ABSTRACT

Title of dissertation: THERMAL, POWER DELIVERY AND RELIABILITY MANAGEMENT FOR 3D ICS

Zhiyuan Yang
Doctor of Philosophy, 2018

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

Three-dimensional (3D) integration technology is promising to continuously improve the performance of electronic devices by vertically stacking multiple active layers and connecting them with Through-Silicon-Vias (TSVs). Meanwhile, the thermal and power integrity problems are exacerbated since the power flux in 3D integrated circuits (3D ICs) increases linearly with the number of stacked layers. Moreover, the TSV structure in 3D ICs introduces new reliability problems since TSVs are vulnerable to various failure mechanisms (*e.g.* electromigration) and the failure of power-ground TSVs will cause voltage drop thereby significantly degrading the performance of 3D ICs. To make things worse, the high temperature, thermal gradient and power load in 3D ICs accelerate the failure of TSVs. Therefore, in order to push the 3D integration technology to full commercialization, the thermal, power integrity and reliability problem should be properly addressed in both design-time and run-time.

In 3D ICs, the heat flux will easily exceed the capability of the traditional air

cooling. Therefore, several aggressive cooling methods are applied to remove heat from the 3D IC, which include micro-fluidic cooling, phase change material based cooling *etc.* . These cooling schemes are usually implemented close to the heat source to gain high heat removal capability, thus causing more challenges to the design of 3D ICs. Unfortunately, physical design tools for 3D ICs with those aggressive cooling methods are lack. In this thesis, we will focus on 3D ICs with micro-fluidic (MF) cooling. The physical design for this kind of 3D ICs involves complex trade-offs between the circuit performance, power delivery noise and temperature. For example both TSVs and micro-cavities for MF cooling are fabricated in the substrate region. Therefore, they will compete in space: the allocation of signal TSVs should avoid micro-cavities to realize a feasible design, thus enforcing more constraints to the physical placement for 3D ICs. Moreover, power delivery networks (PDNs) in 3D ICs are enabled by power-ground (P/G) TSVs. The number and distribution of P/G TSVs are also constrained by micro-cavities which will influence the power integrity of the 3D IC. In addition, the capability of MF cooling degrades downstream the flow of coolant thereby causing large in-layer temperature gradient. The spatial temperature variance will affect the reliability of 3D ICs. in order to avoid it, the gate/modules in 3D ICs should be placed properly. In order to address the trade-offs 3D ICs with MF cooling, different design-time methods for application-specific ICs (ASICs) and field programmable gate arrays (FPGAs) are proposed, respectively. For 3D ASICs, we propose a co-design method that integrates the design of MF cooling heat sink and P/G TSVs to the physical placement for 3D ICs. Experiments on publicly available benchmarks show that using our method, we can achieve better results compared to the traditional sequential design flow. The case for 3D FPGAs is more complicated than ASICs since the routing and logic resources are fixed and the chip power and temperature is hard to estimate until the circuit is routed. Therefore, in this thesis, we first build a design space exploration (DSE) framework to study how MF cooling affects the design of 3D FPGAs. Following this, we utilize an existing 3D FPGA placement and routing tool to develop a cooling-aware placement framework for 3D FPGAs to reduce the temperature gradient.

Since the activity of 3D ICs cannot be completely estimated at the design stage, the run-time management, besides design-time methods, is required to address the thermal, power and reliability problems in 3D ICs. However, the vertically stacked structure makes the run-time management for 3D ICs more complicated than 2D ICs. The major reason of this is that the power supply noise and temperature can be coupled across layers in 3D ICs. This means the activity of one layer may affect the performance and reliability of other layers through voltage/temperature coupling. As a result, we cannot perform run-time management for each layer (perhaps implemented with different chips) of 3D ICs separately as in 2D systems. Therefore, the space of control nodes will become larger and more complicated. To make things worse, the existing run-time management techniques have various drawbacks (*e.g.* large off-line characterization overhead, poor scalability *etc.*), which needs more effort to improve. In this thesis, we propose a phase-driven Q-learning based run-time management technique which can tune the activity of the processor to maximize the 3D CPU performance subject to the reliability constraint.

THERMAL, POWER DELIVERY AND RELIABILITY MANAGEMENT FOR 3D ICS

by

Zhiyuan Yang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Ankur Srivastava, Chair/Advisor
Professor Donald Yeung
Professor Joseph F. JaJa
Professor Manoj Franklin
Professor Bruce L. Golden

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to express my appreciation to my advisor, Professor Ankur Srivastava for his guidance, advice and support throughout my Ph.D. study at University of Maryland. Professor Srivastava has introduced me to the exciting field of Electronic Design Automation, and he has provided numerous helpful guidance and encouragement during my five years in Maryland.

I would also like to thank Professor Donald Yeung, Professor Joseph JaJa, Professor Manoj Franklin and Professor Bruce Golden for their time to serve on this committee and their valuable technical feedback on the content of this dissertation.

Furthermore I would like to thank my wonderful colleagues. I should first thank my senior colleague Dr. Caleb Serafy, Dr. Tiantao Lu and Dr. Chongxi Bao for their guidance during my research and job searching. Second, I thank all my colleagues including the senior ones and my current colleagues, Yang Xie, Yuntao Liu, Ankit Mondal, Abhishek Chakraborty, for the great technical projects we have collaborated on and the fun times we have together throughout the years.

Finally I thank my beloved wife Yiyang for all her encouragement and support to make this dissertation possible. I thank my parents for their endless love and devotion throughout my studies.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ANN	Artificial Neural Network
ARMA	Autoregressive Moving Average
ASIC	Application Specific Integrated Circuit
CB	Connection Box
CDF	Cumulative Distribution Function
CLB	Configurable Logic Block
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DDR	Double Data Rate
DRAM	Dynamic Random Access Memory
DRM	Dynamic Reliability Management
DSE	Design Space Exploration
DSP	Digital Signal Processor
DTM	Dynamic Thermal Management
DVFS	Dynamic Voltage Frequency Scaling
ECC	Error Correcting Code
ECO	Engineering Change Order
EDA	Electronic Design Automation
EDP	Energy Delay Product
EM	Electromigration
EX	Execution Unit
FM	Fiduccia-Mattheyses
FPALU	Floating Point Arithmetic Logic Unit
FPGA	Field Programmable Gate Array
FPMULT	Floating Point Multiplier
GPGPU	General Purpose Graphics Processing Unit
HMC	Hybrid Memory Cube
HPWL	Half-parameter Wire-length
HQBP	Hierarchical Quadri-partitioning Based Placement
IALU	Integer Arithmetic Logic Unit
IC	Integrated Circuit
IFU	Instruction Fetch Unit

IMULT	Integer Multiplier
I/O	Input/Output
IP	Intellectual Property
IPnS	Instruction per Nano-second
LFU	Least Frequent Used
LSU	Load Store Unit
LUT	Look Up Table
MC	Micro-channel
MC	Memory Controller
MCP	Multi-core Processor
MDP	Markov Decision Process
MF	Micro-fluidic
MMU	Memory Management Unit
MST	Minimum Spanning Tree
MTTF	Mean Time To Failure
MWTA	Minimum Width Transistor Area
PDN	Power Delivery Network
P/G	Power-ground
P&R	Placement and Routing
PTPT	Post TSV-allocation Placement Algorithm
QoR	Quality of Results
RAT	Rename Unit
RL	Reinforcement Learning
RTL	Register Transfer Level
SB	Switch Box
SoC	System on Chip
SWM	Sampling Working Modes
TC	Thermal Cycling
TDDB	Time-dependent Gate Oxide Breakdown
TSV	Through Silicon Via
WL	Wire-length

Chapter 1: Introduction

For many decades of years, the semiconductor industry has witnessed an exponential increase in computing capacity due to the advantage of aggressive scaling of the CMOS technology. The consumer market has become used to such a rate of growth and expects this to continue into the future. However, as the technology node keeps decreasing, the traditional transistor scaling is approaching physical and economic limits which has slowed down the increase in computing power.

The 3D integration is a revolutionary technology to solve this problem. By vertically connecting active layers using through-silicon-vias (TSVs), this technology significantly reduces the interconnect delay and power, increases bandwidth and enables new computer architectures through heterogeneous integration (*e.g.* stacked memory-on-logic architecture). Two trends of the 3D integration is logic-on-logic stacking and memory-on-logic stacking. Logic-on-logic stacking can create highly connected circuits with high transistor density and large inter-core communication bandwidth in multi-core CPUs. Similarly, memory-on-logic stacking can significantly increase the communication bandwidth between memory and CPUs which can overcome the “memory wall” problem [10].

1.1 Thermal, Power Integrity and Reliability Issues in 3D ICs

Despite the advantages brought by the 3D integration, this new technology also faces several problems. Two chief challenges due to the nature of 3D integration are thermal [11–13] and power integrity [14, 15] problems. By stacking more layers in 3D ICs, the power flux increases significantly while the thermal resistance of the 3D stack also rises due to the inter-layer dielectrics. This will cause the heat to be trapped in the stack thus leading to severe thermal problems. On the other hand, while the TSV structure enables the inter-layer communication, it also increases the impedance of the PDN in 3D ICs [14]. Moreover, the TSV structure introduces new failure problems [14,16] which require to be properly addressed in order to maintain the reliability of 3D ICs.

In 3D ICs, the circuit performance, power integrity, chip temperature and reliability issues are highly coupled with each other. Large circuit activity will lead to high power which causes the temperature to increase. The increased temperature will in turn affect the critical path delay and leakage power. Large power load in 3D ICs will also cause significant voltage drop in the 3D PDN and large current density in P/G TSVs. The reliability of TSVs is a coupled function of temperature, current density and activity of 3D ICs. If P/G TSVs are failed, on the other hand, the increased impedance will affect the power integrity. Although similar relationships exist in 2D chips, the higher connectivity and spatial coupling between stacked layers in 3D ICs make simultaneous modeling and optimization is a must. In order to solve these problems, we can pursue two directions: design time methods and run-time

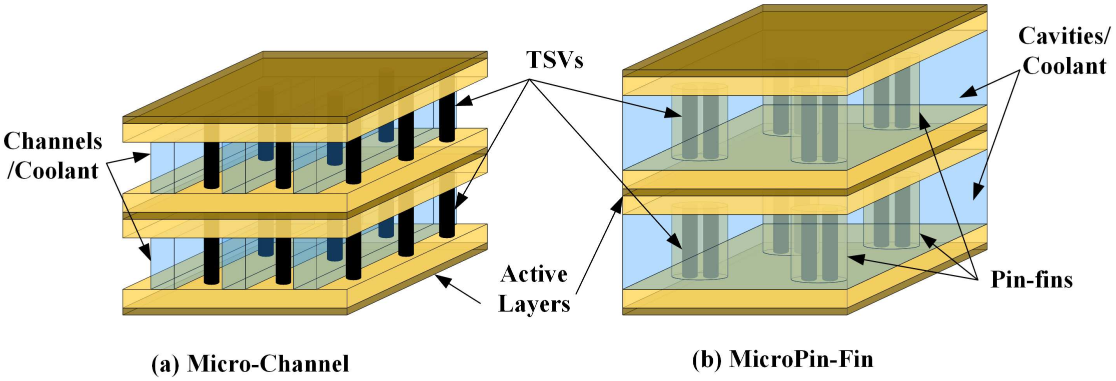


Figure 1.1: Two most common structures of micro-fluidic cooling: (a) micro-channel based cooling and (b) micropin-fin based cooling

methods. The property of 3D ICs brings new challenges to both directions.

1.1.1 Design-time Challenges

During the design time, the thermal problems of 3D ICs are solved with aggressive cooling methods such as thermoelectronics [17], phase-change material [18, 19] and micro-fluidic [20–22] *etc.* . Compared to the traditional air-cooling scheme, these new methods can significantly increase the heat removal capability. However, many of these advanced heat sinks are implemented close to the heat source (*e.g.* in the inter-layer region, integrated in the circuit *etc.*), hence they introduce new challenges to the physical design of 3D ICs (*e.g.* placement, PDN design *etc.*).

In this thesis, we focus on 3D ICs with embedded micro-fluidic (MF) cooling [21, 23] and investigate how this structure influences the physical design of 3D ICs. MF cooling comprises micro-cavities etched into the silicon substrate of each layer (as illustrated in Figure 1.1). Coolant (usually deionized water) is pumped into the cavities and flows across the circuit. Heat dissipated from active layers is transferred to the coolant and pumped out of the chip. Compared to the air cooling, MF cooling provides a much lower resistance path for heat flow from junction to ambient. However, this cooling scheme interacts with the floorplan of circuits and the 3D PDN, thereby causing problems to the physical design of 3D ICs. For example, TSVs (including signal and power-ground TSVs) and micro-cavities will compete for resources since they are all implemented in the substrate region. Moreover, the cooling capability of this method progressively diminishes along the direction of the flow as the fluid heats up, thus resulting in systematic in-layer thermal gradients from inlet to outlet even with uniform power profile. This property will potentially degrade the circuit reliability and affect the floorplan of the circuit. In Section 2.4, we will introduce the interactions in 3D ICs during the design stage in detail. In order to handle the complex thermal, power and performance trade-offs in 3D ICs with MF cooling, we need to develop advanced design tools.

