristic allows for proper coordination and synchronization when generating layouts. The Python Hotspot API also required refactoring to include process identification numbers to avoid file collisions, competition for executable files, and race conditions. As a result our simulation is able to construct multi-chip layouts and evaluate them on the basis of heat generation and network characteristics faster through parallelism afforded by MPI.

## 6.2 Simulation Assumptions

Command-line arguments are passed to our simulation framework to configure the simulation conditions. We describe here the default set of simulation parameters we use in most of the experiments described in the remainder of this thesis. All of our layouts are constructed with the same chip, whose specifications are given in Table 6.1. We consider a 16-tile chip, the specifications of which are based on those of the Xeon E5-2667 v4 chip, for which we have detailed power profiles. Certain modifications were made, which include allowing a maximum DVFS level of 56.72 watts at 3.6 GHz so as to be representative of currently available chips in the consumer market. Also, we assume that the chip is square. Although our heuristic and simulation framework can handle non-square rectangular chips, in this thesis we only present results in the square case (so as to reduce the already large layout design space). While we contend that our heuristic is effective (especially w.r.t. baseline layouts) for arbitrary numbers of chips, we present results for layouts with 6, 9, and 13 chips.

A threshold of 50 °C is chosen as the maximum operating temperature for any chip

in our layout. If any physical point in the layout exceeds this threshold, we consider the resulting layout infeasible and the power of the chips (i.e., their frequencies) must be scaled back or a new layout must be created. Note that vendors often specify higher temperature thresholds (around 70 °C), however, recent high-end server products have more conservative temperature thresholds, and many "cool" data centers enforce even lower thresholds. As mentioned in Chapter 2, various cooling media have been explored and offer their own set of pros and cons. Our simulation is capable of simulating the use of different coolants which include water, mineral oil, and Fluorinert. In this thesis, all our simulations assume air cooling, which is the most pervasive and economical cooling approach to date.

We have the option to constrain the inductor geometry to a rectangular strip or a square and our heuristic can use both strip or square inductor geometries in a single layout. The baseline layouts, stack and checkerboard, cannot be constructed with a strip inductor geometry if crosstalk is to be avoided. Given crosstalk and our inductor geometries, the overlap between chips that allows them to be connected via an inductor is less than 25%. An overlap greater than 25% would drastically reduce the number of possible connections a chip can make to neighboring chips (due to physical collisions between those chips). Furthermore, high bandwidth can be achieved with less than 25% overlap. We run simulations with 10% and 20% overlap, which provides sufficient communication bandwidth without overconstraining the number of inter-chip connections a single chip can have. Our chip area is 169 $mm^2$ (see Table 6.1). An overlap area of 10% for this chip, equates to 16.9 $mm^2$ inductor. If we apply the findings that a 675$\mu$m $\times$ 675$\mu$m induction area results in 64 Gbps bandwidth [14] (see Chapter 2), then we could conceivably achieve a bandwidth of $0.1 \times 169\text{mm}^2 \times 64\text{Gbps}/(675)^2\mu\text{m}^2 =$ 2.374 Tbps. A 0.2 overlap would double this bandwidth. These bandwidths are high and, in fact, it is likely that useful systems could be designed with lower bandwidths. However, in this thesis we consider these high-bandwidth scenarios as they correspond to high-performance systems for which temperature concerns are then more prevalent.

## 6.3 Baseline Layouts

Two known previously considered multi-chip layouts are the stack and checkerboard layouts. Each has its own advantages and shortcomings. We use these layouts as baselines for the purpose of evaluating our heuristic in terms of network topology quality, power output, and operating temperatures. Our goal is for our proposed heuristic to overcome the shortcomings of the baseline layouts, and more broadly to produce the best layout possible given chip,

overlap, and temperature threshold specifications.

The stack layout is unique in that there is a 100% overlap amongst all the chips contained in the layout. As a result, there is sufficient overlap to assume, with TCI, full connection and maximum bandwidth. However, crosstalk does not allow inductors to cover the whole surface of the top or bottom of a chip within this layout. Inductors must be reconfigured or staggered on the chips in order to prevent crosstalk. This leads to an in efficient layout because the full overlap does not benefit communication and ultimately leads to higher temperatures. Therefore, while it is possible to use TCI for inter-chip communication in a stack layout, TSV is a likely better option and can provide maximum bandwidth without crosstalk interference (without resolving the temperature issue though). TSV is what is used in current stack layouts used for production chips.

Table 6.2: Stack layout results. All layouts are above the temperature threshold even if chips are operated at their lowest possible frequencies.

| numchips | edges | levels | diameter | ASPL | power | frequency | temperature |
|---|---|---|---|---|---|---|---|
| 13 | 12 | 13 | 12 | 4.67 | N/A | N/A | N/A |
| 9 | 8 | 9 | 8 | 3.33 | N/A | N/A | N/A |
| 6 | 5 | 6 | 5 | 2.33 | N/A | N/A | N/A |

One issue with TCI-based stack layouts is network topology scalability as the number of chips increases. For example, as the number of chips grows, the diameter of the inter-chip network also grows, which increases maximum inter-chip latency. More importantly, due to high overlap, and regardless of whether TCI or TSV is used, a stack layout suffers from high temperature. Table 6.2 presents results for the stack layouts containing 6, 9, and 13 chips. Temperature, frequency, and power metrics are unavailable for these layouts because they exceed our temperature threshold. Setting the chips in these layouts to the lowest DVFS level, i.e., the lowest 1.2GHz frequency, still results in layouts that are too hot. It is not possible to build a stack layout, assuming air cooling, with 6 or more chips given our simulation assumptions. The bottom line is that stack layouts are not viable for building integrated 3D multi-chip systems with more than a few chips.

The checkerboard is another baseline layout that has been recently proposed [3]. By comparison, this layout proves to be superior to the stack while naturally avoiding crosstalk. Overlap areas are smaller, which leads to a sparser physical layout and thus better heat dissipation. Each chip also makes more connections to neighboring chips, so that the topology graph has more edges and higher vertex degree.

In comparison to the stack layout, the checkerboard layout offers a better network. While the stack layout does allow for a potentially higher bandwidth, the checkerboard makes up for it with a better connected network with a smaller diameter, more edges, and smaller ASPL. Furthermore, as discussed in Section 6.2, even a 10% overlap area provides high inter-chip bandwidth given current TCI technology. The checkerboard layout is also geometrically sparser, which enables higher chip operating frequencies and/or lower temperatures when compared to a stack layout.

Table 6.3: Checkerboard Metrics

| numchips | overlap | edges | levels | diameter | ASPL | power | frequency | temperature |
|---|---|---|---|---|---|---|---|---|
| 13 | 0.2 | 16 | 2 | 4 | 2.46 | 25.7721 | 2.40 | 47.48 |
| 9 | 0.2 | 12 | 2 | 4 | 2.00 | 41.9991 | 3.20 | 48.40 |
| 6 | 0.2 | 6 | 2 | 3 | 1.73 | 56.7207 | 3.60 | 48.44 |
| 13 | 0.1 | 16 | 2 | 4 | 2.46 | 33.0215 | 2.80 | 47.91 |
| 9 | 0.1 | 12 | 2 | 4 | 2.00 | 41.9991 | 3.20 | 44.52 |
| 6 | 0.1 | 6 | 2 | 3 | 1.73 | 56.7207 | 3.60 | 45.02 |

The results in Table 6.3 shows results for the checkerboard layout, which is clearly superior to the stack layout. It is possible to build checkerboard layouts that operate within our temperature threshold for all considered numbers of chips. The temperature is cooler, which allows for higher operating frequencies and power outputs when compared to the stack layout (which actually cannot operate at all under the same conditions). Note that, compared to the stack layout, the network topology characteristics of the checkerboard layout are more scalable (e.g., the diameter is not increased when going from 9 chips to 13 chips). In the end, the checkerboard is the better of the two baseline layouts and we use it as our baseline for comparison when evaluating the performance of our heuristic.

## 6.4 Heuristic Tuning Experiments

Our simulation framework takes several command-line arguments to specify parameters that modify the behavior of our proposed heuristic. It is thus possible to experiment with several variants of our heuristics. In this section, we conduct experiments to determine values of these heuristic configuration parameters that work well in practice. More specifically:

- We first evaluate the greedy selection criterion and prioritization for selecting candidate layouts, so as to determine which layout characteristics are most important (e.g., should

the heuristic greedily search for a better network first or for a cooler layout first). (See Section 5.1.3.)

- As a follow up to the above exploration of the greedy selection criteria, we investigate the use of an adaptive greedy selection approach that can switch between prioritization schemes throughout layout construction. (See Section 5.1.3.)
- Lastly, we experiment with chip addition strategies that involves adding multiple chips at a time as opposed to adding chips only one at a time. The motivation is to try and reduce the number of Hotspot calls, so as to allow more exploration of the design space given some layout construction time constraint. (See Section 5.3.1.)

We run our simulations on a 16-core machine but, due to the cache collision issue described in Chapter 5, we only use 7 MPI worker threads to run concurrent Hotspot executions. Simulations are run for 10 trials and the averages of these trials are reported in the next sections.

### 6.4.1 Candidate Selection Criteria

Our heuristic achieves trade-offs between the desired objectives (high power, good network, low temperature) by selecting the best layout among candidate layouts at each step. As described in Section 5.1.3, this selection is done systematically as follows. The heuristic picks the best layout using a candidate layout selection criterion that corresponds to one of the objectives, breaking ties with another one of the objectives, and finally breaking ties with the last objective. The heuristic can be configured to use a particular criterion order, and different orders can lead to very different result layouts. The `pick_by` command-line argument to our simulation framework allows the user to decide on which criterion to apply first (the order of the other two criteria is arbitrarily fixed in the current implementation). In this section, we experimentally determine which criterion should be applied first so as to obtain the best overall layouts. 10 trials for each of the 3 options are executed, in which layouts are constructed with 6, 9, and 13 chips using a .1 and a .2 overlap. Results are presented in Table 6.4. The average coefficient of variation, CV, for the edges, levels, diameter, ASPL, power, and temperature are 5.40%, 19.31%, 9.79%, 5.93%, 5.74%, 2.44%, and 3.60% respectively. In addition, the max CVs for edges, levels, diameter, ASPL, power and temperature are 11.40%, 29.57%,19.91%,12.02%,15.88%,6.15%, and 5.31% respectively.

The results in Table 6.4 indicate that picking by power first yields the best network results overall. For all layout sizes (6, 9, and 13 chips), picking by power first leads to the

Table 6.4: Results from different candidate layout selection criterion for 6, 9, and 13 chip layouts with a .1 and .2 overlap. Note that non-integer values are the result of averaging over 10 trials. Best network and power metrics are shown in boldface.

| Layout Size (Chips) | Overlap | Selection Criterion | Edges | CV | Levels | CV | Diameter | CV | ASPL | CV | Power (W) | CV | Frequency (GHz) | CV | Temperature (C) | CV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 0.1 | network | 5.30 | 9.11% | 3.10 | 18.31% | 3.70 | 13.06% | 1.987 | 6.09% | 56.7207 | 0.00% | 3.60 | 0.00% | 42.914 | 3.62% |
| 6 | 0.1 | power | **6.00** | 0.00% | 2.30 | 21.00% | **3.00** | 0.00% | **1.733** | 0.00% | **56.7207** | 0.00% | **3.60** | 0.00% | 45.463 | 3.68% |
| 6 | 0.1 | temp | 5.00 | 0.00% | 4.00 | 20.41% | 4.80 | 8.78% | 2.287 | 4.36% | 56.7207 | 0.00% | 3.60 | 0.00% | 39.813 | 2.24% |
| 6 | 0.2 | network | 5.20 | 8.11% | 3.10 | 18.31% | 3.60 | 14.34% | 2.000 | 7.37% | 56.7207 | 0.00% | 3.60 | 0.00% | 44.925 | 5.28% |
| 6 | 0.2 | power | **6.00** | 0.00% | 2.80 | 15.06% | **3.00** | 0.00% | **1.733** | 0.00% | **56.7207** | 0.00% | **3.60** | 0.00% | 47.657 | 2.23% |
| 6 | 0.2 | temp | 5.00 | 0.00% | 4.20 | 15.06% | 5.00 | 0.00% | 2.333 | 0.00% | 56.7207 | 0.00% | 3.60 | 0.00% | 42.099 | 2.38% |
| 9 | 0.1 | network | 9.40 | 11.44% | 4.00 | 16.67% | 4.80 | 16.43% | 2.400 | 9.44% | **56.7207** | 0.00% | **3.60** | 0.00% | 46.354 | 3.79% |
| 9 | 0.1 | power | **11.10** | 2.85% | 3.30 | 14.64% | **4.00** | 0.00% | **2.050** | 0.86% | 55.2485 | 8.43% | 3.56 | 3.55% | 48.037 | 3.72% |
| 9 | 0.1 | temp | 8.30 | 8.13% | 4.90 | 29.57% | 7.40 | 13.06% | 3.117 | 11.60% | 56.7207 | 0.00% | 3.60 | 0.00% | 41.890 | 3.79% |
| 9 | 0.2 | network | 9.10 | 9.62% | 3.60 | 14.34% | 5.40 | 19.91% | 2.533 | 12.02% | 47.8877 | 15.88% | 3.36 | 6.15% | 45.283 | 3.47% |
| 9 | 0.2 | power | **10.90** | 5.21% | 3.40 | 15.19% | **4.40** | 11.74% | **2.111** | 3.04% | 52.3042 | 13.60% | 3.48 | 5.55% | 47.676 | 4.38% |
| 9 | 0.2 | temp | 8.00 | 0.00% | 6.00 | 26.06% | 7.50 | 9.43% | 3.211 | 5.34% | **56.7207** | 0.00% | **3.60** | 0.00% | 43.654 | 3.40% |
| 13 | 0.1 | network | 14.60 | 10.81% | 4.30 | 19.15% | 6.30 | 10.71% | 2.921 | 9.40% | 47.8877 | 15.88% | 3.36 | 6.15% | 46.556 | 5.31% |
| 13 | 0.1 | power | **16.80** | 6.76% | 3.80 | 20.76% | **5.60** | 15.06% | **2.562** | 5.69% | 46.4156 | 15.32% | 3.32 | 5.82% | 47.038 | 4.44% |
| 13 | 0.1 | temp | 12.30 | 3.93% | 6.00 | 28.33% | 10.80 | 11.38% | 4.262 | 8.61% | **55.2485** | 8.43% | **3.56** | 3.55% | 43.447 | 2.81% |
| 13 | 0.2 | network | 14.80 | 6.98% | 4.00 | 16.67% | 6.30 | 13.07% | 2.882 | 6.87% | 41.1013 | 6.91% | 3.16 | 4.00% | 46.718 | 3.07% |
| 13 | 0.2 | power | **16.70** | 7.50% | 3.70 | 18.24% | **5.20** | 8.11% | **2.523** | 5.93% | 40.6883 | 7.35% | 3.14 | 4.30% | 48.403 | 3.34% |
| 13 | 0.2 | temp | 12.50 | 6.80% | 6.20 | 19.83% | 10.60 | 11.07% | 4.164 | 10.08% | **53.7764** | 11.54% | **3.52** | 4.79% | 46.529 | 3.92% |

largest number of edges, the lowest diameter, and the lowest ASPL. The number of levels is also lower, which intuitively makes sense since the number of levels is a lower bound on the diameter. Picking by network seems to be the next best option in terms of resulting inter-chip networks. While it is not as good as picking by power, it beats out picking by temperature, which yields the worst layouts with respect to network topology metrics.

Considering chip frequency/power, there is no difference between the 3 candidate selection approaches for 6-chip layouts. However, for larger layouts, differences are observed. For theses larger layouts, picking by power first does not result in layouts with the highest chip frequencies and power. In fact, picking by power first leads to the lowest power for all 13-chip layouts and to the second best power for the 9-chip layout with a .2 overlap. Prioritizing good network characteristics instead lead to the second best temperature and operating chip frequency. Picking by temperature first leads to the highest power.

Some of the results above are counter-intuitive, as optimizing for some criterion first is not necessarily the best option to optimize that very criterion. This is due to the complexity and non-linearity of the temperature behavior, and to the fact that our greedy heuristic can be stuck in local minima. Regardless, we experience the typical trade-off where we are unable to achieve both a good network and high power, i.e., high chip frequencies, at the same time. Typically, the poorer inter-chip network allows for the highest chip operating frequency. This is because these poorer networks allow for sparser physical layouts, which in turn allow for better heat dissipation.

Rather than trying to resolve the above trade-off arbitrarily without a frame of reference,

we instead consider the behavior of the baseline checkerboard layout, and determine which candidate layout selection criterion should be used to obtain layouts that compare favorably to this baseline. To this end, we compare the resulting layout metrics from our experiments in Table 6.4 to the baseline checkerboard metrics in Table 6.3. Picking by power first results in layouts with comparable inter-chip network properties with the advantage of a higher operating power and frequency. Furthermore, this option is more advantageous, when compared to the checkerboard layout, as the number of chips in the layout grows. The by network first and by temperature first options result in cooler layouts where each chip can be clocked to a higher frequency when compared to the checkerboard. Unfortunately, these options do not result in layouts with superior network properties. Overall, our results suggest that our heuristic benefits from selecting candidate layouts first based on higher power so as to outperform checkerboard layouts according to both power and network metrics.

### 6.4.2 Adaptive Selection Criterion

In an attempt to improve our heuristic we consider the use of an adaptive candidate layout selection approach, as described in Section 5.1.3. The logic entails having the heuristic first use one candidate layout selection criterion and then switch to another layout selection criterion. Our results from the previous experiment indicate that our heuristic can generate layouts, for all considered sizes and overlaps, that are equivalent to or better than checkerboard layouts with respect to power. However, most heuristic-generated layouts do not strictly outperform the checkerboard layout in terms of the inter-chip network topology. In this section we determine whether an adaptive candidate selection approach can improve the network while still producing layouts with high chip powers. Recall that this approach first aims for some objective as long as the overall layout temperature remains relatively cool. When the layout temperature is too high (e.g., close to the temperature threshold), the heuristic then changes its candidate selection criterion and begins selecting candidates that result in a cooler layout. In a nutshell, the heuristic is aggressive at first, and then later becomes risk-adverse (where the risk is to exceed the temperature threshold). The point at which the candidate selection criterion switches is determined by a parameter, $\alpha$, which has a value between 0 and 1. The product of the temperature threshold and $\alpha$ determine the temperature at which the selection criterion switches.