1.1.2 Run-time Challenges

In practice, design-time methods alone are not enough to completely solve the thermal, power and reliability problems in 3D ICs. This is because we cannot get the accurate activity information of the 3D IC at the design stage, especially for large general purpose ICs such as CPUs. Therefore, we need dynamic management techniques to tune the activity of 3D CPUs during runtime to control the temperature, power integrity and reliability. Although dynamic management methods have been widely studied for 2D CPUs [6, 7, 24–27], they cannot be directly applied to 3D CPUs since the dynamic management in 3D CPUs is more complicated due to the high connectivity and spatial coupling:

In traditional 2D or 2.5D systems, different components (*e.g.* processor, memory *etc.*) are implemented separately and connected using off-chip bus lines. Therefore, the coupling of power and temperature between components is very weak. In this case, we can manage each component separately during the run-time. In 3D CPUs, however, chips are stacked on top of each other and the connectivity and physical coupling is much higher than that in 2D/2.5D counterparts. As a result, the activity of one layer will affect the physical property (*e.g.* temperature, power noise *etc.*) of other layers (hence affecting the performance of other layers besides its own layer). The coupled temperature and power will further affect the reliability of TSVs in other layers. Therefore, efficient run-time management for 3D CPUs requires proper modeling of the cross-layer interactions in the 3D stack.

On the other hand, existing dynamic management methods have various drawbacks. For example, a lot of methods require off-line characterization [24, 25]. However the behavior of 3D CPUs during runtime can be affected by various environmental factors (*e.g.* power noise, temperature *etc.*), thus making off-line characterization costly and even infeasible. On the other hand, due to the cross-layer coupling effect, the activity of different layers (*e.g.* different processor layers, memory layers, DSP layers *etc.*) need to be managed simultaneously. This will significantly increase the number of control nodes in run-time management. However, some existing techniques are very poor in scalability with the number of control nodes [6].

Therefore, more efforts are required to investigate efficient dynamic management methods for 3D CPUs.

1.2 Thesis Outline

In the following of this thesis, we will provide some in depth background on 3D ICs in Chapter 2. This includes details on the advantages and challenges for 3D integration, the background of micro-fluidic cooling, the physical design and run-time management issues for 3D ICs. The main research work will be introduced in Chapter 3 to Chapter 5. The design-time methods for ASICs and FPGAs will be discussed in Chapter 3 and Chapter 4, respectively. In Chapter 5, we investigate the run-time management issues for 3D CPUs.

In Chapter 3, we propose methodologies for placing 3D ASICs with MF cooling to co-optimize the performance (indicated by wire-length), temperature and power integrity. In this chapter, we solve this problem in two steps. In the first step, the MF cooling heat sinks are already designed before the placement for ASICs, while in the second step, we determine the placement of gates, signal TSVs, P/G TSVs and micro-channels for 3D ASICs with micro-channel based MF cooling.

In Chapter 4, we investigate the trade-offs in 3D FPGAs with MF cooling, which is much more complex than those in ASICs. We propose a design space exploration framework to achieve this and find optimal designs for 3D FPGAs with MF cooling with respect to temperature, performance and energy efficiency. Based on the proposed framework, we then propose an engineering-change-order (ECO) based placement and routing tool for 3D FPGAs with micro-channel based MF cooling.

In Chapter 5, we investigate the effect of thermal and power delivery noise coupling in 3D CPUs and how this affects the reliability of TSVs in the 3D CPU. In this chapter, we also propose a learning based run-time management method for 3D CPUs to maximize the CPU performance subject to some reliability constraints.

Finally, we conclude this thesis and propose some future works in Chapter 6.

Chapter 2: Background and Motivation

After several decades of exponential growth, CMOS technology has approached a point where traditional device scaling are unable to keep up with Moore's Law. This is because of several challenges in advanced technology nodes:

- As the feature size of transistors gets smaller, the leakage (*e.g.* gate tunneling leakage, sub-threshold leakage *etc.*) increases significantly. Although this problem can be solved with high-k dielectrics, this approach is doubted in compatibility with CMOS process.

- The wire delay and power in advanced technology nodes become dominant and the scaling in wire delay and power is much slower than the scaling of devices. This limits the bandwidth of off-chip interconnection and cause several problems such as the "memory wall" problem [10].

By bounding multiple chips with existing technologies together and connecting chips with TSVs, the 3D integration provides a revolutionary solution to increase the logic density without costly device scaling while reducing the interconnect delay and power. Moreover, the 3D integration also enables heterogeneous bounding. One application of this property is stacked-memory-on-logic, which, to a great extend, addresses the "memory power" problem due to the large memory-processor interconnect bandwidth.

Despite the advantages of 3D integration, it also comes with challenges. Thermal and power integrity are two of the most important problems in the 3D IC. Moreover, the TSV structure introduces new reliability issues and the high temperature and power load will make such reliability problems even more severe.

The 3D integration technology can be applied to implement different types of chips, *e.g.* ASICs, FPGAs *etc.* . Different types of 3D ICs have different properties and challenges. However, they share some common temperature, power delivery and reliability challenges, which will be introduced in detail in the following sections.

2.1 Thermal Problem and Micro-fluidic Cooling

The thermal problem in 3D ICs is exacerbated due to the vertical stacking of multiple active layers. According to [22], power density in 3D ICs can reach as high as $5kW/cm^2$ which significantly exceeds the capability of traditional air-cooling. One viable solution to the thermal problem is to use embedded micro-fluidic cooling. Two common structures of the MF cooling heat sink are micro-channels and micropin-fins which are illustrated in Figure 1.1. In MF cooling, micro-cavities are etched into the inter-layer region with coolant (*e.g.* de-ionized water) pumped into the channels to take away heat from the chip. Previous researches demonstrate that the single-phase micro-channel cooling enjoys large cooling capability (as high as $700W/cm^2$ [23]) while this value is even higher if two-phase cooling is applied. However, this new structure also brings challenges to the design of 3D ICs:

1. TSVs cannot go through micro-cavities thus leading to the trade-off between the performance (impacted by the vertical bandwidth) and cooling capability (determined by the micro-cavity parameters). For example, if micro-channel based MF cooling is applied, the number of micro-channels determines the cooling capacity when the pressure drop across the channel is fixed [28]. Therefore, with this cooling scheme, the number and allocation of micro-channels should be considered during the placement of TSVs.

2. Since the coolant keeps absorbing heat while flowing downstream to the outlet, the cooling capability of this embedded heat sink degrades along the flow direction. As a result, the temperature gradient within each layer may be extremely large even if the power profile is uniform. Large temperature variation will cause severe reliability problems and should be properly handled during the physical design stage of 3D ICs.

In order to overcome the challenges brought by the micro-fluidic cooling, people first investigate to optimize the design of MF heat sinks [29, 30]. However, the irregular geometry of the heat sink structure will increase the fabrication cost and reduce the yield of 3D ICs. Moreover, all these works assume the floorplan of 3D ICs and the position of signal TSVs are already fixed before the design of micro-fluidic cooling. In practice, there are complex interactions between the placement of gates, signal TSVs and micro-channels in 3D ICs with micro-fluidic cooling, hence merely optimizing the heat sink will not completely solve the problem. Therefore, another research direction is pursued where the MF cooling structure and the 3D IC physical design are co-optimized [31,32]. In this thesis, we will focus on the placement stage for the 3D IC.

As introduced above, there are two major types of MF cooling: micro-channel based and micropin-fin based cooling schemes. MF cooling with different schemes are modeled differently. So far MF cooling thermal models are widely studied [3,28,33]. In the following part of this section, we will introduce the thermal models for the two major MF cooling schemes which will be used in our work.

2.1.1 Thermal Model for Micro-channel Based MF Cooling

Resistance network is one of the most popular approaches of modeling the steady state thermal behavior of micro-channel based MF cooling (Figure 1.1(a)). This method is motivated by the duality between voltage/current and temperature/heat flow [28]. This modeling approach partitions a 3D IC into several thermal grids and the heat transfer path between every two grids is represented with a resistance, which is called thermal resistance. A thermal resistance is calculated based on the material and dimension of the related thermal grids. For example, the model proposed in [28] is illustrated in Figure 2.1. In this model, three types of thermal resistance are used: R_{cond}, R_{conv} and R_{heat}. More details of the thermal model can be found in [28]. Given the power distribution of the 3D IC, we can calculate the heat flux of each thermal grid. The heat flux of all thermal grids forms the power profile, $\boldsymbol{P}$. The temperature profile is then computed by $\boldsymbol{T} = \boldsymbol{R}\boldsymbol{P}$, where $\boldsymbol{T}$ contains the average temperature for each thermal grid.

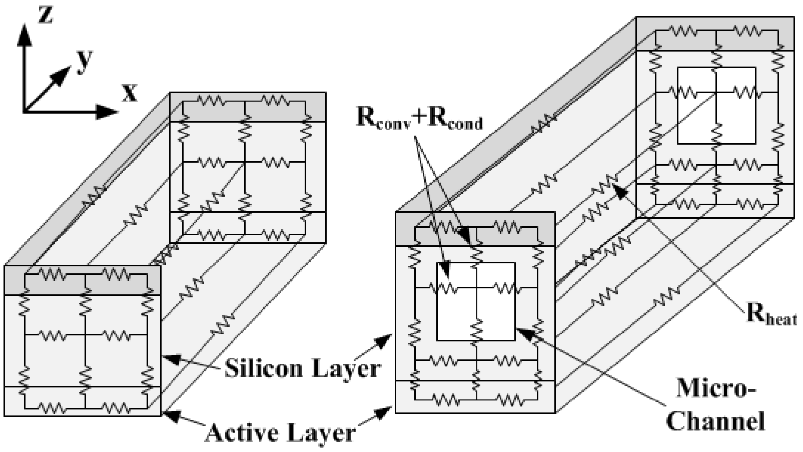


Figure 2.1: Thermal resistance network of 3D ICs with micro-channel based MF cooling

2.1.2 Thermal Model of Micropin-pin Based MF Cooling

Wan et al. [3] developed a thermal model for micropin-fin fluid cooling (Figure 1.1(b)). According to the thermal model, 3D IC is divided into several thermal volumes, each modeling the temperature around one pin. The size of a thermal volume is determined by the height of each layer in the 3D IC as well as the pitch of pins. The top view of the thermal volumes is illustrated in Figure 2.2. Note that the edge length of a thermal grid is exactly the same as the pitch of pins (S).

Each thermal volume is assumed to have a uniform fluid temperature T_f and a uniform silicon temperature T_s. The heat flux in each thermal volume is obtained from the input power map. Energy balance analysis is conducted for each thermal volume through a set of equations. Since the thermal parameters, such as the thermal conductivity, specific heat capacity *etc.* are affected by the temperature, this model uses several iterations to acquire the energy balance. Readers may refer

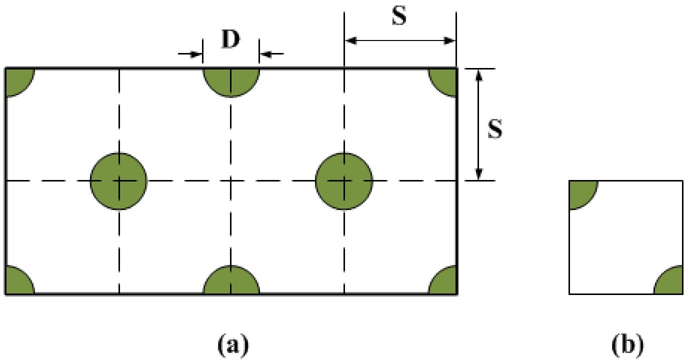


Figure 2.2: The top view of the thermal volumes with (a) the illustration of the pin diameter (D) and pitch (S) and (b) a single thermal volume (the shaded spots represent the pins)

to [3] for more details about the thermal model.

2.2 Power Delivery Problem

The power of 3D ICs is supplied from the off-chip circuit through power-ground bumps. Afterwards, the power is distributed to each layer using power-ground TSVs (P/G TSVs). The power delivery in 3D ICs has several challenges:

1. P/G TSVs introduce new impedance to the power delivery network which will cause more voltage drop compared to the 2D PDN.

2. While the power demand potentially increases with the number of layers, the number of power supply pins and P/G TSVs may even decrease if the footprint area of the 3D IC shrinks. The imbalance between the supply and demand of power leads to significant voltage drops in 3D ICs thus degrading the performance and reliability of the circuit.

3. Due to the restriction of the footprint area, there is a tradeoff between the number of signal TSVs and P/G TSVs. More P/G TSVs will take away spaces for signal TSVs thus causing the reduction of the vertical signal bandwidth and signal routing congestion.

In order to suppress the power noise in 3D ICs, a lot of researches have been done on the design of 3D PDNs (assuming the power profile of the 3D IC is already known). A number of researchers try to optimize the size, density as well as the distribution of P/G TSVs to reduce the voltage drop [34, 35]. Gu *et al.* propose a new design of 3D PDNs using multi-story supply voltages and this 3D PDN can reduce the current demand and hence suppress the voltage drop [36]. Among all the current works on the design of 3D PDNs, only a few works investigate the co-design of 3D PDNs and the floorplan of 3D ICs to handle the interactions between the two structures [37].

The design of 3D PDNs becomes more complex when the embedded MF cooling is used to remove heat from 3D ICs, due to the complex interactions between gates, signal TSVs, micro-channels and P/G TSVs (Figure 2.4). We will introduce this in detail in Section 2.4.

3D PDN Modeling

The 3D PDN can be modeled as the combination of an off-chip and on-chip RLC network as illustrated in Figure 2.3 [38]. The off-chip parameters can be estimated through simulation or measurement of the real chip [38]. The on-chip network is a grid network with each edge indicating a power delivery path between

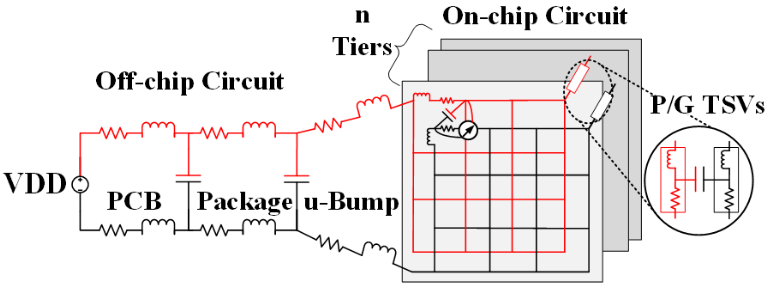


Figure 2.3: Illustration of the PDN model in the 3D IC

two grid points. Each power delivery path is modeled as a series of resistance and inductance which is calculated based on the P/G wire material and dimension as well as the grid size [38]. A P/G TSV is also modeled as a series of resistance and inductance. The power grids and ground grids are decoupled with capacitance. Each on-chip power grid point is attached with a current load which can be calculated from the power distribution of the 3D IC. Given the current load profile and the PDN network as well as the source supply voltage, we can calculate the voltage at each power grid point which serves as the real supply voltage to the gates in the vicinity. Then, the maximum voltage drop across the chip can be calculated. In this thesis, we mainly care about the steady state voltage drop, thus the PDN model can be reduced to a resistance network. Then we can just simulate the power-network and the voltage noise in the ground-network can be obtained symmetrically.

2.3 Reliability Issues

TSVs are crucial structures in the 3D IC which enable the inter-layer communication. However, they suffer from various reliability degradations, such as

Electromigration (EM), thermal cycling, cracking *etc.* .

EM is one of the most important failure mechanisms in metal interconnects. EM causes the metal atomic diffusion thus generating voids in the interconnect which causes open-circuit or short circuit. The EM in TSVs is influenced by several factors such as temperature, current density, mechanical stress *etc.* . So far, TSV's EM has attracted a lot of research interests and the modeling of TSV's EM has been widely studied [13,39–41].

Most TSV reliability degradation mechanisms (including EM) are associated with the chip layout geometry, the dimension of TSVs and the chip temperature *etc.* . For example, large temperature (resulted from the high workload of the 3D IC) will accelerate the EM process thus causing more TSV loss and the temperature is determined by the power load which is related to the work load of the 3D CPU. Moreover, different sizing of wires or distribution TSVs will influence the density of current flow through the TSV, which will affect the TSV loss. Therefore, the TSV reliability can be mitigated during the design stage or at the run-time. While there are a lot of work on design-time optimization of TSV reliability [16,42,43], there is a lack of research on run-time management for TSV reliability in 3D ICs.

2.4 Interdependence Between Temperature, Power Delivery and Reliability

In 3D ICs with micro-fluidic cooling, there is complex interdependence between the temperature, power delivery and reliability both at the design stage and during

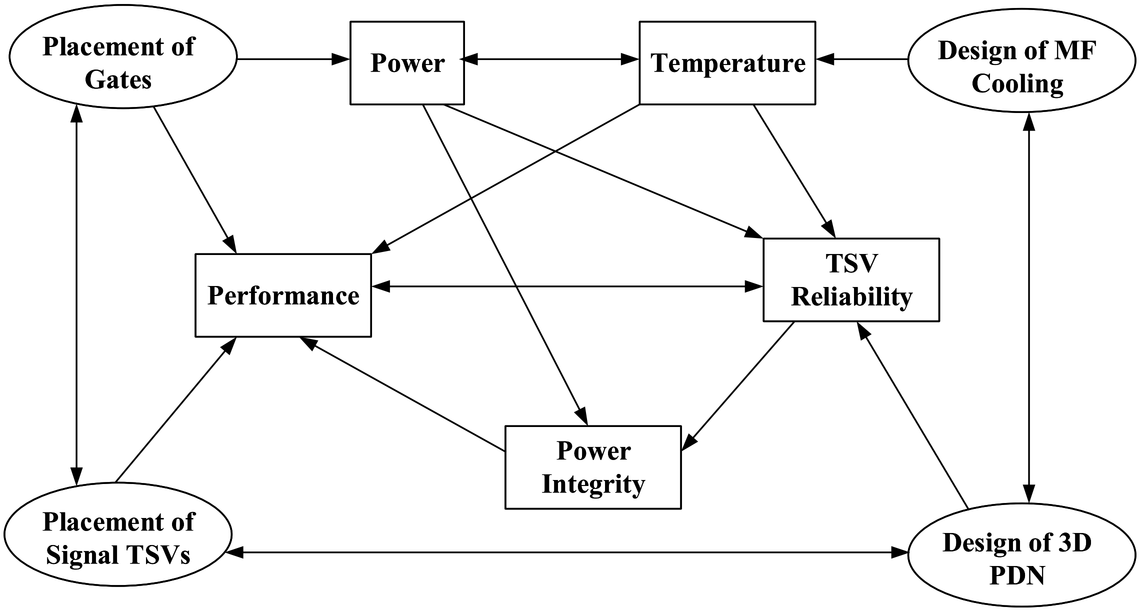


Figure 2.4: The interdependent relationships in 3D ICs with MF cooling

the run-time.

During the design stage, these properties are determined by the design of different aspects of 3D ICs (*e.g.* placement of gates, design of 3D PDNs *etc.*) which interact with each other as well. These interactions are summarized in Figure 2.4. Due to these interactions, partially optimizing one aspects in the design stage (*e.g.* placement of gates and TSVs) might cause negative effects to other aspects thus resulting in suboptimal or even infeasible designs. We will introduce the interdependent relationships as follows.

1. **Placement of Gates.** The placement of gates affects the 3D IC performance as well as the power profile. The power consumption in the 3D IC determines the temperature and power supply noise, thus affecting the design of MF cooling and 3D PDN systems in the 3D IC. On the other hand, both the

temperature and the power supply noise will influence the performance of the circuit (e.g. high temperature will cause the interconnect delay to increase *etc.*).

2. **Placement of Signal TSVs.** The placement of signal TSVs interacts with the placement of gates to affect the performance of the circuit. Moreover, due to the spatial conflict between TSVs and micro-cavities (*e.g.* micro-channels), the placement of signal TSVs will influence the allocation of P/G TSVs and micro-cavities, thus affecting the design of 3D PDN and MF cooling systems. Thereby, the power delivery noise and temperature will be affected which further influence the performance and reliability of the 3D IC

3. **Design of MF Cooling.** The design of MF cooling has impacts on the performance and reliability of the 3D IC by influencing the temperature. This will force the gates and signal TSVs to be placed accordingly. On the other hand, the allocation of micro-cavities will determine where the signal and P/G TSVs can be placed thus affecting the performance and power noise of the chip.

4. **Design of 3D PDN.** The design of 3D PDNs first influences the power noise of the chip. Inferior supply voltage will affect the chip latency, thus forcing the gates and signal TSVs to be placed accordingly to achieve a certain performance requirement. On the other hand, the allocation of P/G TSVs interacts with the placement of signal TSVs and the design of micro-cavities.

During the run-time, the TSV reliability is at the central position among the interdependent relationships in 3D ICs. The failure process of TSVs is a coupling effect of power, temperature and the circuit performance. However, failure of TSVs will significantly degrade the performance as well as the power integrity of the 3D IC. Moreover, the TSV reliability issue cannot be fully addressed during the design stage. Although we can tune the TSV dimension and distribution or implement redundant TSVs during the design stage to mitigate the TSV failure induced system reliability degradation [42–44], these design-time methods are usually very expensive. Without circuit activity information, most methods will result in over-design [44]. Therefore, the TSV reliability issue can only be efficiently addressed during the run-time stage with dynamic management techniques.

2.5 Placement for 3D ICs

Figure 2.5 illustrates the basic IC design flow. After a circuit is designed, it is synthesized to the gate-level netlist which is fed into the **Physical Design Stage**. After physical design, we will perform verification to the design and then proceed to the packaging stage. As illustrated by the figure, **Physical Design Stage** is the center of the IC design flow, while placement is an important step in the physical design stage. Therefore, in this thesis, we will focus on the placement of 3D ICs. Moreover, in the traditional IC design flow (as illustrated by Figure 2.5), PDN design is completed at the Power-planning stage while thermal issues are addressed at the Packaging stage. However, in 3D ICs, power delivery, temperature and circuit placement are highly interacted (as mentioned in Section 2.4). Therefore, we will investigate placement methods for 3D ICs that can design the 3D PDN and the MF

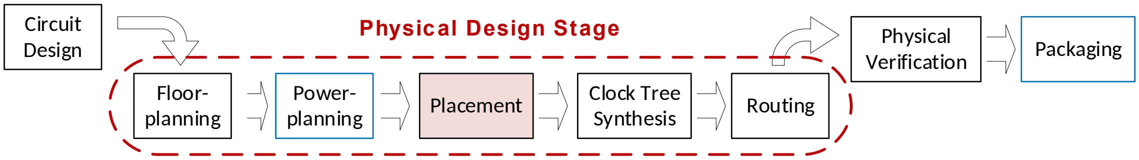


Figure 2.5: The IC design flow

cooling system simultaneously

In the following part of this section, we will introduce the placement problem in 3D ICs and summarize popular methods to achieve the placement. For a given circuit netlist, the 3D placement problem aims to assign a position (x_i, y_i, z_i) to each gate and signal TSV such that the total wire-length and the number of signal TSVs are minimized, subject to constraints such as non-overlapping gates. Two popular wire-length models are 3D half-perimeter wire-length (HPWL) [8] and quadratic total wire-length model [45]. There are two different design flows for 3D placement: 1-step Design Flow and 2-step Design Flow. The two design flows will be introduced in Section 2.5.1. Besides the design flow, there are four existing placement paradigms which will be described in Section 2.5.2.

2.5.1 Different Design Flows

The two design flows of 3D IC placement are 1-step Design Flow (also known as "full-3D placement" [46]) and 2-step Design Flow (also known as "pseudo-3D placement" [46]).

1-step Design Flow Given the gate-level netlist, the 1-step Design Flow determines the position of gates and TSVs in the 3D space in one step by solving the following problem:

- **P1:** minimizing the wire-length and the total number of TSVs by determining the 3D position of gates subject to the non-overlapping constraint and other constraints.

Note that, sometimes, some constraints can be enforced as a part of the objective function. This type of design flow explores the full design space (for placement) of 3D ICs thus can achieve optimal results theoretically. However, in practice, due to the complexity of the constraint space, the 3D placement problem is usually not convex. Therefore, this design flow (with large design space) might significantly degrade the performance of heuristic methods.

2-step Design Flow Different from the "1-step Design Flow", 2-step Design Flow solves the 3D placement problem by solving the following two sub-problems successively:

- **P2-1:** minimizing the total number of TSVs by determining the layer assignment of gates subject to the area balance constraint and other constraints.
- **P2-2:** minimizing the total wire-length by determining the 2D position of gates and TSVs on each layer subject to non-overlapping constraint and other constraints.

Similar to the 1-step Design Flow, some constraints can be enforced as a part of the objective function. In this design flow, the layer assignment of gates will not be changed after **P2-1** is solved. This reduces the design space of the 3D placement problem. Although this may lead to suboptimal results, more complex constraints can thus be implemented in this design flow while the problem can still be solved efficiently.

2.5.2 Different Placement Paradigms

Currently, there exist four placement paradigms: Transformation-based Placer, Partition-based Placer, Analytical Placer and Force-directed Placer. These four paradigms are introduced as follows:

1. **Transformation-based Placer:** This paradigm transforms a 2D placement result into 3D space by folding and stacking a 2D design [47].

2. **Partition-based Placer:** This paradigm assigns weights to different nets and determines the position of gates and TSVs using either recursive bi-partition [48] or quadri-partition methods [32].

3. **Analytical Placer:** This paradigms formulates the 3D placement problem using analytical functions. For example, it uses log-sum-exp model to relax the HPWL model and applies a density smoothing function to penalize the overlap between gates (and signal TSVs) [49].

4. **Force-directed Placer:** This paradigm uses the spring energy of an elastic spring system to mimic the quadratic wire-length model [50]. The derivative of the spring energy is a force (referred to as net force). Overlapping constraint and other constraints are also modeled as “forces” in the spring system. The

gates and TSVs are moved iteratively under all the forces to achieve a balance.

2.6 Run-time Management Methods

Traditionally, without enough information of the circuit activity, the chips and packaging are designed such that a safe operating condition is maintained even when the chip is dissipating the maximum power for a sustained period of time. This will lead to costly over-design (e.g. implementing to many redundant TSVs for reliability issues, using expensive cooling method *etc.*). On the other hand, if run-time management is applied, the system can be designed for a target condition that is much closer to the average-case for the real working load. Therefore, in order to fully solve the thermal, power delivery and reliability problems in 3D ICs, run-time management methods should also be investigated.

The existing run-time management method can be generally categorized into two types:

- **Off-line Profiling Based Methods**. These methods first profile over a small set of benchmarks (estimated according to the real working mode) to get a profile indicating the best CPU configuration for each possible condition. During the management stage, we only need to match the real working load with the profiled condition and tune the CPU configuration accordingly.

- **Learning Based Methods**. These methods do not require the off-line profiling but can learn the optimal action for each condition during the run-time.

Initially, most dynamic management methods are off-line profiling based [24–

26]. This kind of methods is efficient in embedded systems whose function is specific which can be easily characterized during the off-line stage. However, for general purpose processors, since their function is more complicated, it is not practical to get the knowledge of tasks a priori thus off-line profiling based methods are no longer efficient and might introduce extremely large profiling overhead. In order to address this problem, learning based management methods started to gain more interests [1, 2]. This kind of management methods gather the information of tasks during the run-time and adjust the management policy accordingly. Therefore, no off-line profiling is required for this kind of methods.

Chapter 3: Co-Design Methodologies for 3D ASICs

In this chapter and the next chapter, we will introduce the methods of fixing the complex thermal and power integrity problems during the design-time. This chapter focuses on application specific integrated circuits (ASICs) while Chapter 4 will focus on the design of FPGAs.