We determine the best value for $\alpha$ experimentally by building layouts using our heuristic over a range of $\alpha$ values. Initially, the heuristic should aim for a good inter-chip network topology. Recall from the previous section that, surprisingly, selecting candidates based on

highest power ultimately leads to the best network topologies. This is by contrast with selecting candidates based on best network topology characteristics, which turns out not to produce the layouts with the best network. In this section, for completeness, we run experiments in which the heuristic initially selects for power and then switches to selecting for temperature, as well as experiments in which the heuristic initially selects for network and then switches to selecting for temperature. We terms these approaches "power then temperature" and "network then temperature", respectively. Similar to the experiments in the previous section we run 10 trials for each value of $\alpha$ for 6- and 9-chip layouts with a .2 overlap. We consider $\alpha = 0.00, 0.25, 0.50, 0.75, 1.00$. The average coefficient of variation, CV, for the edges, levels, diameter, ASPL, power, and temperature are 8.41%, 14.78%, 4.75%, 2.75%, 2.17%, 0.93%, and 2.86% respectively. In addition, the max CVs for edges, levels, diameter, ASPL, power and temperature are 14.14%, 19.56%, 10.77%, 5.94%, 13.40%, 5.74%, and 5.66% respectively.

Table 6.5: "Power then temperature" results for different $\alpha$ values for 6- and 9-chip layouts with .2 overlap. Values are shown as averages over 10 trials. Best network and power metrics are shown in boldface.

| Layout Size (Chips) | Overlap | Alpha | Edges | CV | Levels | CV | Diameter | CV | ASPL | CV | Power (W) | CV | Frequency (GHz) | CV | Temperature (C) | CV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 0.2 | 0.00 | 5.00 | 14.14% | 4.70 | 14.18% | 5.00 | 8.43% | 2.33 | 4.27% | 56.721 | 0.00% | 3.60 | 0.00% | 40.54 | 2.33% |
| 6 | 0.2 | 0.25 | 5.00 | 10.33% | 4.65 | 16.96% | 5.00 | 0.00% | 2.33 | 0.00% | 56.721 | 0.00% | 3.60 | 0.00% | 40.54 | 2.81% |
| 6 | 0.2 | 0.50 | 5.00 | 6.32% | 4.67 | 16.90% | 5.00 | 0.00% | 2.33 | 0.00% | 56.721 | 0.00% | 3.60 | 0.00% | 40.40 | 1.85% |
| 6 | 0.2 | 0.75 | 5.00 | 10.33% | 4.73 | 14.11% | 5.00 | 6.32% | 2.33 | 2.71% | 56.721 | 0.00% | 3.60 | 0.00% | **40.27** | 2.63% |
| 6 | 0.2 | 1.00 | **5.89** | 0.00% | 4.42 | 11.68% | **3.76** | 0.00% | **1.95** | 0.00% | **56.721** | 13.40% | **3.60** | 5.74% | 41.18 | 5.66% |
| 9 | 0.2 | 0.00 | 8.20 | 12.13% | 6.00 | 19.56% | 7.70 | 6.27% | 3.23 | 4.07% | 55.249 | 0.00% | 3.56 | 0.00% | 42.09 | 2.09% |
| 9 | 0.2 | 0.25 | 8.15 | 5.93% | 6.05 | 17.07% | 7.80 | 6.62% | 3.26 | 4.36% | 55.985 | 0.00% | 3.58 | 0.00% | 41.92 | 3.93% |
| 9 | 0.2 | 0.50 | 8.10 | 8.73% | 6.23 | 17.25% | 7.83 | 10.77% | 3.27 | 5.59% | 56.230 | 8.28% | 3.59 | 3.53% | **41.75** | 2.84% |
| 9 | 0.2 | 0.75 | 8.08 | 8.66% | 6.15 | 11.50% | 7.83 | 9.04% | 3.27 | 5.94% | **56.353** | 0.00% | **3.59** | 0.00% | 41.77 | 2.32% |
| 9 | 0.2 | 1.00 | **11.00** | 7.57% | 3.60 | 8.56% | **4.08** | 0.00% | **2.88** | 0.57% | 55.837 | 0.00% | 3.58 | 0.00% | 42.98 | 2.15% |

Results for the "power then temperature" approach are shown in Table 6.5. The best layouts are achieved for $\alpha = 1.00$, i.e., our non-adaptive "by power" selection method as described in the previous section. The only exception is for the 9-chip layout with 0.2 overlap, for this using $\alpha = 0.75$ produces layout with slightly higher power on average, but at the cost of a poorer network topology. The conclusion is that our adaptive "power then temperature" approach is not useful in practice.

Results for the "network then temperature" adaptive selection option approach are given in Table 6.6. The average coefficient of variation, CV, for the edges, levels, diameter, ASPL, power, and temperature characteristics are 1.06%, 25.23%, 6.79%, 3.42%, 2.20%, 0.93%, and 2.50% respectively. In addition, the max CVs for edges, levels, diameter, ASPL, power and temperature are 3.59%, 31.72%, 15.81%, 7.56%, 13.61%, 5.78%, and 5.00% respectively.

The results for the "network then power" approach are shown in Table 6.6. Here again,

Table 6.6: "Network then temperature" results for different $\alpha$ values for 6- and 9-chip layouts with .2 overlap. Values are shown as averages over 10 trials. Best network and power metrics are shown in boldface.

| Layout Size (Chips) | Overlap | Alpha | Edges | CV | Levels | CV | Diameter | CV | ASPL | CV | Power (W) | CV | Frequency (GHz) | CV | Temperature (C) | CV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 0.2 | 0 | **5.50** | 0.00% | 2.80 | 23.81% | 3.70 | 11.40% | 1.973 | 5.05% | **56.7207** | 0.00% | **3.60** | 0.00% | **46.329** | 2.04% |
| 6 | 0.2 | 0.25 | 5.45 | 0.00% | 2.85 | 27.68% | 3.75 | 0.00% | 1.983 | 0.00% | 56.7207 | 0.00% | 3.60 | 0.00% | 46.626 | 2.45% |
| 6 | 0.2 | 0.5 | 5.33 | 0.00% | 2.70 | 29.22% | 3.83 | 0.00% | 2.016 | 0.00% | 56.7207 | 0.00% | 3.60 | 0.00% | 46.900 | 1.59% |
| 6 | 0.2 | 0.75 | 5.35 | 0.00% | 2.70 | 24.69% | 3.80 | 8.32% | 2.010 | 3.15% | 56.7207 | 0.00% | 3.60 | 0.00% | 46.560 | 2.27% |
| 6 | 0.2 | 1 | 5.48 | 0.00% | 2.64 | 19.56% | **3.64** | 0.00% | **1.955** | 0.00% | 55.8374 | 13.61% | 3.58 | 5.78% | 46.603 | 5.00% |
| 9 | 0.2 | 0 | 8.90 | 3.55% | 3.70 | 31.72% | 5.30 | 9.11% | 2.567 | 5.12% | 55.2485 | 0.00% | 3.56 | 0.00% | 48.117 | 1.83% |
| 9 | 0.2 | 0.25 | 8.80 | 3.59% | 3.40 | 30.38% | 5.25 | 9.84% | 2.564 | 5.54% | **55.9846** | 0.00% | **3.58** | 0.00% | 48.365 | 3.41% |
| 9 | 0.2 | 0.5 | 8.70 | 0.00% | 3.50 | 30.71% | 5.33 | 15.81% | 2.585 | 7.08% | 55.7393 | 8.35% | 3.57 | 3.54% | 48.296 | 2.45% |
| 9 | 0.2 | 0.75 | 8.68 | 0.00% | 3.50 | 20.20% | 5.28 | 13.40% | 2.574 | 7.56% | 55.2485 | 0.00% | 3.56 | 0.00% | 48.168 | 2.01% |
| 9 | 0.2 | 1 | **9.10** | 3.48% | 3.38 | 14.29% | **5.02** | 0.00% | **2.473** | 0.71% | 52.5987 | 0.00% | 3.49 | 0.00% | **47.771** | 1.94% |

using $\alpha = 1.00$, i.e., the non-adaptive "by network" approach in the previous section, produces good layouts in terms of network topology. Using $\alpha < 1.00$ in some cases can lead to improvements in some network topology characteristics (e.g., higher number of edges for 6-chip layouts) and/or power. However, these improvement are slight and no $\alpha$ value leads to layouts strictly superior to those obtained with $\alpha = 1.00$.

The above results are conclusive in showing that, in spite of being intuitively useful, an adaptive candidate selection approach does not produce better layouts in practice. In the end, the non-adaptive "by power" selection approach from the previous section leads to the best layouts (higher chip power and frequencies, denser network topologies).

### 6.4.3 Multi-Chip Additions

The strength of our heuristic is dependent on the number of random candidates it can choose from when adding each new chip to the layout. Unsurprisingly, we observe that running our heuristic with a high number of candidates yields more favorable layouts. Unfortunately, candidate evaluation is compute-intensive, which is partially relieved by our use of MPI (see Section 5.2.2). But candidate evaluation, i.e., Hotspot simulations, still accounts for the vast majority of our simulation runtime. As a result, while considering more candidates leads to better layouts it can also lead to prohibitively long layout construction times. While long layout construction times are perhaps acceptable if designing a single layout for production purposes, they preclude answering research/design questions that require large number of experiments. To remedy this problem we implement and investigate the effect of adding multiple chips at a time. As described in Section 5.3.1, the goal is to speed up layout construction with our heuristic by reducing the number of Hotspot invocations. We suspect an inverse relationship between the layout construction time and the layout quality using this

technique, but the question is whether this inverse relationship is worthwhile. In other words, as we speed up the runtime by adding more chips at a given time before calling Hotspot, we may decrease the quality of the produces layouts. This said, if we bound the layout construction time, an interesting question is whether it is better to do one-chip-at-a-time with fewer candidates or to do multiple-chips-at-a-time with more candidates.

The implementation of the multi-chip addition feature allows for the additions of any number of chips that is less than or equal to the total number of chips that needs to be added to the starting 3-chip cradle. We evaluate layouts containing 6, 9, and 13 chips, so that starting with a cradle requires the addition of a total of 3, 6, and 10 chips, respectively. When the number of chips being added at one time is not a multiple of the remaining number of chips to be added, the last step may add fewer chips. For example, if we want to add chips in multiples of two to build a 6-chip layout, we would start with a three chip cradle and have three chips remaining to add to the rest of the layout. Two is not a multiple of three and thus only one 2-chip addition can be performed with one chip remaining to be added. This remaining chip is added to the layout in a last step to complete the 6-chip layout. In the following experiments, we compare the effectiveness adding chips in multiples of 1 (our original heuristic), 2, and 3, as well as adding chips in cradle structures (a special case of adding chips in multiples of 3).

### Effect of Multi-Chip Addition on Layout Construction Time

To explore the effects of adding multiple chips at a time, we run the following experiments. Over 10 trials we construct layouts with 6 or 9 chips with 0.1 or 0.2 overlaps, while adding chips in multiples of 1, 2, 3, or cradle-by-cradle. We set the number of candidates to be evaluated in each step of the heuristic to twice the number of MPI threads, i.e., 14. All heuristic executions are on a 2.4GHz dual 4-core (with hyperthreading) machine with 23 GiB of RAM. Results are presented in Table 6.7. The average coefficient of variation, CV, for runtime, edges, levels, diameter, ASPL, power, and temperature characteristics are 19.85%, 6.77%, 7.96%, 3.52%, 3.59%, 2.19%, 0.83%, and 2.13% respectively. In addition, the max CVs for edges, levels, diameter, ASPL, power and temperature are 60.40%, 16.67%, 24.74%, 17.32%, 8.93%, 16.40%, 6.66%, and 4.05% respectively.

The results, in Table 6.7, confirm our expectation that adding more chips at a time leads to a speedup in layout construction time. 6-chip layouts with lower overlaps seem to experience larger speedups as more chips are added at once. For 9-chip layouts, the opposite is true and the denser of the layouts with a 0.2 overlap experiences higher speedup as more

Table 6.7: Results for adding chips in multiples of 3, 2, 1, and by cradles. Best network and power metrics are shown in boldface.

| Layout Size (Chips) | Overlap | Addition Method | Runtime (Seconds) | CV | Edges | CV | Levels | CV | Diameter | CV | ASPL | CV | Power (W) | CV | Frequency (GHz) | CV | Temperature (C) | CV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 0.1 | cradle | 115.25 | 10.85% | **6.67** | 8.66% | 2.67 | 21.65% | 3.33 | 17.32% | **1.7333** | 6.66% | **56.7207** | 0.00% | **3.60** | 0.00% | **43.1567** | 2.15% |
| 6 | 0.1 | 3 | **109.41** | 0.50% | 6.00 | 0.00% | 3.00 | 0.00% | **3.00** | 0.00% | 1.7333 | 0.00% | 56.7207 | 0.00% | 3.60 | 0.00% | 45.4867 | 0.62% |
| 6 | 0.1 | 2 | 223.36 | 16.09% | 6.67 | 8.66% | 3.00 | 0.00% | 3.00 | 0.00% | 1.6889 | 2.28% | 56.7207 | 0.00% | 3.60 | 0.00% | 43.7067 | 3.37% |
| 6 | 0.1 | 1 | 359.68 | 4.10% | 6.00 | 0.00% | 2.00 | 0.00% | 3.00 | 0.00% | 1.7333 | 0.00% | 56.7207 | 0.00% | 3.60 | 0.00% | 45.6900 | 2.54% |
| 6 | 0.2 | cradle | **102.57** | 60.40% | **7.00** | 5.09% | 3.00 | 0.00% | **3.00** | 12.37% | **1.6667** | 5.43% | **56.7207** | 0.00% | **3.60** | 0.00% | 46.7700 | 0.92% |
| 6 | 0.2 | 3 | 132.31 | 16.35% | 6.00 | 5.09% | 2.33 | 0.00% | 3.00 | 0.00% | 1.8000 | 2.70% | 56.7207 | 0.00% | 3.60 | 0.00% | **46.7467** | 2.65% |
| 6 | 0.2 | 2 | 276.85 | 34.54% | 6.00 | 14.32% | 2.33 | 0.00% | 3.00 | 0.00% | 1.7333 | 8.39% | 56.7207 | 0.00% | 3.60 | 0.00% | 48.4733 | 1.50% |
| 6 | 0.2 | 1 | 381.10 | 34.34% | 6.33 | 5.41% | 2.33 | 0.00% | 3.00 | 0.00% | 1.7111 | 3.98% | 56.7207 | 0.00% | 3.60 | 0.00% | 47.0967 | 2.34% |
| 9 | 0.2 | cradle | **604.36** | 16.16% | **10.67** | 14.32% | 3.67 | 15.75% | 4.33 | 13.32% | 2.1852 | 8.93% | 56.7207 | 16.40% | **3.60** | 0.00% | 48.5300 | 2.38% |
| 9 | 0.2 | 3 | 996.35 | 25.76% | 10.33 | 5.59% | 3.00 | 0.00% | **4.00** | 0.00% | **2.1296** | 1.51% | 41.9991 | 0.00% | 3.20 | 0.00% | **45.8667** | 2.51% |
| 9 | 0.2 | 2 | 1202.51 | 20.30% | 10.00 | 10.00% | 3.00 | 0.00% | 4.00 | 0.00% | 2.1852 | 6.40% | 51.8135 | 0.00% | 3.47 | 6.66% | 48.5833 | 2.81% |
| 9 | 0.2 | 1 | 1893.61 | 5.79% | 10.67 | 5.41% | 3.67 | 15.75% | 4.33 | 13.32% | 2.1296 | 1.51% | 51.8135 | 16.40% | 3.47 | 6.66% | 47.7333 | 4.05% |
| 9 | 0.1 | cradle | **371.72** | 21.74% | **11.33** | 0.00% | 3.00 | 0.00% | 4.67 | 0.00% | 2.1296 | 0.00% | **56.7207** | 0.00% | **3.60** | 0.00% | **46.4733** | 0.00% |
| 9 | 0.1 | 3 | 531.50 | 32.00% | 11.33 | 16.67% | 3.00 | 24.74% | **4.00** | 0.00% | 2.0556 | 7.41% | 56.7207 | 0.00% | 3.60 | 0.00% | 47.9000 | 2.20% |
| 9 | 0.1 | 2 | 825.95 | 13.05% | 10.67 | 0.00% | 3.00 | 24.74% | 4.00 | 0.00% | 2.1296 | 0.00% | 56.7207 | 0.00% | 3.60 | 0.00% | 49.1233 | 1.91% |
| 9 | 0.1 | 1 | 1000.42 | 5.59% | 10.67 | 9.12% | 3.00 | 24.74% | 4.00 | 0.00% | 2.1296 | 2.25% | 56.7207 | 0.00% | 3.60 | 0.00% | 48.0467 | 2.19% |

chips are added at one time. With the exception of the 6-chip layout with a 0.1 overlap, a continuous increase in speedup is observed as more chips are added at a time, as well as when going from adding in multiple of 3 to adding cradle-by-cradle.

Although we would expect some slight differences in runtime between adding in multiples of 3 and adding cradle-by-cradle, the results show consistently (with the exception of the 6-chip layout with a 0.1 overlap) that adding chips cradle-by-cradle is significantly faster than adding chips in multiples of 3. To investigate the reason for this behavior, we profiled the execution in each run case using Python's cProfile built-in tool. Profiling results indicate that the number of random searches for feasible chip positions, and accompanying checks for feasibility, are reduced in the case of the cradle-by-cradle approach. This is expected as this approach is more constrained in terms of where chips that be placed. However, the reduction in the number of random searches does not have a significant effect on the overall runtime. It turns out that the speedup seen when adding cradle-by-cradle is due to faster Hotspot executions. According to the profiler, on average, the simulation spends less time in the Hotspot function (by 52.99%) when adding cradle-by-cradle than when adding by multiples of 3. While we have no definite explanation why Hotspot simulations are faster in this case, we suspect it may have to do the fact that adding cradle substructures results in less dense layouts, which leads to easier and faster Hotspot calculations, likely due to the adaptive grid numerical methods used in Hotspot. This is corroborated by the results in Table 6.7 that show that increasing the overlap (from 0.1 to 0.2) leads to longer runtimes, which is due to longer Hotspot calculations. Further investigation into the inner workings of Hotspot to precisely explain and quantify this behavior is beyond the scope of this thesis.

As discussed above, Table 6.7 shows that adding more chips at a time reduces layout construction time, which was expected. This table also shows the network and power

characteristics of produced layouts. Some layout metrics seem to benefit from the addition of more than one chip at a time. However, although we include these results here for completeness, it is difficult to make a fair comparison between the different approaches due to the confounding factors regarding the differences in runtimes. To resolve this difficulty, in the next section we re-run these experiments but impose a maximum layout construction time budget for all the approaches.