In this chapter, we introduce the co-design methodologies for 3D ASICs with MF cooling. The aim of these methodologies is to handle the interdependent relationships described in Section 2.4. The co-design problem is divided into two levels according to the trade-offs considered:

1. **First Level:** The MF cooling heat sink is fixed and the we place the gates and signal TSVs aware of the MF cooling to reduce the peak temperature, in-layer temperature gradient while maintaining a structurally feasible design. In this level of problem, the PDN design is not considered. Therefore, we just model the impact of MF cooling during the placement for 3D ICs. The results will be compared against placement methods that are unaware of the MF cooling structure.

2. **Second Level:** The MF cooling heat sink is designed simultaneously with the placement of gates, signal and P/G TSVs such that all the trade-offs

introduced in Section 2.4 can be handled during the placement stage of 3D ICs. Note that, in this level of problem, we consider all the design-time relationships introduced in Section 2.4.

In this chapter, we propose a hierarchical partitioning based placement framework which is then extended to solve these two levels of the co-design problems.

The chapter is organized as follows: Section 3.1 introduces the Hierarchical Quadri-Partitioning based Placement framework proposed in this work which is used to solve the co-design problems. The two levels of problems will be introduced in Section 3.2 and 3.3 and the related experimental results will be shown and discussed in Section 3.4. In Section 3.5, we will conclude this chapter.

3.1 Hierarchical Quadri-Partitioning Based Placement Framework

The methodologies proposed in this chapter to solve the co-design problems in 3D ASICs are based on a hierarchical quadri-partitioning based placement (HQBP) framework. This framework belongs to the partition-based placement methods and follows the **2-step Design Flow** as described in Section 2.5.1. Therefore, our framework proceeds successively with two steps: **layer partitioning** and **in-layer placement**. During the "layer partitioning" step, **P2-1** is solved while **P2-2** is solved in the "in-layer placement" step. Both **P2-1** and **P2-2** are defined in Section 2.5.1. This framework is then followed by a detailed placement method (*e.g.* [51]) to remove the overlap and refine the placement. In Section 3.1.1, we will introduce the overall flow of the HQBP framework. Note that the HQBP framework alone only determines the position of gates and signal TSVs. (In the rest part of this chapter, we will use "TSV" to indicate the "signal TSV" if not specified.) However, it is open to modifications such that the impacts of MF cooling and the design of MF cooling heat sinks (*e.g.* the placement of micro-channels) can be integrated into this framework. In this section, we will first introduce the basic algorithm of the HQBP framework. Any modifications to the HQBP framework to solve the co-design problem will be introduced in Section 3.2.2 and Section 3.3.1.

3.1.1 Overall Flow of the Algorithm

3.1.1.1 Layer Partitioning

In this step, we partition the netlist across layers to determine the assignment of gates on each layer. This is achieved by partitioning the netlist across layers. This can be achieved using the hMETIS partitioning tools [52]. hMETIS performs multi-way multi-level Fiduccia-Mattheyses (FM) partitioning for the netlist to minimize the total number of cuts between each pair of sets while balancing the size of each set. Note that, reducing the number of TSVs is a common criterion when performing layer assignment since more TSVs will introduce higher area overhead, fabrication and testing cost. After the vertical partitioning step, the following values are determined: (1) the number of gates on each layer, (2) the number of TSVs between each pair of adjacent layers and (3) the footprint area (A_{fp}) of the 3D IC given the white-space ratio, α_W, for packing the gates and TSVs (Equation 3.1 where A_l^{gate} and A_l^{TSV} are the total area of gates and TSVs on layer l, respectively).

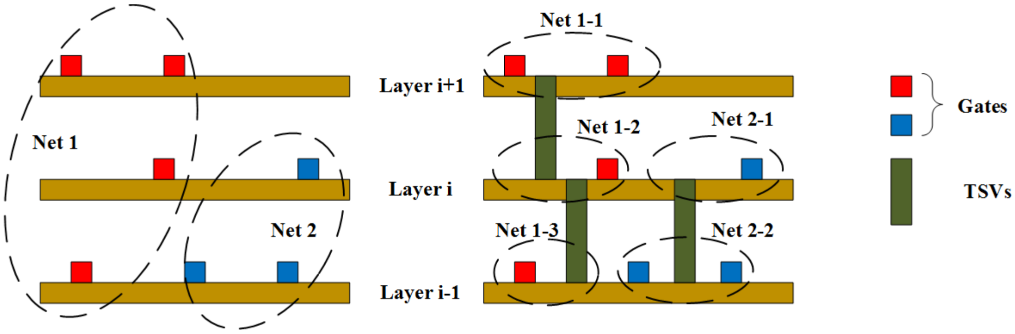


Figure 3.1: Illustration of the 3D nets before (left) and after (right) the net splitting

These values will not change during the HQBP.

$$A_{fp} = \alpha_W \times \max_{l \in \{1,2,\ldots,N_{layer}\}} (A_l^{gate} + A_l^{TSV}) \tag{3.1}$$

When MF cooling is considered during the design of 3D ICs, this vertical partitioning stage will be modified by enforcing more constraints to the gate partitioning or adding more steps for designing MF cooling and power delivery systems. This will be elaborated in Section 3.2 and 3.3.

3.1.1.2 In-layer Placement

In this step, we first perform net splitting [53] to eliminate the cross-layer nets. This is achieved as follows: for each net which connects gates in adjacent layers, we create a TSV cell and connect terminals from a given layer to this TSV cell rather than to the terminals in the next layer directly. For nets crossing more than one layer, a similar splitting approach is used (Figure 3.1). After net splitting, all gate terminals are connected to other gate terminals on the same layer or TSV cells. A TSV cell does not belong to any layers to which it connects. However, it will affect the space for placement in the . In this work, we assume Via-first Technology [54]

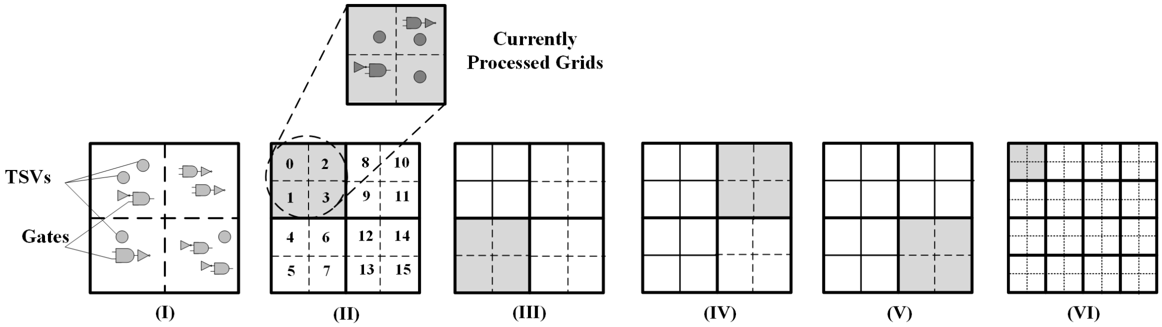


Figure 3.2: Illustration of the HQBP framework with the first three partition levels ((I) illustrates the first partition level, (II)-(V) illustrate the second partition level and (VI) illustrates the third partition level; the shaded grids indicates the currently processed grids)

is applied to fabricate TSVs, thus only the placement of gates on one layer will be affected. That is, if a TSV connects layer i and $i+1$, we assume layer i is affected. This area constraint can be easily modified if other TSV fabrication technologies are applied [54].

Following the net splitting, we first partition each layer of the 3D IC into four grids and randomly assign the gates and TSV cells to these grids. This is illustrated in Figure 3.2(I). The partition of gates will affect the wire-length and the overlap. However, since the specific position of gates and TSV cells in each grid is not specified at this stage, the "wire-length" (which is referred as **wire-length cost** in this work) is estimated using the method introduced in Section 3.1.2 and the total overlap is estimated using Equation 3.2:

$$OverLap = \sum_{l=1}^{N_{layer}} (\sum_{g=0}^{N_{grid}} (\max(A_{l,g}^{gate} + A_{l,g}^{TSV} - A^{grid}, 0))), \tag{3.2}$$

where N_{layer} and N_{grid} are the number of layers and the number of grids on each layer, respectively. $A_{l,g}^{gate}$ and $A_{l,g}^{TSV}$ are the area of gates and TSVs, respectively, in grid g on layer l. A^{grid} is the area of a grid. The initial cost is then calculated as the sum of wire-length cost and the total overlap. Following this, we greedily move one gate or TSV cell in each iteration to reduce the cost. To achieve this, we need to calculate the "gain" of each gate of TSV cell if moving it from the current grid to another grid. The "gain" is calculated as the current cost minus the new cost after the move. The gate or TSV cell with the largest gain is then selected to be moved to the corresponding grid. This gate or TSV cell is then "locked" until all gates and TSV cells are moved. The gain of each gate and TSV cell is updated after each move. We follow the methods introduced in [55] to calculate and update the gain of each gate or TSV cell which can be performed efficiently. When all the gates and TSV cells are moved, we check the change of cost compared to the initial cost (*i.e.* the cost when no gates or TSV cells are moved). If the cost reduction is significant, we unlock all the gates and TSV cells and another round of move is performed to further reduce the cost. Otherwise, we will proceed to partition each grid into four parts (as illustrated in Figure 3.2(II)) and randomly assign the gates and TSV cells in that grid to the four parts. Note that, in order to speed up the process, we do not need to wait to check the cost until all the gates and TSV cells are moved. We can stop move when we discover a number (*e.g.* N_{inc}) of continuous increase in cost because we can infer that no better cost will appear afterwards.

At the second partition level (Figure 3.2(II)-(V)), we have 16 grids on each layer, among which grids 0-3, 4-7, 8-11, 12-15 are partitioned from four different grids of the first partition level, respectively. We first process the grids 0-3 by moving gates and TSVs cells in these grids. We call these four grids the currently processed grids. The procedure of processing these grids is similar to the first partition level while the wire-length cost and overlap are calculated using the gates and TSV cells in these grids. The calculation of the wire-length cost for each partition level will be introduced later in Section 3.1.2. The total overlap for grids 0-3 at this level is modified from Equation 3.2 and shown below (note that only grids 0-3 are included in this calculation):

$$OverLap = \sum_{l=1}^{N_{layer}} (\sum_{g=0}^{3} (\max(A_{l,g}^{gate} + A_{l,g}^{TSV} - A^{grid}, 0))), \tag{3.3}$$

When finishing processing the first four grids at this partition level (*i.e.* grids 0-3), we proceed to process the grids 4-7 and so forth. When all the 16 grids are processed, we continuously partition the chip into 4^l grids for the l^{th} partition level following the same procedure as introduced above.

The algorithm stops partitioning the chip when the area of a grid is small enough (*e.g.* each grid contains less than 10 gates). Following this, we can perform detail placement to further refine the position of gates and TSV cells.

3.1.2 Calculation of the Wire-length Cost

Wire-length cost (C_{WL}) is an essential part of the total cost. Due to the property of the partition-based placement, the location of each gate and signal TSV is refined by iteratively partitioning the chip. Therefore, the wire-length cannot be accurately computed during each partition level. Moreover, at higher partition levels, we would like to estimate the impacts on the total wire-length by just focusing

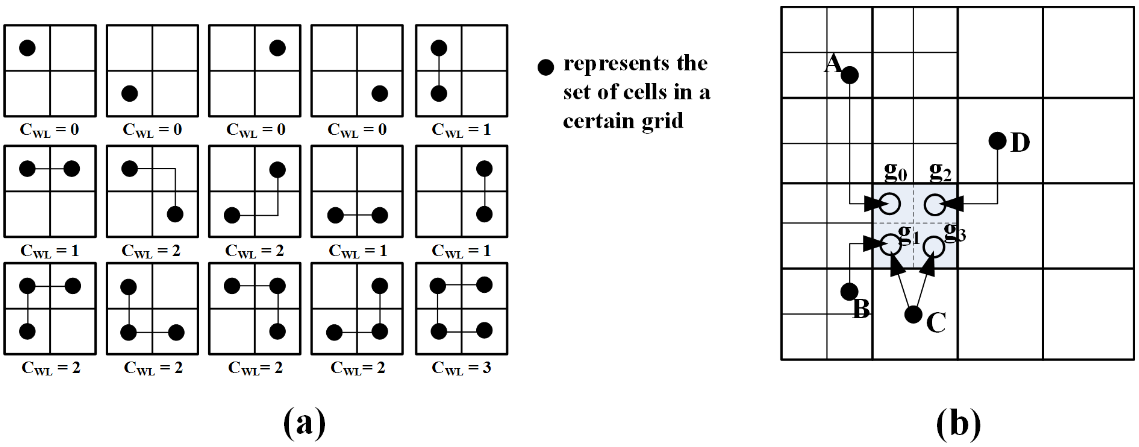


Figure 3.3: Illustrations of the (a) net configurations and the associated wire-length cost and (b) the terminal propagation with the shaded grids indicating the currently processed grids

on cells within the currently processed grid (thus the computation cost for each subproblem is reduced). These facts motivate us to propose a methodology to calculate C_{WL} which will be introduced as follows:

As illustrated in Figure 3.2(I), the gates and signal TSVs are partitioned into four grids during the first partition level. Within each grid, the actual physical positions of gates and signal TSVs are not specified. Therefore, we assume they are located at the center of the grid. In this way, the distribution of cells (*i.e.* gates and signal TSVs) belonging to the same net can be captured with 15 configurations, as illustrated in Figure 3.3(a). For each net, we can determine its configuration and the wire-length cost of the net can thus be calculated using the minimum spanning tree (MST) of the configuration.