### Layout Construction under a Time Budget

The preliminary results in Table 6.7 reveal that the runtime is influenced by the number of chips in the layout, the overlap, the chip addition strategy, and the number of candidates to evaluate at each step. In this section we set a 5-minute time budget for 6-chip layout construction, and a 20-minute time budget for 9-chip layout construction. To enforce the above time budgets, we empirically determine the number of candidates to use for each layout construction (Recall that candidate evaluation virtually account for the entirety of layout construction time). From each experiment in the previous section, we divide the number of candidates by the total execution time and then multiply by the our execution time budget. This gives an empirical number of candidates that should match the execution time budget. This number, however, is only an estimate and we refine it by running our experiment using a range of numbers of candidates in a neighborhood of that estimate. In our results, all layout constructions have runtimes that are under our time budgets by a few seconds. Results are shown in Table 6.8. The average coefficient of variation, CV, for the edges, levels, diameter, ASPL, power, and temperature characteristics are 9.55%, 4.16%, 7.75%, 3.81%, 2.47%, 2.03%, and 4.04% respectively. In addition, the max CVs for edges, levels, diameter, ASPL, power and temperature are 17.48%, 8.42%, 17.26%, 8.04%, 14.85%, 11.57%, and 9.12% respectively.

Table 6.8 shows results for layout construction with time budgets. This table shows numbers of candidates used at each step for all layout construction options. For all layout sizes and overlap values, the cradle-by-cradle approach can evaluate more candidates in a given amount of time that the other approaches. This is due to the fact that Hotspot simulations are faster with this approach. The other approaches tend to use roughly similar numbers of candidates. For instance, for a 9-chip layout with 0.2 overlap, the cradle-by-cradle approach can evaluate 60 candidates in total, while the other approaches can evaluate between 44 an 48 candidates.

The broad observation from Table 6.8 is that adding chips one by one is not the best

Table 6.8: Multi-Chip addition methods under time constraints (300 seconds for 6 chip layouts and 1200 seconds for 9 chip layouts). Best network and power metrics are shown in boldface.

| Runtime Limit (Seconds) | Layout Size (Chips) | Overlap | Addition Method | Candidates Evaluated per Round | Round of Evaluation | Total Candidate Evaluations | Edges | CV | Levels | CV | Diameter | CV | ASPL | CV | Power (W) | CV | Frequency (GHz) | CV | Temperature (C) | CV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 300 | 6 | 0.1 | cradle | 45 | 1 | 45 | **7.00** | 3.25% | 3.00 | 5.83% | **3.00** | 11.03% | **1.67** | 7.61% | **56.721** | 0.00% | **3.60** | 0.00% | 43.245 | 8.74% |
| 300 | 6 | 0.1 | 3 | 40 | 1 | 40 | 6.46 | 14.01% | 2.64 | 3.37% | 3.00 | 14.70% | 1.71 | 6.55% | 56.721 | 0.00% | 3.60 | 0.00% | 44.453 | 2.28% |
| 300 | 6 | 0.1 | 2 | 19 | 2 | 38 | 6.10 | 5.01% | 2.67 | 8.42% | 3.00 | 15.16% | 1.73 | 0.00% | 56.721 | 0.00% | 3.60 | 0.00% | 45.075 | 4.58% |
| 300 | 6 | 0.1 | 1 | 14 | 3 | 42 | 6.04 | 12.41% | 2.60 | 3.95% | 3.00 | 0.00% | 1.73 | 0.00% | 56.721 | 0.00% | 3.60 | 0.00% | 45.241 | 7.15% |
| 300 | 6 | 0.2 | cradle | 45 | 1 | 45 | **7.00** | 15.27% | 3.00 | 5.29% | **3.00** | 12.00% | **1.67** | 0.00% | **56.721** | 0.00% | **3.60** | 0.00% | 46.030 | 1.40% |
| 300 | 6 | 0.2 | 3 | 34 | 1 | 34 | 6.36 | 16.18% | 2.74 | 0.00% | 3.04 | 0.00% | 1.73 | 7.66% | 56.721 | 0.00% | 3.60 | 0.00% | 47.062 | 0.72% |
| 300 | 6 | 0.2 | 2 | 17 | 2 | 34 | 6.20 | 0.13% | 2.74 | 7.96% | 3.00 | 5.83% | 1.73 | 6.32% | 56.721 | 0.00% | 3.60 | 0.00% | 47.418 | 4.04% |
| 300 | 6 | 0.2 | 1 | 11 | 3 | 33 | 6.03 | 7.89% | 2.80 | 5.56% | 3.00 | 5.22% | 1.74 | 5.20% | 56.721 | 0.00% | 3.60 | 0.00% | 47.620 | 0.00% |
| 1200 | 9 | 0.1 | cradle | 65 | 2 | 130 | **12.00** | 9.71% | 3.23 | 3.89% | **4.00** | 8.70% | **2.00** | 0.00% | **56.721** | 0.00% | **3.60** | 0.00% | 47.653 | 2.12% |
| 1200 | 9 | 0.1 | 3 | 45 | 2 | 90 | 11.28 | 4.39% | 3.08 | 4.19% | 4.00 | 8.52% | 2.05 | 6.32% | 56.721 | 0.00% | 3.60 | 0.00% | 48.451 | 0.00% |
| 1200 | 9 | 0.1 | 2 | 30 | 3 | 90 | 11.34 | 16.50% | 2.96 | 0.00% | 4.04 | 0.00% | 2.05 | 0.00% | 56.721 | 0.00% | 3.60 | 0.00% | 48.651 | 3.25% |
| 1200 | 9 | 0.1 | 1 | 14 | 6 | 84 | 11.06 | 17.48% | 3.08 | 2.41% | 4.02 | 7.42% | 2.06 | 0.00% | 56.426 | 0.00% | 3,59 | 3.30% | 48.149 | 6.89% |
| 1200 | 9 | 0.2 | cradle | 30 | 2 | 60 | **11.71** | 9.81% | 3.48 | 3.91% | **4.25** | 5.21% | **2.06** | 7.12% | **56.414** | 9.04% | **3.59** | 7.98% | 48.687 | 1.88% |
| 1200 | 9 | 0.2 | 3 | 22 | 2 | 44 | 10.20 | 4.76% | 3.20 | 3.45% | 4.40 | 17.26% | 2.22 | 6.09% | 52.304 | 11.35% | 3.48 | 1.40% | 47.951 | 4.04% |
| 1200 | 9 | 0.2 | 2 | 16 | 3 | 48 | 10.34 | 9.82% | 3.26 | 0.00% | 4.34 | 12.89% | 2.17 | 8.04% | 50.832 | 14.85% | 3.44 | 8.22% | 47.618 | 9.12% |
| 1200 | 9 | 0.2 | 1 | 8 | 6 | 48 | 10.55 | 6.26% | 3.30 | 8.28% | 4.25 | 0.00% | 2.14 | 0.00% | 52.304 | 4.34% | 3.48 | 11.57% | 47.902 | 8.46% |

approach. In fact, the cradle-by-cradle approach seems to be best. Results for 6-chips layout show that for all approaches chips can be operated at their maximum frequencies. This is because these layouts do not run the risk of exceeding the temperature threshold. However, in terms of the network, the cradle-by-cradle approach leads to the best network (better number of edges, better ASPL, equivalent diameter). Additionally, we note that the cradle-by-cradle approach produces the coolest layouts when compared to the other approaches. Results for 9-chip layouts show that it is not always possible to operate chips at their maximum frequencies. Nevertheless, the cradle-by-cradle approach allows the highest chip frequencies in addition to, as for 6-chip layouts, producing the better networks. Recall that the results in Table 6.8 are averages over 10 trials. It turns out that if we pick the best layout for each approach over its 10 trials, then the cradle-by-cradle approach looks even more competitive.

Although it is difficult to precisely understand why the cradle-by-cradle approach performs well, we can venture a couple of guesses. First, due to faster Hotspot simulations, it can do a broader random exploration of the design space by considering more candidates. Second, the cradle is a preset substructure that is 2-level high with a diameter of 2, that will cause

generated layouts to be relatively sparser due to geometric constraints. It thus seems that adding chips in a cradle-by-cradle fashion strikes a good balance between network topology density and heat dissipation.

Given the results in Table 6.8, we use the cradle-by-cradle approach for the rest of the experiments in this chapter. In the next section, we also consider 13-chip layouts, which we construct by first adding 3 cradles to the initial cradle, and then a single chip.

## 6.5 Main Results

The results in the previous section allow us to fine tune our heuristic. We now know which configuration parameters lead our heuristic to produce better layouts in practice. More specifically, our greedy candidate selection criteria is power, we do not employ an adaptive candidate selection approach, and we use the cradle-by-cradle addition method. We run our heuristic to build 6-, 9-, and 13-chip layouts with both 0.1 and 0.2 overlaps. Our goal is to compare the produced layouts to the baseline checkerboard layout. Results in the previous section are accompanied by CVs. Some metrics reported high CV in some experiments which indicates high diversity and suggests a possible benefit to running more trials. However, we are only interested in having at least one heuristic generated layout to beat the baseline checkerboard, for each layout size and overlap. Thusly, to obtain our final results we run 10 trials, for each layout size and overlap, and pick, from the 10 trials, the layout with the best characteristics. The high CVs reported in the previous section indicate that even better results are likely using more trials, but we will see that even using only 10 trials leads to layouts that outperform the baseline checkerboard layout significantly. The final layouts are depicted in Appendix A and their characteristics are given in Table 6.9, along with those of the baseline checkerboard layout.

The broad observation from Table 6.9 is that our heuristic produces layouts that outperform the checkerboard layout (in terms of network, in terms of power, and/or in terms of both network and power) in all our experimental scenarios. In what follows, we discuss these results in more details for each layout size.