When the grids are further partitioned, we intend to use the "wire-length cost" of cells within the currently processed grid (denoted as "inner cells") to estimate the total wire-length. This cannot be simply achieved using the approach stated above since the inner cells may connect to cells of the neighboring grids (denoted as "external cells") and the external cells may have impacts on the placement of inner cells. In order to overcome this problem, we use terminal propagation [55] to capture the impacts of external cells as illustrated in Figure 3.3(b). In this figure, the current processed grids (g_0, g_1, g_2 and g_3) are shaded and four external cells (A, B, C and D) are highlighted. Terminal propagation technique will, basically, "propagate" an external cell to a dummy cell (illustrated with an empty circle in the figure) in the currently processed grid that is closest to the external cell according to Manhattan Distance. As illustrated in the figure, A, B and D are propagated to g_0, g_1 and g_2, respectively. Since both g_1 and g_3 have the same distance to C, this external cell is propagated to both of these grids. If more than one external cells are propagated to the same grid (*e.g.* both B and C are propagated to g_1), they are regarded as being propagated to the same dummy cell. After the terminal propagation is performed, the C_{WL} among the currently processed grids can thus be calculated including the dummy cells and capture the change of the total wire-length.

3.2 1^{st}-level Co-Design Problem: Cooling-Aware Placement

In this section, we solve the cooling-aware placement problem for 3D ICs with MF cooling (1st-level problem). We use the micropin-fin based MF cooling (Figure 2.1(b)) for illustration, due to its more complex cooling schemes than micro-channel based methods. However, the technique proposed in this section can be applied to micro-channel based MF cooling given a proper thermal model.

The structure of the micropin-fin is fixed (*i.e.* the dimension and the pitch of pins are fixed) before the placement of gates and signal TSVs. We modify the HQBP framework introduced in Section 3.1 to minimize the total wire-length and the average in-layer temperature gradient with the following two constraints: (1) TSVs can only be placed at the position of pins and (2) the overlap between cells (*i.e.* gates and TSVs) is within a tolerable range.

The key challenge of this problem is:

- In order to co-optimize temperature uniformity and the placement of 3D ICs, we need a thermal model to characterize the cooling of micropin-fin. Although we have models describing the micropin-fin cooling (Section 2.1.2), the existing model takes quite a long time to estimate the temperature, which is obviously impractical to implement these models directly to the placement framework.

In Section 3.2.1, we will simplify this model so that it can be applied in the physical design flow. In Section 3.2.2, we will introduce the modification for the HQBP framework (Section 3.1). The experimental results will be shown in Section 3.2.3.

3.2.1 Simplification of the Thermal Model

In Section 2.1.2, we have introduced a model for micropin-fin based MF cooling. The problem of this thermal model is its complicated calculation which makes it too slow to be used in the physical design flow of 3D ICs. However, if we regard the thermal model as a "black box", the process of calculating temperature can be expressed simply as follows:

- The input power map ($\boldsymbol{P}$) is constituted of the power of each thermal volume. $\boldsymbol{P}$ and the physical design of micropin-fin heat sink (such as pitch and diameter of pins, pumping power, velocity of the fluid *etc.*) are then fed into the thermal model to generate the temperature profile $\boldsymbol{T}$ which is constituted of the average temperature in each thermal volume. Therefore, we can represent the thermal model using $\boldsymbol{T} = f(\boldsymbol{P})$, where $f(\boldsymbol{P})$ represents the set of functions used to capture the thermal properties of the micropin-fin based MF cooling (Section 2.1.2).

In the following paragraphs, we will introduce a method to linearize the thermal model which significantly speeds up the calculation.

During our research, we found if we fix the total power to P_0, for a certain physical design of the heat sink, θ, the thermal model $\boldsymbol{T} = f_\theta(\boldsymbol{P})$ [3] (where $\mathbf{1}^T\boldsymbol{P} = P_0$ can be approximated by a linear model. That means we can find a thermal resistance matrix $\boldsymbol{R}_{\theta,P_0}$ such that $\boldsymbol{T} = \boldsymbol{R}_{\theta,P_0}\boldsymbol{P}$ given the total power P_0 and a certain physical design of the micropin-fin heat sink θ. In order to get $\boldsymbol{R}_{\theta,P_0}$, we first feed a set of random distributed 3D power profiles with the same total power P_0 into the thermal model [3] and generate corresponding temperature profiles. Following this, $\boldsymbol{R}_{\theta,P_0}$ can be calculated by the method of least squares fitting. For each sample power map, we use the following equation to calculate the deviation of the temperature profile generated by our linear model ($\boldsymbol{T}_{linear}$) from that derived by the original thermal model ($\boldsymbol{T}_{original}$):

$$Temperature\ Deviation \ = \frac{||\boldsymbol{T}_{linear} - \boldsymbol{T}_{original}||_2}{max(\boldsymbol{T}_{original})} \tag{3.4}$$

The results show that for a certain physical design of heat sink($H = 300\mu m$, $S = 180\mu m$, $D = 100\mu m$, pumping power $= 1mW$) with 32 W total power dissipation, maximum temperature deviation between the two models is just 1.492×10^{-7}. In our research, we are especially interested in the peak temperature since it has significant impact on the thermal feasibility of 3D IC. Figure 3.4 illustrates the comparison of the peak temperature between our linear model and the original thermal model. The results demonstrate a perfect match of the peak temperature between the two thermal models. In conclusion, our linear model has high accuracy to approximate the original thermal model. This linear model (*i.e.* $\boldsymbol{T} = \boldsymbol{R}_{\theta,P_0}\boldsymbol{P}$) will be used in Section 3.2.2 to solve the first level co-design problem.

3.2.2 Proposed Method

In this section, we modify the HQBP framework introduced in Section 3.1 such that the new framework can be used to solve the co-design problem in this section. The overall design flow is illustrated in Figure 3.5.

Step 1: Determine the footprint area of the 3D IC

Note that our proposed technique follows the 2-step design flow (Section 2.5.1). The Layer partitioning step is the same as introduced in Section 2.5.1. After this, we need to determine the footprint area of the 3D IC. We assume the diameter and pitch of pins as well as the diameter of the TSVs are predetermined parameters.

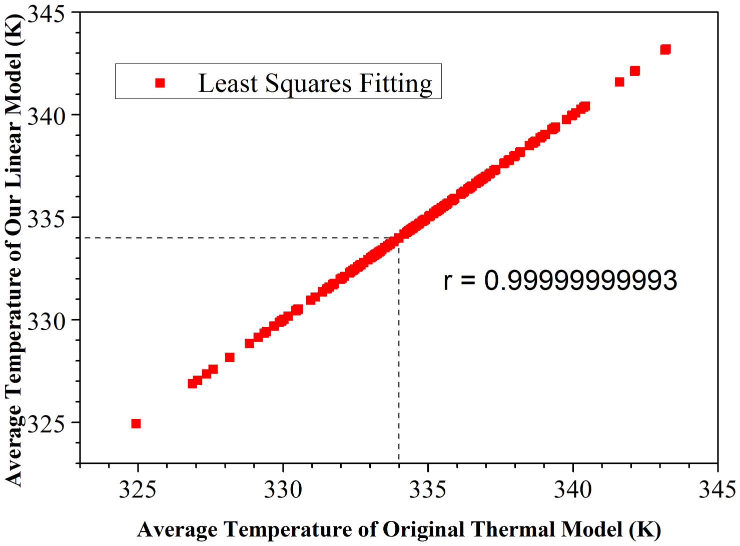


Figure 3.4: Comparison of the peak temperature between our linear model and the original thermal model [3] (r is the correlation coefficient)

Therefore, the number of TSVs that can be placed in a single pin is a constant (referred to as n_{tsv}). In this case, the footprint area for a chip is determined by the number and distribution of micropin-fins which should provide enough space to place TSVs.

In this work, we assume the micropin-fins are arranged in the staggered style with identical pin-pitches along the width and length direction (denoted as S). And we assume the micropin-fin structure on each layer is the same. There for, the upper bound of the number of TSVs that can be placed in each layer can be determined according to n_{tsv}:

First, we can calculate the upper bound of the number of TSV that can be placed in one layer according to n_{tsv} using Equation 3.5

$$U_{TSV} = n_{tsv} \times \text{Number of pins in one layer} \tag{3.5}$$

We also assume the pins are arranged in the staggered style and the pin pitch along the width and length directions of the chip are identical (denoted as S). As illustrated in Figure 3.5, we determine the heat sink structure after the layer assignment by ensuring that U_{TSV} is larger than the maximum number of TSVs in one layer in order to achieve a feasible design. We propose a heuristic algorithm to achieve this:

1. After layer partitioning of gates, the initial footprint area, A_{fp}, of the 3D IC can be determined using Equation 3.1 with an initial white-space ratio, α_W. We assume aspect ratio of the chip (*i.e.* W_{chip}/L_{chip}) is AR. This gives an initial size (in 2D) of 3D IC: $W_{chip} = \sqrt{A_{fp} \times AR}$ and $L_{chip} = \sqrt{A_{fp}/AR}$. Then we can calculate the the number of thermal grids in one layer is $[\frac{W_{chip}}{S}] \times [\frac{L_{chip}}{S}]$. Since the pins are arranged in the staggered style, a thermal grid contains a half pin on average (Figure 2.2). Without loss of generality and for the ease of locating pins, we suppose a pin to be located at the center of a thermal grid. Sometimes, there could be two possible patterns (as illustrated in Figure 3.6 showing the pattern with 5 × 5 thermal grids in one layer). In this case, we take the pattern with more pins for the sake of TSV bandwidth.

2. When the pin pattern is determined, we compare the current U_{TSV} with the maximum number of TSVs in one layer. If U_{TSV} is small, we increase α_W with a little step and go to step (1) to regenerate the pin pattern. This loop stops

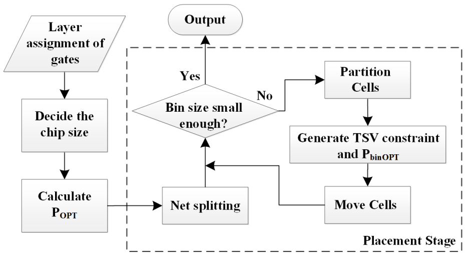
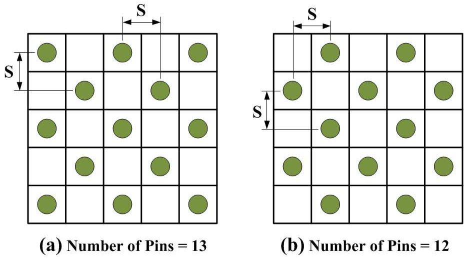


Figure 3.5: Flow chart of the proposed algorithm



(a) Number of Pins = 13

(b) Number of Pins = 12

Figure 3.6: Illustration of two possible pin-fin patterns for the chip with 5 × 5 thermal volumes in one layer

until there is enough pins accommodate for the TSVs in each layer. After this, the pin pattern as well as the footprint area of the 3D IC are fixed and we proceed to the in-layer placement stage for further process.

Step 2: Calculate the optimal power map and formulate the problem

Recall that one of the objectives in this problem is to minimize the in-layer temperature gradient. In this work, the optimization of the in-layer temperature gradient is achieved by minimizing ΔT:

$$\Delta T = \frac{1}{L}\sum_{l=1}^{L}(\max_{i\in l} T_i - \min_{i\in l} T_i), \tag{3.6}$$

where L is the total number of layers in the 3D IC. In Section 3.2.1, we have proposed a simplified linear model for the micropin-fin based MF cooling: $\boldsymbol{T} = \boldsymbol{R_{\theta,P_0}}\boldsymbol{P}$. Given this linear model, we will find the optimal power map $\boldsymbol{P_{OPT}}$ at the beginning of each partition level (Section 3.1.1) such that ΔT in Equation 3.6 is minimized. Formally stated, we will solve the following problem to get the $\boldsymbol{P_{OPT}}$:

$$\begin{aligned}
\min_{\boldsymbol{P}} \quad & \frac{1}{L}\sum_{l=1}^{L}(\max_{i\in l}(T_i) - \min_{i\in l}(T_i)) \\
\text{s.t.} \quad & \sum_{i\in l} P_i = P_l \qquad (l = 1, 2, ..., L) \\
& \sum_{l=1}^{L} P_l = P_0 \\
& T_i = \boldsymbol{r}^i_{\theta,P_0}\boldsymbol{P} \qquad \forall i \\
& P_i \geq 0 \qquad \forall i
\end{aligned} \tag{3.7}$$

In the above optimization problem, the variable is $\boldsymbol{P} = \{P_0, P_1, ..., P_n\}$ where n is the number of power grids. $\boldsymbol{r}^i_{\theta,P_0}$ is the i^{th} row vector of $\boldsymbol{R}_{\theta,P_0}$. L is the total number of layers in 3D IC, P_0 is the total power of the circuit and P_l is the total power of gates in the l^{th} layer. Both P_0 and P_l are constant. This problem can be reformulated as a linear programing problem and solved efficiently. The solution to the problem is $\boldsymbol{P}_{OPT}$. $\boldsymbol{P}_{OPT}$ is then used as the metric in the placement stage to minimize ΔT. Since the thermal model is linear and unchanged (i.e. $\boldsymbol{R}_{\theta,P_0}$ is fixed) during the placement stage, if the power map matches $\boldsymbol{P}_{OPT}$, ΔT of the temperature profile generated from this power map is minimized.

Given $\boldsymbol{P_{OPT}}$, the placement problem to be solved at each partition level can be formulated as follows:

$$\begin{aligned}
\text{variable:} \quad & x^l \\
\min \quad & WL(x^l) + \beta_1\text{Overlap}^l + \beta_2||\boldsymbol{P}(x^l) - \boldsymbol{P}^l_{OPT}||_2 \\
\text{s.t.} \quad & N^i_{TSV}(x^l) \leq U^i_{TSV} \qquad \forall i
\end{aligned} \tag{3.8}$$

Here, x^l is the variable representing the distribution of gates and TSV cells at the l partition level . $WL(x^l)$ and $\boldsymbol{P}(x^l)$ are the wire-length cost and power map of the distribution x^l, respectively. Overlap^l is the overlap cost for the l partition level which is calculated using Equation 3.2. β_1 and β_2 are the adjusting parameters. $N^i_{TSV}(x^l)$ is the number of TSVs within the i grid of the partition level l and U^i_{TSV} is the capacity to accommodate TSVs of that grid calculated according to the pin pattern generated previously.