### 6.5.1 6-Chip Layouts

The 6-chip layouts, whether checkerboard or heuristic-generated, are not at risk of exceeding the temperature threshold. As a result, in all these layouts, chips can operate at they maximum frequency (3600 GHz) for both 0.1 and 0.2 overlap values. The inter-chip network

Table 6.9: Comparison of best heuristic-generated layouts vs checkerboard. Best network and power metrics are shown in boldface.

| **Layout Description** | | | **Network Properties** | | | | **Computational Power** | | **Temperature** |
|---|---|---|---|---|---|---|---|---|---|
| overlap | layout size | generation method | edges | levels | diameter | ASPL | power (W) | frequency (GHz) | (°C) |
| 0.2 | 13 | Heuristic | **16** | 4 | **4** | **2.46** | **41.9991** | **3.20** | 48.80 |
| | | Checkerboard | 16 | 2 | 4 | 2.46 | 25.7721 | 2.40 | 47.48 |
| | 9 | Heuristic | **12** | 3 | **4** | **2.00** | **56.7207** | **3.60** | 49.02 |
| | | Checkerboard | 12 | 2 | 4 | 2.00 | 41.9991 | 3.20 | 48.40 |
| | 6 | Heuristic | **7** | 3 | **3** | **1.67** | **56.7207** | **3.60** | 45.29 |
| | | Checkerboard | 6 | 2 | 3 | 1.73 | 56.7207 | 3.60 | 48.44 |
| 0.1 | 13 | Heuristic | **16** | 3 | **4** | **2.46** | **41.9991** | **3.20** | 47.37 |
| | | Checkerboard | 16 | 2 | 4 | 2.46 | 33.0215 | 2.80 | 47.91 |
| | 9 | Heuristic | **12** | 3 | **4** | **2.00** | **56.7207** | **3.60** | 48.99 |
| | | Checkerboard | 12 | 2 | 4 | 2.00 | 41.9991 | 3.20 | 44.52 |
| | 6 | Heuristic | **7** | 3 | **3** | **1.67** | **56.7207** | **3.60** | 42.33 |
| | | Checkerboard | 6 | 2 | 3 | 1.73 | 56.7207 | 3.60 | 45.02 |

topology characteristics differ between the checkerboard and the heuristic-generated layouts. More specifically, heuristic-generated layouts have more edges and a smaller ASPL than checkerboard layouts, indicating that they have better connected network topologies. Therefore, heuristic-generated layouts should lead to higher parallel application performance. Note that both checkerboard and heuristic-generated layouts have the same diameter, but the heuristic-generated layouts use more levels than the checkerboard (which always uses 2 levels). The number of levels relates to the diameter of the overall layout in that it is a lower bound of it. Our heuristic is able to use more than 2 levels to obtain a denser network topology but still with good heat dissipation properties. We conclude that our heuristic produces layouts that are strictly preferable to checkerboard layouts (similar aggregate computational power, better network topology).

### 6.5.2 9-Chip Layouts

As expected, 9-chip layouts generate more heat than 6-chip layouts. Table 6.9 indicates that a 9-chip checkerboard layout, either with a 0.1 or a 0.2 overlap, can only operate chips at 3200 GHz, which is two DVFS levels lower than the maximum 3600 GHz frequency. By contrast, the heuristic-generated 9-chip layout with a 0.2 overlap is able to operate chips at their maximum frequency while remaining below the temperature threshold. The checkerboard and heuristic-generate layouts have identical network characteristics (except for the number of levels, as explained for 6-chip layouts in the previous section), with number of edges, diameter, and ASPL at 12, 4, and 2.00 respectively. The additional level used in the heuristic-generated layout allows for a greater surface area for better heat dissipation,

which in turns makes it possible to operate the chips at a higher frequency. Here again, we conclude that our heuristic produces layouts that are strictly preferable to checkerboard layouts (similar network topology, better aggregate computational power).

### 6.5.3 13-Chip Layouts

With more chips, heat dissipation becomes more of an issue. We find that in our experiments no layout can operate their chips at their maximum frequencies. As seen in Table 6.9, the 13-chip checkerboard layout with a 0.1 overlap is severely affected by heat dissipation and can only operate chips at 2800 GHz, which is 4 DVFS levels below the maximum 3600 GHz frequency. The heuristic-generated layout is less affected by heat dissipation and can operate chips at 3200 GHz, which is 2 DVFS levels below the maximum frequency. Similar to 9-chip layout results in the previous section, both layouts have the same network topology characteristics and the heuristic-generated layout uses one more levels.

Going to 0.2 overlap, which decreases heat dissipation, we find that the checkerboard layout can now only operate chips at 2400 GHz, which is 6 DVFS levels below the maximum frequency. By contrast, even with this larger overlap, the heuristic-generated layout can still operate chips at 3200 GHz. Similar that results with 0.1 overlap, both layouts have the same network topology characteristics, but the heuristic-generated layout uses two more levels. As in the previous section, we conclude that our heuristic produces layouts that are strictly preferable to checkerboard layouts (similar network topology, better aggregate computational power).

### 6.5.4 Summary of Results

Our results show that layouts generated by our heuristic strictly outperform checkerboard layouts when considering both aggregate computational power and network topology. Our heuristic can produce layouts with more than 2 levels so as to increase heat dissipation while producing a network topology that is better connected than that of the checkerboard layout. Compared to the checkerboard layout, heuristic-generated layouts are better at mitigating heat build up as the number of chips increases. However, as the number of chips increases, the network topologies have characteristics that become similar to that of the checkerboard layout. Regardless, we conclude that our heuristic produces layouts that afford higher parallel application performance than the previously proposed checkerboard layout.

# CHAPTER 7
# CONCLUSION

Integrated 3D multi-chip systems consist of multiple (multi-core) microprocessor chips interconnected via some network topology. High amounts of aggregated compute power can then be delivered to parallel applications at low cost and by a single system. Higher compute power can be achieved with larger numbers of chips in the system and/or running chips at high frequencies in the system, but heat dissipation then becomes a problematic issue. As explained in Section 1.2, there is in fact a complex trade-off between temperature, chip frequencies, inter-chip network topology, and physical layout for these systems. The broad question we have studied in this work is that of constructing physical layouts for these systems that achieve a desirable trade-off, or at least, a trade-off strictly superior to that achieved be previously proposed layout options (namely, the stack layout and the checkerboard layout). We assume the use of inductive coupling, as possible with the ThruChip Interface (TCI), which interconnects overlapping chips wireless, thus making it possible to explore a wide range of options for constructing desirable physical layouts for integrated 3D multi-chip systems.

## 7.1 Summary of Contributions and Findings

We have developed a heuristic to construct multi-chip physical layouts given a temperature threshold. The broad objective is to produce good solutions to the multi-objective optimization layout construction problem, or at least to outperform the previously proposed baseline checkerboard layouts in terms of chip operating frequencies and inter-chip network topology. Our heuristic is greedy and performs at each step an exhaustive search over a random sample of candidate chip position options. In this manner it can construct a layout with any specified number of chips.

Our heuristic is implemented in Python, and integrated with a simulation framework that computes layout temperature. This computation, which is necessary to evaluate each candidate chip position, relies on invoking the Hotspot simulator. Unfortunately, Hotspot is compute-intensive and thus limits the number of randomly sampled chip position candidates that can be searched. We have tackled this problem in two ways. First, we have used MPI to parallelize the exhaustive search in an attempt to alleviate the Hotspot bottleneck. While this is moderately effective, scalability is limited on a single system due to cache competition

among MPI processes. Second, we have experimented with adding multiple chips at each step of the heuristic. We have found that adding chips 3 by 3 (using our "cradle" substructure) decreases the layout construction time by roughly a factor 3 without significantly decreasing the qualify of the produced layout.

Our main finding is that our heuristic produces layouts that strictly outperform the previously proposed checkerboard layout in all our considered experimental scenarios with 6, 9, and 13 chips. For 6-chip layouts, our heuristic produces layouts with the same aggregate compute power than the checkerboard layout but with a better connected inter-chip network topology. For 9- and 13-chip layouts, our heuristic produces layouts with equivalent network topologies as that in the checkerboard layout, but with significantly higher aggregate compute power. This is because our heuristic benefits from exploring the layout design space, albeit using merely a greedy search over random samples in that space. Our results also indicate that our heuristic-generated layouts scale better than checkerboard layouts in terms of heat dissipation as the number of chips increases. Ultimately, our heuristic-generated layout deliver more aggregate performance to parallel applications than state-of-the-art checkerboard layouts.

## 7.2 Future Work

In this thesis we have assumed the standard air cooling approach. However, different coolant media are available (mineral oil, fluorinert, or even water). It would be interesting to see how our heuristic-generated layouts compare to checkerboard layouts with different cooling options. We would expect that with better cooling, our heuristic would produce denser layouts with network topology characteristics vastly superior to that of the checkerboard layout even with chips operating at high or their maximum frequencies.

In this thesis, we have enforce homogeneous power and frequency across all chips within our integrated 3D multi-chip system. Homogeneous frequencies provide some advantages in the context of running parallel applications (e.g., straightforward load-balancing and synchronization). However, allowing for heterogeneous frequencies could enable higher aggregate power for the entire system. For instance, given a particular layout, areas of high chip density can operate at a lower frequency to minimize heat build up. Conversely, the chip frequencies can be ramped up and in less dense or sparser areas of the layout. Chip frequencies are independent of each other and allowing heterogeneous frequencies thus widens the layout design space.