Step 3: Calculate the Gain for each move

The HQBP framework requires the "gain" of moving each gate or TSV to be calculated before each iteration of the move (Section 3.1.1). The gain is calculated as the current cost minus the new cost after the move. According to Equation 3.8, the cost function is $WL(x^l) + \beta_1\text{Overlap}^l + \beta_2||\boldsymbol{P}(x^l) - \boldsymbol{P}^l_{OPT}||$, thus the gain is constituted of three entries: the gain of wire-length cost, the gain of overlap cost and the gain of the power distance. The calculation of the first two gains has been introduced in Section 3.1. The gain of the power can be efficiently calculated by comparing the cell (*i.e.* gate or TSV) power, current total power in the grid and the optimal power in the grid.

3.2.3 Data and Analysis

In order to evaluate our algorithm, we perform placement for 10 benchmarks from IWLS 2005 benchmark suits [56] on a three-tier stacked 3D IC with micropin-fin based MF cooling. For some benchmarks, we duplicate them to acquire large circuit designs. The information of each benchmark circuit is shown in Table 3.4.

Circuits	**Description**	**No. of Gates**	**#TSVs**	**Chip Size** (mm^2)	Q_{chip} **(W)**
Chip 1	wb_conmax	29,034	1071	3.31	13.6
Chip 2	ethernet	46,771	2378	5.88	26.8
Chip 3	4 dup. of pci_bridge32	67,264	897	5.80	33.8
Chip 4	4 dup. of ac97_ctrl	71,130	1385	7.58	32.4
Chip 5	2 dup. of b17	74,234	2739	4.18	25.0
Chip 6	6 dup. of usb_funct	76,848	1073	5.23	30.0
Chip 7	7 dup. of mem_ctrl	80,080	1211	5.15	28.9
Chip 8	b18	92,048	2350	4.92	24.1
Chip 9	des_perf	98,341	4743	5.88	32.0
Chip 10	5 dup. of aes_core	103,970	1970	5.88	27.7

Table 3.1: Circuits information (dup. = duplication)

In this table, "Chip Area" represents the total area of the 3D IC, Q_{chip} is the total power of the circuit, and "#TSVs" represents the total number of TSVs in the 3D IC after the layer partitioning step. Other parameters of the MF cooling heat sink is listed as follows: the height of pins $H = 300\mu m$, the diameter of pins $D = 100\mu m$, the pitch of pins $S = 180\mu m$ and the pumping power is $1mW$. The diameter of TSV is $8\mu m$. The maximum temperature limit $T_{max} = 85^oC$.

Our cooling-aware placement method is evaluated against the method that just minimizes the wire-length during placement without considering the position of pins. This baseline result is generated using the same hierarchical method as introduced above while the power distribution term is removed from the cost function. A TSV legalization procedure follows the this technique to re-allocate TSVs to their closest pins while maintaining the cell-overlapping constraint. For these

	PTPA				Co-placement Algorithm				
Name	WL (m)	max T	ΔT	max ∇T	WL (m)	max T	ΔT	max ∇T	RT(min)
Chip 1	0.79	83.4	28.8	43.8	0.97	60.0	3.1	17.3	51.29
Chip 2	3.20	89.5	40.8	27.6	3.28	68.5	3.7	12.0	215.3
Chip 3	3.13	82.3	29.0	30.1	4.00	74.0	5.9	17.8	87.1
Chip 4	3.94	88.0	30.9	55.9	4.06	73.1	10.0	19.6	79.4
Chip 5	3.42	92.2	25.1	44.1	4.20	72.6	6.4	13.5	345.8
Chip 6	4.50	102.1	22.4	50.9	5.08	82.7	6.0	13.5	138.3
Chip 7	4.10	96.4	20.2	48.0	4.95	80.5	5.9	13.1	148.1
Chip 8	5.00	88.0	23.8	45.8	5.57	78.7	3.4	12.7	514.6
Chip 9	6.26	82.4	28.7	19.5	5.80	69.5	7.1	11.6	157.8
Chip 10	5.67	83.8	29.0	47.8	7.31	62.5	6.4	11.6	124.5

Table 3.2: Experimental results of PTPA (Post-TSV-Allocation Placement Algorithm) and our co-placement algorithm

two methods, we compared the half-perimeter wire-length (WL), peak temperature (max T), ΔT (which is calculated by Equation 3.6) and the maximum temperature gradient (max ∇T) of the layout generated from this "Post-TSV-allocation Placement Algorithm" (PTPA) with those derived by our co-placement algorithm. The results are illustrated in Table 3.2. In Table 3.3, we calculate the improvement by using or method for each circuit. In the table, the unit of max T and ΔT is ^oC and the unit of max ∇T is $^oC/mm$. "RT" represents running time of our algorithm.

Compared to the baseline case, our cooling-aware placement technique has better temperature uniformity. According to Table 3.3, our method can achieve an average of 79.2% reduction in ΔT thus leading to smaller temperature gradient

	Overhead and Improvement			
Name	WL	max T	ΔT	max ∇T
Chip 1	22.8%	-28.0%	-89.3%	-60.5%
Chip 2	2.5%	-23.4%	-90.9%	-56.4%
Chip 3	27.8%	-10.0%	-79.8%	-41.0%
Chip 4	3.0%	-16.9%	-67.6%	-64.9%
Chip 5	22.4%	-21.2%	-74.4%	-69.4%
Chip 6	12.8%	-19.0%	-73.1%	-73.4%
Chip 7	20.7%	-16.5%	-70.8%	-72.7%
Chip 8	11.4%	-10.6%	-85.7%	-72.3%
Chip 9	-8.1%	-15.3%	-73.6%	-22.1%
Chip 10	28.9%	-25.4%	-78.0%	-75.8%
Average	13.0%	-18.7%	-79.2%	-64.6%

Table 3.3: Overhead and improvement by comparing the co-placement algorithm to the PTPA

in each layer (64.6% reduction on average) and lower peak temperature (18.7% reduction on average). In addition, our method can guarantee the design of 3D IC is thermally feasible while the peak temperature for some circuits placed by PTPA exceeds the limit. It should be noted that while we can always reduce peak temperature by increasing pumping power, this action introduces extra overhead and cannot improve the uniformity of temperature distribution. Therefore, in our experiment, the pumping power is fixed to $1mW$.

In order to improve the cooling efficiency, our proposed method may cause the increase in wire-length. However, as demonstrated by the result, such cost is not very high. In one case, the wire-length of the circuit placed by our method is even smaller than that placed by PTPA. This is due to the fact the TSV legalization procedure increases the wire-length by re-allocating TSVs. On average, the increase of wire-length generated from our algorithm compared to PTPA is 13.0%.

As for the run time of our algorithm, different circuit has different run time (RT) as shown in Table 3.2. In general, the run time is proportional to the input size which is characterized by the number of gates, the maximum size of a net and the number of layers.

3.3 2^{nd}-level Co-Design Problem: Cooling-Power-delivery Co-Placement

In this section, we propose a co-design method based on the HQBP framework (Section 3.1) to determine the placement of gates, signal TSVs, P/G TSVs as well as the allocation of micro-channels in 3D ASICs with micro-channel based MF cooling. This methodology is intended to handle all the interdependent relationships in the 3D IC during the physical design stage of MF cooled 3D ICs (as introduced in Section 2.4). In order to achieve this, we need to modify HQBP framework such that the design of micro-channels and 3D PDNs can be incorporated into this framework. The key challenges of this problem are:

- Micro-channels span across the whole length of the chip. Therefore, the local update process in the HQBP framework needs to be modified to accommodate the modification of positions of micro-channels.
- Since micro-channels are built through the inter-layer regions, it will conflict

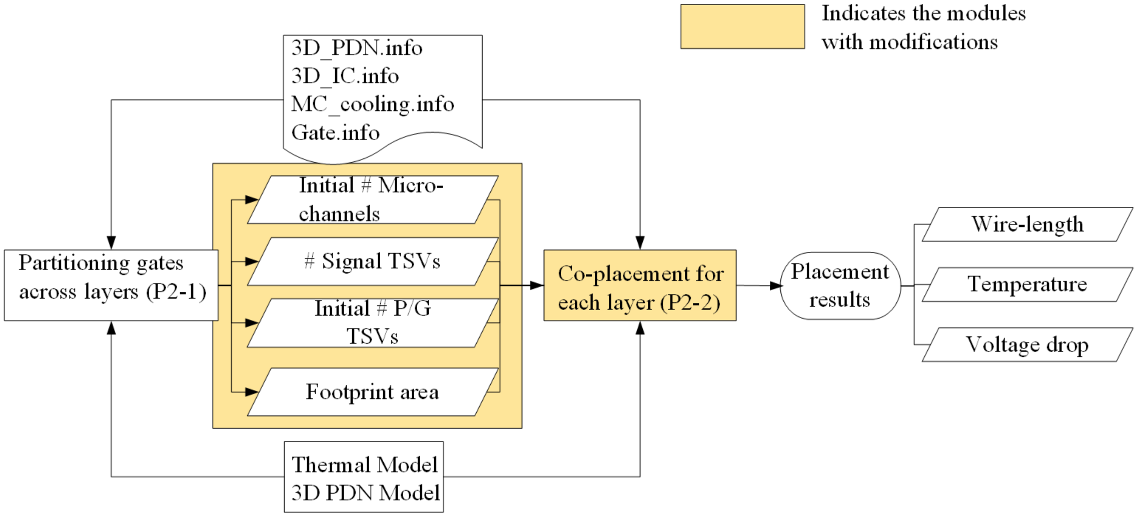


Figure 3.7: Illustration of the design flow to solve the second-level co-design problem based on the HQBP framework

with both signal and P/G TSVs. This interaction should be captured when modifying the position of any of the structures.

- Temperature and voltage calculation is very time consuming. We should find some efficient models to capture the influence of each move (of gates, signal TSVs, micro-channels and P/G TSVs) on the temperature and voltage.

The thermal and PDN model used in this project have been introduced in Section 2.1.1 and 2.2. The modification of the HQBP framework and the proposed methodology will be described in Section 3.3.1. In Section 3.4, we will show the experimental results.

3.3.1 Proposed Method

Figure 3.7 illustrates the design flow of the proposed technique. As illustrated in the figure, we will perform the following modifications to the HQBP framework:

1. Initialize the number of micro-channels and P/G TSVs for each layer after the layer assignment of gates (which is introduced in Section 3.1.1).

2. Enable the co-placement of micro-channels and P/G TSVs.

3. Calculate the gain (as defined in Section 3.1.1) induced by the temperature and voltage drop.

In the rest parts of this section, we will introduce the three modifications in detail.

3.3.1.1 Initialize the Number

Following the layer assignment of gates, we will determine the number of P/G TSVs and micro-channels on each tier successively (as illustrated in Figure 3.7). This is a complicated problem because on one hand, we would like to allocate sufficient P/G TSVs and micro-channels for power delivery and cooling respectively, while on the other hand, we hope the addition of these structures should have as small impacts on the 3D IC performance as possible. For instance, large number of P/G TSVs will compete with signal TSVs for space and cause routing congestion thus affecting the 3D IC performance; over-allocation of micro-channels will come in conflict with all types of TSVs thus causing significant performance degradation and even infeasible designs. Therefore, on this stage, we would like to calculate a reasonable number of P/G TSVs and micro-channels. However, the main challenge is although the assignment of gates and the number of signal TSVs on each tier is already known up to this step, their positions are not yet specified. Therefore, we cannot have an accurate estimation of whether a certain number and distribution of P/G TSVs would achieve **just** sufficient power supply. Similarly, the conflict between TSVs and a certain allocation of micro-channels cannot be estimated accurately either. Hence at this step, we provide a greedy algorithm to roughly estimate the number of P/G TSVs and micro-channels, successively. The number of P/G TSVs and micro-channels would be further refined in the "in-layer placement" stage (Section 3.1.1 and the modifications to this part will be introduced later) where the actual gate, TSV and micro-channel location would be determined. The initialization of the number of micro-channels follow the similar procedure. In the following, we will introduce the algorithm to determine the number of P/G TSVs:

- **Initialization:** The number of P/G TSVs allocated to each tier is initialized as the maximum possible number of P/G TSVs on the tier, which is denoted as N_{PGT}^{max}. N_{PGT}^{max} is decided by the predefined minimum pitch of P/G TSVs and the footprint area of the 3D IC (determined after the partitioning of gates across layers).

- **Computing the Initial ΔV_{max}:** Here, ΔV_{max} represents the maximum voltage drop across the 3D IC. After initializing the number of P/G TSVs, we can evaluate ΔV_{max} of the chip using the PDN model introduced in Section 2.2. Currently the P/G TSVs are distributed uniformly as introduced in the previous step. Therefore, construct the 3D PDN by uniformly adding resistors to connect the power grid points on the adjacent layers. The power profile of the 3D IC ($\boldsymbol{P}$) is determined by assuming the power on each tier to be distributed uniformly and the current load is calculated through $\boldsymbol{I} = \boldsymbol{P}/V_{dd}^0$, where V_{dd}^0 is the source supply voltage. The current load is then fed into the 3D PDN to calculate the real supply voltage, $\boldsymbol{V_{dd}}$, on each grid point in the 3D PDN. Then the initial maximum voltage drop, $\Delta V_{max} = \max(V_{dd}^0 - \boldsymbol{V_{dd}})$.

- **Removing Redundant P/G TSVs:** The maximum voltage drop across the chip (ΔV_{max}) generated by allocating maximum possible number of P/G TSVs to each tier should be as low as possible (*i.e.* lower than the preset voltage drop limit, ΔV_{limit}). If the condition is violated, the design is infeasible and we should return to the layer partitioning level and allocate fewer gates to the top layers. If there is a slack between the estimated voltage drop and the limit, however, we will remove the redundant P/G TSVs. To achieve this, we gradually increase the pitch between P/G TSVs on each layer (and keep each layer to contain identical number and distribution of P/G TSVs) until ΔV_{max} is just lower than ΔV_{limit}. The number of P/G TSVs on each layer is thus determined.

Following this, we get the tier-assignment of each gate as well as the initial number of P/G TSVs and micro-channels between the adjacent tiers.

3.3.1.2 Modify the Cost Function

In order to enable the co-placement of micro-channels and P/G TSVs, we need to modify the "in-layer placement" stage (Section 3.1.1). Meanwhile, the cost function of the problem should also be modified to capture the impacts of temperature and voltage drop. The modified in-layer placement flow is stated as follows:

The process is started by randomly partitioning all gates and signal TSVs in each layer into four grids. P/G TSVs are uniformly partitioned into the four grids and micro-channels are uniformly bi-partitioned to the associated columns. As illustrated in Figure 3.8(II), a column constitutes the set of grids along the y direction in which the micro-channel spans. For instance, the Column 1 in Figure 3.8(II) contains two grids: g_0 and g_1. The process described above is known as the "first-level partition". Following this, we will process the first-level partition. By processing, we imply moving gates, signal TSVs and P/G TSVs across grids and moving micro-channels across columns to minimize the following cost:

$$Cost = \alpha_{WL}C_{WL} + \alpha_{A}Overlap + \alpha_{T}T_{max} + \alpha_{\Delta V}\Delta V_{max}. \tag{3.9}$$

In this equation, C_{WL} represents the wire-length cost which is introduced in Section 3.1.2; *Overlap* represents the overlap cost which equals to the sum of all types of overlap (including gate-gate, gate-TSV, TSV-channel *etc.*) in all grids under

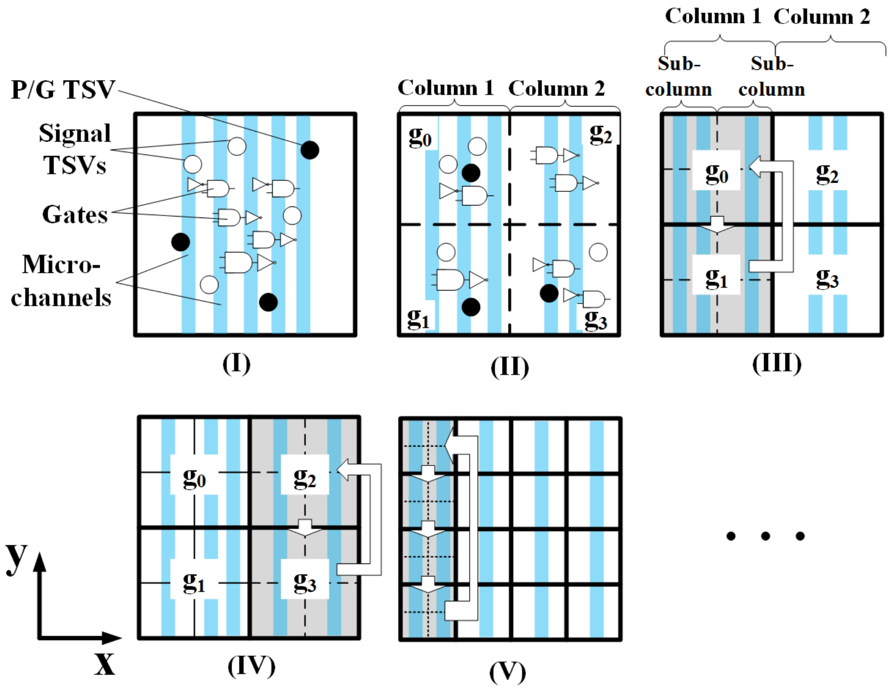


Figure 3.8: Illustration of the evolution of the "in-layer placement" stage after modifications by showing the first partition level to the third partition level (I, II represent the first partition level; III, IV represent the process of the second partition level; V illustrates the process of the third partition level. The shaded column indicates the currently processed column. The white arrows indicate the order of grids to be processed)

processing and is calculated as follows:

$$\begin{aligned} C_A = \sum_{l=1}^{L} (\sum_i \max(0, A_{g,i} + A_{TSV,i} - \hat{A}) \\ + \sum_i \max(0, A_{TSV,i} + A_{ch,i} - \hat{A})). \end{aligned} \tag{3.10}$$

Here, $A_{g,i}$ is the total gate area within grid i; $A_{TSV,i}$ is the total area of signal and P/G TSVs within grid i; $A_{ch,i}$ is the total area of micro-channels intersected with the grid i; $\hat{A}$ is the area capacity of grid i. Note that Equation 3.10 is the extension of Equation 3.2 by considering the spacial conflict between micro-channels and TSVs. The process of a group of four grids is similar to the method introduced in Section 3.1.1. However, in this problem, we need to deal with two more factors: T_{max} and ΔV_{max}. T_{max} and ΔV_{max} are the peak temperature and the maximum voltage drop of the **whole chip**, respectively. T_{max} and ΔV_{max} can be estimated using the models introduced in Section 2.1.1 and 2.2. Note that, in order to use the 3D PDN model, we uniformly distribute the P/G TSVs in each grid thus forming the 3D PDN netowrk. As introduced in Section 3.1.1, before each iteration of moving gates, TSVs or micro-channels, we need to calculate the "gain" of each potential move. The calculation of the gain induced by wire-length and overlap has been described in Section 3.1.1 and we will introduce a method to calculate the gain of T_{max} and ΔV_{max} efficiently (Section 3.3.1.3).

When the *Cost* cannot be further reduced, the process is stopped and we have each layer of the 3D IC partitioned into four grids with appropriate cells (*i.e.* gates and signal TSVs), micro-channels and P/G TSVs as illustrated in Figure 3.8(II). If the peak temperature (or the maximum voltage drop) decreases after the process, we removes micro-channels (or P/G TSVs) while the thermal (or voltage drop) constraint is still maintained. If either the number of micro-channels or the P/G TSVs changes, we process the first-level partition for another round. On the other hand, if the peak temperature (or the maximum voltage drop) increases, we discard the result for this round of process. This is because the current design is infeasible (with respect to temperature or voltage drop). Then we start from the result for the last round of process (which is feasible), increase the weight for the temperature (or voltage drop) in Equation 3.9 and perform another round of process. When a design is feasible and the number of micro-channels and P/G TSVs does not change the algorithm continues processing by further partitioning (Figure 3.8(III)-(V)).

During each of the following hierarchical partition levels, we first process the grids constituting the same column and then come to the grids of the next column. As illustrated in Figure 3.8(III)-(IV), we first process column 1 (including grid g_1 and g_2) and then process column 2 (including grid g_3 and g_4). Before processing the grids of a certain column, the algorithm uniformly partitions the micro-channels passing through the column into two sub-columns (Figure 3.8(III)) and uses the number of channels in each sub-column to represent the initial distribution of micro-channels, which is denoted as MC_{ini}. Following the partitioning of micro-channels, each grid of that column is processed successively. As illustrated in Figure 3.8(III), in Column 1, grid g_2 is processed after g_1. For each grid under processing, the algorithm first **randomly** partitions the cells (including gates and signal TSVs) of that grid into four sub-grids and **uniformly** partitions the P/G TSVs of that grid into four sub-grids. Following this, the current peak temperature and maximum voltage drop across the 3D IC are evaluated using the models introduced above. The gates, TSVs and micro-channels are then moved to reduce the *Cost* within the currently processed grid. The form of *Cost* is similar to that in Equation 3.9 with T_{max} and ΔV_{max} representing the peak temperature and the maximum voltage drop of the **whole** chip, respectively. However, C_{WL} and C_A now represent the wire-length cost and overlap cost within the processed grid rather than the whole chip. This point has been covered in Section 3.1.2. Note that micro-channels within the column can be moved between the sub-columns during the processing of **each** grids in the column. However, this action may increase the overlap cost of the already processed grids within the same column. Therefore, if MC_{new} (*i.e.* the distribution of micro-channels after processing all the grids within the column) is not equal to MC_{ini}, the current column is processed again (*i.e.* all the grids in the column will be processed again) as illustrated by the "U" shape arrow in Figure 3.8(III). Since the algorithm tends to reduce the peak temperature of the whole chip when processing grids, we can finally achieve $MC_{ini} = MC_{new}$ after a finite number of iterations of processing the column. After this, the algorithm proceeds to process the grids in the next column (as illustrated in Figure 3.8(IV)). When all the grids have been processed, the algorithm checks if the number of micro-channels (or P/G TSVs) can be reduced while the thermal (or voltage drop) constraint is still maintained. If the number of either structure is changed, the current partition level is processed again, otherwise the algorithm proceeds to the next partition level (as illustrated in Figure 3.8(V)). The algorithm stops when the preset maximum number of partition levels is reached. It should be noted that when the size of the partitioning grid is smaller than the thermal grid (or the power grid), the position of micro-channels (or P/G TSVs) will not be changed while the position of gates and signal TSVs can be further updated.

3.3.1.3 Calculate the Gain Induced by the Temperature and Voltage Drop

Before moving cells in each iteration, our algorithm calculates the *Gain* of each cell which is the sum of four types of gains: G_{WL}, G_A, G_T and $G_{\Delta V}$. The calculation of G_T and $G_{\Delta V}$ can be very time consuming since, naively, we need to perform one matrix multiplication for each move of each cell. In this section, we will introduce an approximate yet efficient method to calculate G_T and $G_{\Delta V}$. First of all, First of all, we quantize G_T (or $G_{\Delta V}$) to three discrete values: -1, 0, 1 to represent three states. If G_T (or $G_{\Delta V}$) equals 1, the peak temperature (or the maximum voltage drop) decreases due to the move of the cell (*i.e.* gates and signal TSVs). On the other hand, if G_T (or $G_{\Delta V}$) equals -1, the peak temperature (or the maximum voltage drop) increases. If the move of the cell will not induce significant change to the peak temperature (or the maximum voltage drop), G_T (or $G_{\Delta V}$) equals 0. In the following, we will introduce a method to calculate the modified G_T, while the calculation of $G_{\Delta V}$ follows the same procedure.

Motivation

According to Section 2.1.1, $\boldsymbol{T} = \boldsymbol{RP}$. Both the change of $\boldsymbol{R}$ and $\boldsymbol{P}$ will change $\boldsymbol{T}$. During each iteration, moving a signal TSV will change $\boldsymbol{R}$ while moving a gate will change $\boldsymbol{P}$. Since the total number of signal TSVs in a 3D IC is usually very large (over several thousands), the change of $\boldsymbol{R}$ induced by moving several signal TSVs is negligibly small. This enables us to assume $\boldsymbol{R}$ to be constant for a number of iterations of moving cells. Actually, our algorithm will modify $\boldsymbol{R}$ after the positions of a large number of signal TSVs have been changed. Based on this assumption, $\boldsymbol{T}$ is a linear function of $\boldsymbol{P}$ which can only be changed by moving gates. Suppose a gate with power Δp is moved from thermal grid i to j. The temperature in thermal grid k (*i.e.* the average temperature in the thermal grid) will change by $\Delta T_k = (R_{kj} - R_{ki})\Delta p$. Here, $(R_{kj} - R_{ki})$ is a constant for all k. Therefore, we can determine whether the peak temperature of the chip is increased or not by simply comparing the gate power (*i.e.* Δp) with some pre-calculated values rather than performing the time-consuming matrix multiplication. The calculation of such values will be introduced in the next paragraph.

Preprocessing

At the beginning of each partition level, we will determine for each pair of grids (*e.g.* grid i and grid j) the threshold value of δP such that moving δP from grid i to j leads to the increase in peak temperature. Suppose the peak temperature of the 3D IC (denoted as T^0_{max}) is currently discovered in grid k. The procedure is stated as follows:

- For each pair of grids (*e.g.* from grid i to j), check $R_{kj} - R_{ki}$. If $R_{kj} - R_{ki} > 0$, moving any gates (with positive power) from i to j will cause the increase of peak temperature.

- if $R_{kj} - R_{ki} \leq 0$, moving any gates from i to j will not increase the peak temperature at grid k. However, this action may cause the temperature in other locations increase which may still increase the peak temperature of the chip. Therefore, we need to find the threshold value for δP moved from grid i

to j which makes the temperature in some other grids increase to reach T^0_{max}. This threshold value is denoted as P^{TH}_{ij}. If the power of a gate in grid i is less than P^{TH}_{ij}, moving it to j will not increase the peak temperature. When this condition is held, if $R_{kj} - R_{ki} < 0$, the peak temperature will be reduced while the peak temperature is regarded as unchanged if $R_{kj} - R_{ki} = 0$.

- For each pair of grids (*e.g.* from grid i to j), we record the sign of $R_{kj} - R_{ki}$ as well as P^{TH}_{ij} if there is one. These values form the criteria for determining G_T in the future.

Determining G_T During the Run-time

During each iteration of move in the "in-layer placement" stage (Section 3.1.1), each cell (*i.e.* gate or signal TSV) can be possibly moved to other three grids. The G_T for signal TSVs is 0. For each gate, we traverse all its possible moves and calculate gain for each move according to the pre-generated criteria. If a move increases the peak temperature, the gain is -1; if a move decreases the peak temperature, the gain is 1; otherwise the gain is 0. The G_T of a cell is the largest gain among all the possible moves. This gain, together with the destination grid of the corresponding move, will be stored.