The type of constrained multi-objective optimization problem that we have tackled in this work with our greedy heuristic is prime candidate for using meta-heuristics. We would expect meta-heuristics to produce better layouts than our greedy heuristic. Unfortunately, typical meta-heuristics would require large numbers of Hotspot invocations, which is not feasible without large computational expenses (e.g., running our layout construction heuristic using MPI over a large-scale HPC cluster). This problem could be addressed in two ways. First, the Hotspot code could be optimized/parallelized. Second, a different simulator could be used that uses different and faster numerical methods, or uses approximate methods that can provide temperature estimates quickly (which would be particularly useful at the beginning of the layout construction process when temperature is not the most pressing concern).

While more sophisticated (meta-)heuristics should be considered, perhaps counter-intuitively some of our results suggests that a simpler heuristic than the one we have developed in this work could be effective. Recall that results in Table 6.9 show that our heuristic-generated layouts have identical network characteristics as that of the checkerboard layout in the 9- and 13-chip cases. Upon looking further at the geometries of the layouts generated by our heuristic, it appears that there is a convergence towards a checkerboard-like configuration when "viewed from above." This convergence is most noticeable in Figures A.5, A.7, A.9, and A.11. In these layouts, the networks are identical to that of the checkerboard layout, but the chip powers can be set higher due to better heat dissipation. This is because the checkerboard layout is constrained to two levels while the heuristic-generated layouts are not. These results show that the number of levels plays a significant role in heat dissipation. Therefore, if this same trend persists at higher scales, an effective and simpler heuristic could be to first generate a checkerboard configuration and then tune the chip levels as to optimizes heat dissipation. Such a heuristic may even execute faster than the heuristic proposed in this thesis, while possibly producing better layouts.

Our heuristic produces competitive layouts for integrated 3D multi-chip systems. Inductive coupling is not only limited to facilitating communication between microprocessor chips but in fact enables 3-D integration of a range of products with different power profiles, e.g., various off-the-shelf GPU, DRAM, NAND and processor chips [21]. It should be possible to apply our heuristic (perhaps with some modifications) to this entire range of systems.

# APPENDIX A
# HEURISTIC GENERATED LAYOUTS

Figures A.1-A.12 show heuristic-generated layouts corresponding to the results in Table 6.9. In these figures each chip is depicted in gray, with a lighter color as the chip level increases (chips on level 1 are black). Red cubes indicate the induction zones, i.e., TCI inter-connect areas. Note that for visual clarity figures are not to scale: the vertical axis is stretched (chips in neighboring levels are closer than depicted).

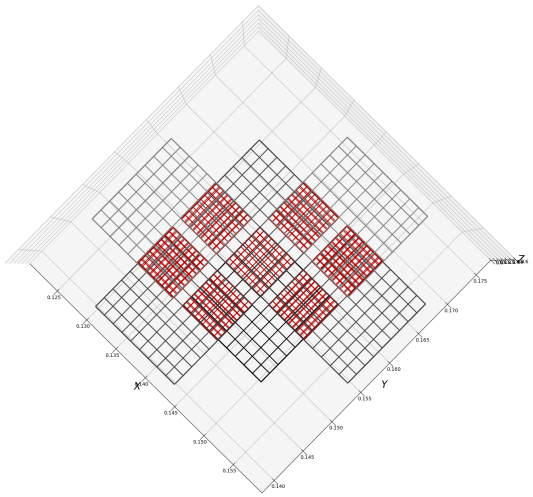



Figure A.1: 6 Chip, 0.2 Overlap, Top View.

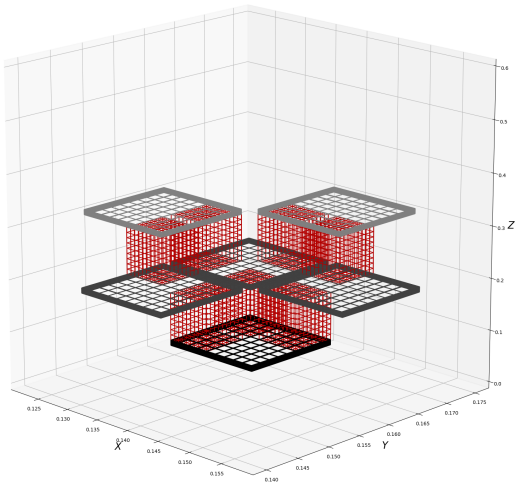



Figure A.2: 6 Chip, 0.2 Overlap, Side View.

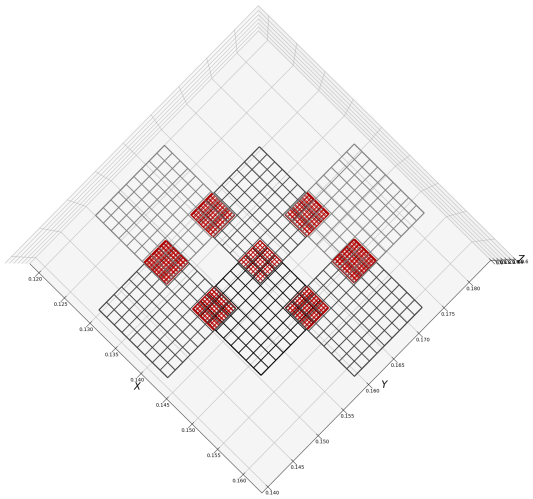



Figure A.3: 6 Chip, 0.1 Overlap, Top View.

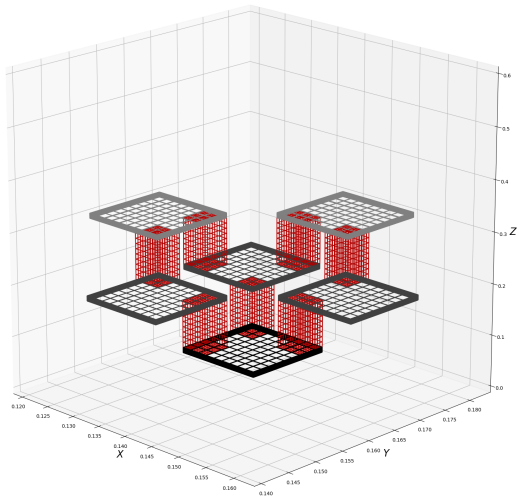



Figure A.4: 6 Chip, 0.1 Overlap, Side View.

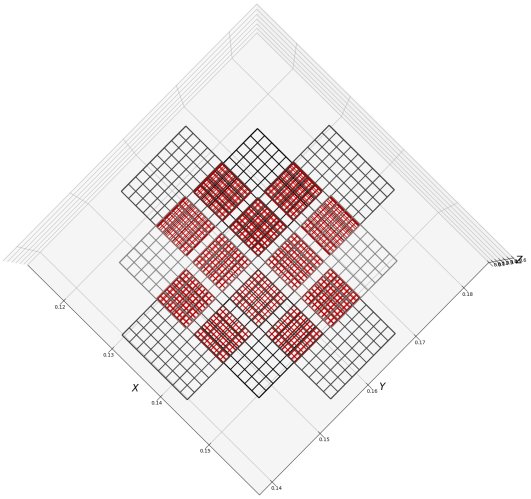



Figure A.5: 9 Chip, 0.2 Overlap, Top View.

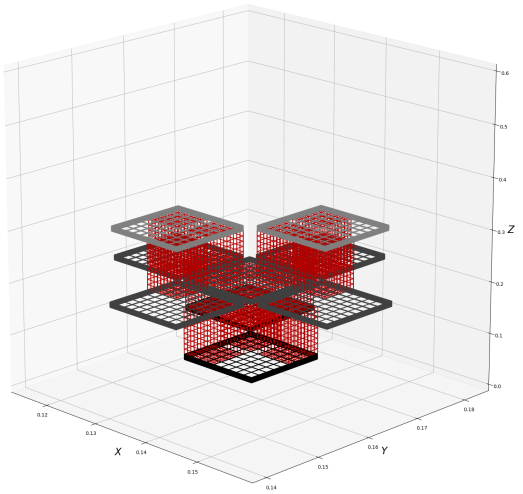



Figure A.6: 9 Chip, 0.2 Overlap, Side View.

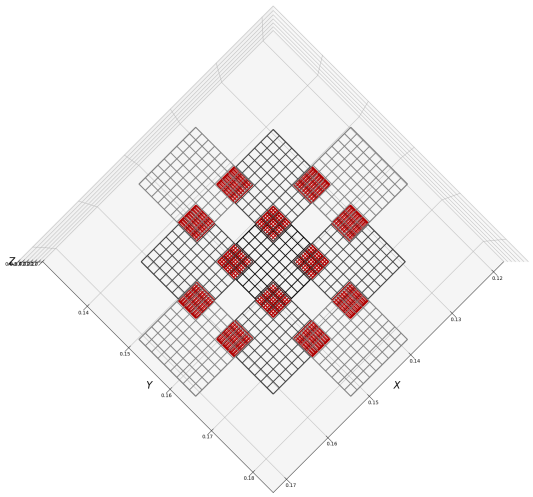



Figure A.7: 9 Chip, 0.1 Overlap, Top View.

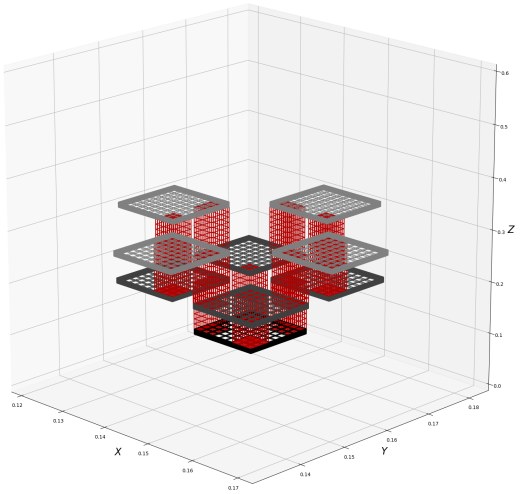



Figure A.8: 9 Chip, 0.1 Overlap, Side View.

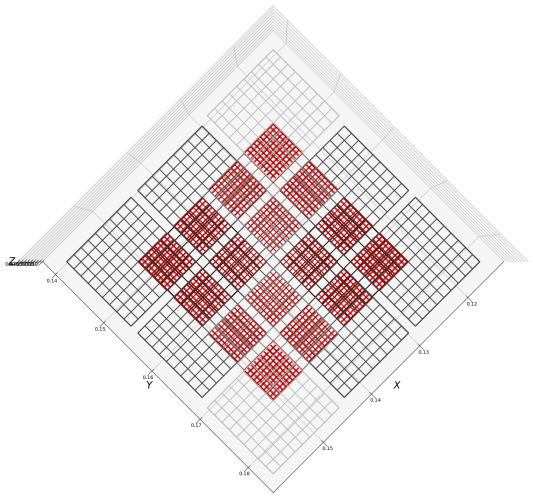



Figure A.9: 13 Chip, 0.2 Overlap, Top View.

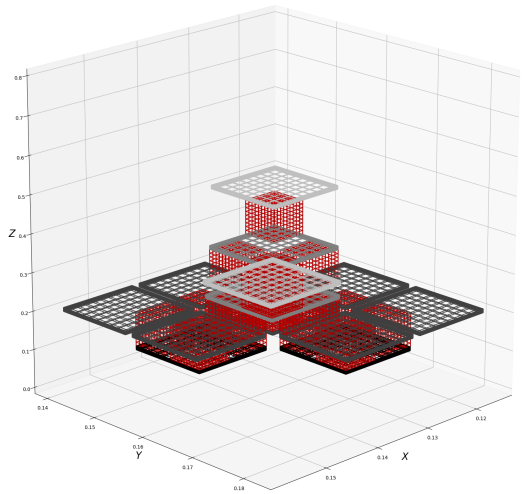



Figure A.10: 13 Chip, 0.2 Overlap, Side View.

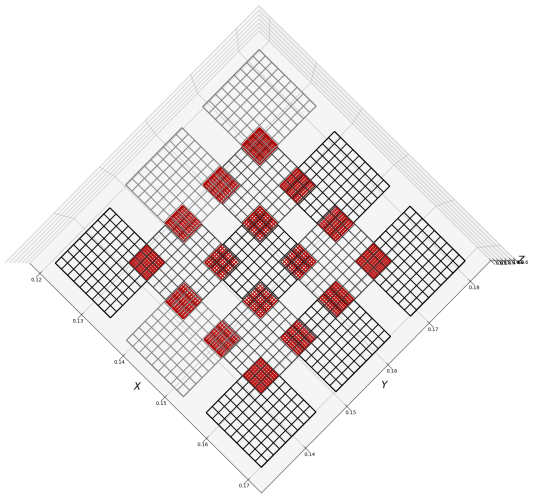



Figure A.11: 13 Chip, 0.1 Overlap, Top View.

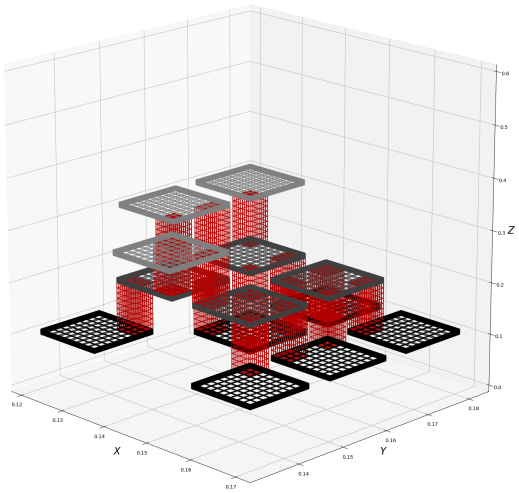



Figure A.12: 13 Chip, 0.1 Overlap, Side View.

# BIBLIOGRAPHY


[1] KISCO Ltd., diX Coating. `http://www.kisco-net.com/business/dix-coating/`.

[2] R. Zhang and M. R. Stan and K. Skadron. HotSpot 6.0: Validation,Acceleration and Extension. University o f Virginia, Tech. Report CS-2015-04.

[3] H. Nakahara, T. Ozaki, H. Matsutani, M. Koibuchi, and H. Amano. Novel chip stacking methods to extend both horizontally and vertically for many-core architectures with throuchip interface. *IEICE Transactions on Information and Systems*, E99.D(12):2871–2880, 2016.

[4] N. Miura, K. Kasuga, M. Saito, and T. Kuroda. An 8tb/s 1pj/b 0.8mm2/tb/s qdr inductive-coupling interface between 65nm cmos gpu and 0.1$\mu$m dram. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 436–437, Feb 2010.

[5] N. Miura, H. Ishikuro, T. Sakurai, and T. Kuroda. A 0.14pj/b inductive-coupling inter-chip data transceiver with digitally-controlled precise pulse shaping. In *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pages 358–608, Feb 2007.

[6] Mitsuko Saito, Noriyuki Miura, and Tadahiro Kuroda. A 2gb/s 1.8pj/b/chip inductive-coupling through-chip bus for 128-die nand-flash memory stacking. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 440–441, February 2010.

[7] Noriyuki Miura, Yusuke Koizumi, Yasuhiro Take, Hiroki Matsutani, Tadahiro Kuroda, Hideharu Amano, Ryuichi Sakamoto, Mitaro Namiki, Kimiyoshi Usami, Masaaki Kondo, and Hiroshi Nakamura. A scalable 3d heterogeneous multicore with an inductive thruchip interface. *IEEE Micro*, 33(6):6–15, 2013.

[8] M. Saito. Plan to develop ExaScaler computing system with proprietary Processor, DRAM and Cooling technology. In *International Workshop on A Strategic Initiative of Computing: Systems and Applications (SISA): Integrating HPC, Big Data, AI and Beyond*, 2017.

[9] Sunao Torii and Hitoshi Ishikawa. ZettaScaler: Liquid immersion cooling Manycore based Supercomputer. In *International Symposium on Computing and Networking (CANDAR)*, pages 10–14, November 2016.

[10] Top 500 Supercomputer Sites. `http://www.top500.org/`.

[11] K. Niitsu, Y. Kohama, Y. Sugimori, K. Osada, N. Irie, H. Ishikuro, and T. Kuroda. Misalignment Tolerance in Inductive-Coupling Inter-Chip Link for 3D System Integration. In *International Conference on Solid-State Devices and Materials*, pages 86–87, September 2008.

[12] K. Niitsu, Y. Kohama, Y. Sugimori, K. Kasuga, K. Osada, N. Irie, H.Ishikuro, and T. Kuroda. Modeling and Experimental Verification of Misalignment Tolerance in Inductive-Coupling Inter-Chip Link for Low-Power 3-D System Integration. *IEEE Trans. VLSI Syst.*, 18(8):1238–1243, 2010.

[13] S. Gopal, S. Das, D. Heo, and P. P. Pande. Energy and Area Efficient Near Field Inductive Coupling: A Case Study on 3D NoC. In *International Symposium on Networks-on-Chip (NOCS)* , pages 5:1–5:8, October 2017.

[14] T. Kagami, H. Matsutani, M. Koibuchi, Y. Take, T. Kuroda, and H. Amano. Efficient 3-D Bus Architectures for Inductive-Coupling ThruChip Interfaces. *IEEE Trans. VLSI Syst.*, 24(2):493–506, 2016.

[15] N. Miura, T. Sakurai, and T. Kuroda. Crosstalk Countermeasures for High-Density Inductive-Coupling Channel Array. *IEEE Journal of Solid-State Circuits (JSSC)*, 42(2):410–421, 2007.

[16] Y. Sugimori, Y. Kohama, M. Saito, Y. Yoshida, N. Miura, H. Ishikuro, T. Sakurai, and T. Kuroda. A 2Gb/s 15pJ/b/chip Inductive-Coupling programmable bus for NAND Flash memory stacking. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 244–245, 2009.

[17] Guido van Rossum and Python Development Team. *Python 2.7.10 Language Reference*. Samurai Media Limited, United Kingdom, 2015.

[18] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B . Brockman, Dean M. Tullsen, and Norman P. Jouppi. Mcpat: An integrated power, area, and timing modeling framework for m ulticore and manycore architectures. In *the IEEE/ACM International Symposium on Microarchitecture*, pages 469–480, 2009.

[19] B. Roy. Problems and methods with multiple objective functions. *Math. Program.*, 1(1):239–266, December 1971.

[20] Anne Auger. Convergence results for the (1,)-sa-es using the theory of -irreducible markov chains. *Theoretical Computer Science*, 334(1):35 – 69, 2005.

[21] H. Matsutani, M. Koibuchi, I. Fujiwara, T. Kagami, Y. Take, T. Kuroda, P. Bogdan, R. Marculescu, and H. Amano. Low-latency wireless 3D NoCs via randomized shortcut chips. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, March 2014.