Updating Criteria

The method stated above is valid only when the following two conditions are satisfied: (1) $\boldsymbol{R}$ is almost constant and (2) the temperature profile does not change too much. In order to meet these requirements, our algorithm modifies the criteria when either a large amount of signal TSVs change their positions or the change

Bechmarks	#Gates	#s-TSVs	PD (W/cm^2)	Chip Size (mm)
usb_funct	13K	5K	675	0.63
pci_bridge32	17K	3K	714	0.70
aes_core	21K	1K	334	0.55
b22	28K	14K	512	0.88
wb_conmax	30K	4K	535	0.70
ethernet	47K	2K	234	1.30
RISC	60K	10K	474	1.20
b18	92K	14K	706	1.35
des_perf	98K	10K	257	1.33
vga_lcd	124K	24K	586	1.72

Table 3.4: Benchmark information (**s-TSVs** indicates the signal TSVs; **PD** indicates the power density of the chip)

of peak temperature exceeds a threshold value. According to our simulation, the modification only takes place every several tens of iterations.

3.4 Experimental Results

In this section, we will show the experimental results for the second-level problem. We first introduce the setup of the experiment in Section 3.4.1. Then the data and analysis will be presented in Section 3.4.2.

3.4.1 Setup of The Experiment

The algorithm is tested with benchmarks from IWLS2005 benchmark suit [56] on a three-tier stacked 3D IC. The benchmark circuits are synthesized using 45nm

Micro-channel dimension	$50\mu m \times 100\mu m$ (width × height)
Micro-channel wall sickness	$50\mu m$
Fluid velocity	$5m/s$
Inlet flow temperature	$25°C$
Ambient temperature	$25°C$
Signal TSV diameter	$10\mu m$
P/G TSV diameter	$20\mu m$
Power grid size	$100\mu m \times 100\mu m$
Source voltage (V_{dd}^0)	1 V

Table 3.5: Parameters used in the simulation

technology library. Since the benchmark circuits do not have the information indicating the power consumption, such as the current load or the gate-level switching activities, we generate power profiles using the following procedure: For each gate, its power density is randomly generated between $10W/cm^2$ and $500W/cm^2$ and its power is computed by multiplying the gate area (acquired from the technology library) with the generated power density. The total power density of a benchmark implemented in the 3D IC is computed by dividing the total power of all gates by the footprint area of the 3D chip. The whitespace ratio for each circuit is 20%. And without loss of generality, we assume the shape o the chip is square. The number of gates, signal TSVs, chip size and the power density for each benchmark used in the experiment are shown in Table 3.4. Some other parameters used in the simulation are shown in Table 3.5. For the 3D PDN model, the resistance of P/G TSVs is calculated according to [14] while other parameters are taken from [38]. The temperature limit is 85°C and the voltage drop limit is 10% of the source voltage.

For each benchmark, we first use hMETIS [52] to partition the netlist and assign gates to the layers of the 3D IC. Afterwards, four different techniques are studied: AC-WL-driven, MC-WL-driven, Cooling-Aware and T-V-Aware.

- **AC-WL-driven Method.** This represents the method of Wire-length-driven placement method [57] with air-cooling. The objective of the wire-length-driven method is only to reduce the wire-length while keeping the overlaps between gates and signal TSVs lower enough. After the placement, air-cooling scheme is applied to cool the 3D IC and the 3D PDN is designed by uniformly allocating P/G TSVs.

- **MC-WL-driven Method.** This represents the method of Wire-length-driven placement method with micro-channel based MF cooling. After the wire-length-driven placement [57], this method adds micro-channels to cool the chip and design the 3D PDN by uniformly allocating P/G TSVs. The micro-channels are allocated following the similar approach in [28]. The overlaps introduced by adding these new structures will be eliminated by redistributing gates and signal TSVs.

- **Cooling-Aware Method.** After partitioning gates across layers, we follow the procedure introduced in Section 3.3.1 to determine the number of micro-channels of each layer in the 3D IC, thus solving **P1** as described in Section 3.3.1. Then, during the placement stage, the algorithm places gates, signal TSVs and micro-channels simultaneously to co-optimize the wire-length and temperature with as fewer micro-channels as possible. The overlaps among gates, signal TSVs and micro-channels are also controlled. When the placement is done, the P/G TSVs are added uniformly and the new overlaps are eliminated by redistributing gates and signal TSVs.

- **T-V-Aware Method.** T-V-Aware is the short form of Thermal-VoltageDrop-Aware. This method handles all the interdependent relationships as introduced in Section 2.4, such that the micro-channel heat sink and P/G TSVs are designed simultaneously with the placement of gates and signal TSVs. Readers may refer to Section 3.3.1 for details of this method.

The results of all benchmarks when applying the four methods are shown in Table 3.2. Note that, except for "AC-WL-driven" method, all the other three methods use micro-channels to cool the chip. For each benchmark, the number of micro-channels used for cooling for the three methods is **identical** (thus the pumping power is the same). Similarly, the same number of P/G TSVs are used for all the methods. In the table, the average wire-length across all benchmarks for each method is reported. As for the "Average for T_{max}" and "Average for ΔV_{limit}" for each method, they are calculated using the following equations, respectively:

$$\text{Average for } T_{max} = \frac{1}{N}\sum_{i=1}^{N}(\max(0, T_{max,i} - T_{limit})), \tag{3.11}$$

$$\text{Average for } \Delta V_{max} = \frac{1}{N}\sum_{i=1}^{N}(\max(0, \Delta V_{max,i} - \Delta V_{limit})). \tag{3.12}$$

Here, N is the total number of benchmarks; $T_{max,i}$ and $\Delta V_{max,i}$ are the peak temperature and maximum voltage drop for the i^{th} benchmark, respectively. T_{limit} is the temperature limit where $T_{limit} = 85°C$ and ΔV_{limit} is the maximum voltage drop margin where $\Delta V_{limit} = 100mV$.

3.4.2 Results and Analysis

To begin with, we will show the benefits of using MF cooling in 3D ICs by comparing the results of air-cooling and MF cooling applied to the WL-driven method. The results are shown in Table 3.6. As illustrated by the data, air-cooling alone is far less sufficient to remove heat from the 3D IC thus resulting in very high peak temperature (with an average of $148.6°C$ exceeding T_{limit}). On the other hand, with micro-channel cooling, the peak temperature is significantly reduced. The cost paid for this method is the wire-length increase due to the redistribution of gates and signal TSVs. The average increase in wire-length is roughly 11% (Table 3.6). Although the cost is not large, this method (*i.e.* post allocating micro-channels after the WL-driven placement) still cannot guarantee thermally feasible designs with a small number of micro-channels. As illustrated in Table 3.6, over half of the benchmarks result in thermal violation even with micro-channel cooling. The average peak temperature violation is $6.5°C$ compared to the temperature limit, $T_{limit} = 85°C$. Similarly, post-designing P/G TSVs may not guarantee a safe voltage drop either (with an average of $7.3mV$ violation when the voltage drop margin is $100mV$). Note that, benchmarks with violated peak temperature or maximum voltage drop are marked with "(-)" in Table 3.6.

The thermal problem due to post-allocating micro-channels can be solved by the second method: Cooling-Aware method with co-placement of gates, signal TSVs and micro-channels (shown in "Cooling-Aware" column in Table 3.6). However, when we turn to the voltage drop, we find that in most cases, the voltage drop is worsened compared to WL-driven method. The average voltage drop violation of the Cooling-Aware method is 15.7mV. This situation is caused by the way the Cooling-Aware method works. Due to the degradation of cooling capability downstream the micro-channels, the Cooling-Aware method tends to place gates with higher power close to the inlet of the channel while allocating fewer gates at the outlet. This might result in significantly non-uniform power profile which adds the load burden to the 3D IC with uniformly distributed P/G TSVs. Therefore, the voltage drop becomes worse.

In order to fix the thermal problem with micro-channel cooling **as well as** maintaining the voltage drop within the margin, we propose the T-V-Aware placement method as introduced in Section 3.3.1. The results are shown in the "T-V-Aware" column in Table 3.6. As illustrated by the results, T-V-Aware placement method is able to achieve thermally feasible design while maintaining the voltage drop within the margin for all benchmarks. For some benchmarks, the reduction of the maximum voltage drop compared to the Cooling-Aware method can reach up to 40%. The wire-length of the Cooling-Aware method and the T-V-Aware method is roughly the same, with an average of 10% increase compared to the WL-driven method with micro-channel cooling.

Benchmark	AC-WL-driven			MC-WL-driven			Cooling-Aware			T-V-Aware		
	WL	T_{max}	ΔV_{max}	WL	T_{max}	ΔV_{max}	WL	T_{max}	ΔV_{max}	WL	T_{max}	ΔV_{max}
	(m)	($^\circ C$)	(mV)	(m)	($^\circ C$)	(mV)	(m)	($^\circ C$)	(mV)	(m)	($^\circ C$)	(mV)
usb_funct	0.3	124.1 (-)	108.0 (-)	0.3	92.0 (-)	108.0 (-)	0.3	84.9	132.5 (-)	0.3	84.6	92.3
pci_bridge32	0.3	151.8 (-)	102.1 (-)	0.4	85.8 (-)	102.1 (-)	0.3	84.7	114.1 (-)	0.3	83.2	97.4
aes_core	1.5	288.5 (-)	83.1	1.5	96.9 (-)	83.1	1.5	84.8	79.1	1.5	82.4	69.5
b22	1.0	210.0 (-)	125.3 (-)	1.5	92.5 (-)	125.3 (-)	1.1	84.3	119.8 (-)	1.1	83.9	97.8
wb_conmax	0.6	137.4 (-)	137.1 (-)	0.6	83.0	137.1 (-)	0.7	81.3	122.0 (-)	0.7	84.0	98.9
ethernet	1.6	384.6 (-)	88.9	1.6	94.3 (-)	88.9	1.8	81.6	106.8 (-)	1.8	84.1	89.5
RISC	1.7	272.2 (-)	91.6	2.1	92.4 (-)	91.6	2.6	84.2	100.6 (-)	2.7	82.1	90.3
b18	3.3	259.4 (-)	94.5	4.0	90.1 (-)	94.5	4.1	81.8	144.3 (-)	3.8	84.5	95.2
des_perf	1.1	293.7 (-)	83.2	1.1	91.2 (-)	83.2	1.2	84.8	79.1	1.2	83.9	71.4
vga_lcd	6.3	214.7 (-)	93.1	6.8	95.1 (-)	93.1	8.5	83.3	117.1 (-)	8.3	85.0	93.9
Average	1.8	148.6	7.3	2.0	6.5	7.3	2.2	0	15.7	2.2	0	0

Table 3.6: Experimental results (**Average for WL** illustrates the average value of wire-length among all benchmarks; **Average for T_{max}** is calculated using Equation 3.11; **Average for ΔV_{max}** is calculated using Equation 3.12)

3.5 Conclusion

In this Chapter, we propose a hierarchical quadri-partitioning based placement framework to handle the trade-offs during the physical design stage of 3D ASICs with MF cooling (as introduced in Section 2.4). Our framework is first used to perform cooling aware placement of gates and signal TSVs with the existence of micropin-fin based MF cooling (1^{st}-level Co-design Problem) and it can achieve significant reduction of the in-layer temperature gradient compared to the traditional placement method. We also extend the framework to co-design the gate-level floor plan, the assignment of micro-channels and the allocation of P/G TSVs for 3D ASICs with micro-channel based MF cooling (2^{nd}-level Co-design Problem). Compared to the traditional sequential physical design flow (*i.e.* placing gates and signal TSVs, allocation micro-channels and designing the 3D PDN are performed in the successive order), our proposed technique is able to guarantee the feasibility of the design (*i.e.* no conflicting between TSVs and micro-channels, no violation of temperature *etc.*) with a little increase in wire-length.

Chapter 4: Co-Design Methodologies for 3D FPGAs

In this chapter, we present a framework to explore the design space of 3D FPGAs with micro-channel based MF cooling. Using this framework, we study the impacts of stacked layers and micro-channel density on the cooling and performance of 3D FPGAs. We also propose guidelines for designing 3D FPGAs with micro-channel cooling based on the experimental results for our proposed framework.

This chapter is organized as follows: Section 4.1 gives a brief review of 3D FPGAs and the existing placement and routing (P&R) tools for 3D FPGAs. Although 3D FPGAs have the same challenges as introduced in Section 2.4, its special structure will bring new problems to the physical design. The motivations of this project is also introduced in this section. Section 4.2 introduces the design-space exploration framework including the modeling of area, delay, power and temperature. In Section 4.4, experimental results are discussed. Finally, Section 4.5 concludes this chapter.

4.1 Background and Motivations

4.1.1 3D FPGAs

A Field-Programmable-Gate-Array (FPGA) is an integrated circuit which enables the customer to reconfigure the functionality of the circuit after manufacturing. Therefore, compared to standard cell based ASICs, FPGAs enjoys higher reusability, simpler design cycle and faster time-to-market. Nowadays, FPGAs are widely used in SoC chips with CPUs or GPGPUs to accelerate certain computation (*e.g.* neural networks [58,59] *etc.*). Meanwhile, the reconfigurable units in FPGAs cause tremendous programming overheads in FPGAs. According to [60], programmable elements can account for 90% total footprint area, 80% total path delay and large portion of power consumption. These overheads make FPGAs area-inefficient compared to ASICs and this gap increases with the continuous scaling of technology nodes. The logic density of FPGAs can be significantly improved by using 3D integration technology. This technology stacks multiple dies on top of each other and uses TSVs for inter-die connection which achieves shorter global wire-length.

Before introducing the 3D FPGA, we would like to describe the two basic categories of FPGAs classified according to different routing architectures: hierarchical style and island style (illustrated in Figure 4.1). In the hierarchical style FPGA, logic blocks are recursively connected to form a hierarchical structure (Figure 4.1(a)). Connections between logic blocks within the same cluster are made by wire segments at the lowest level of hierarchy while the connection between blocks

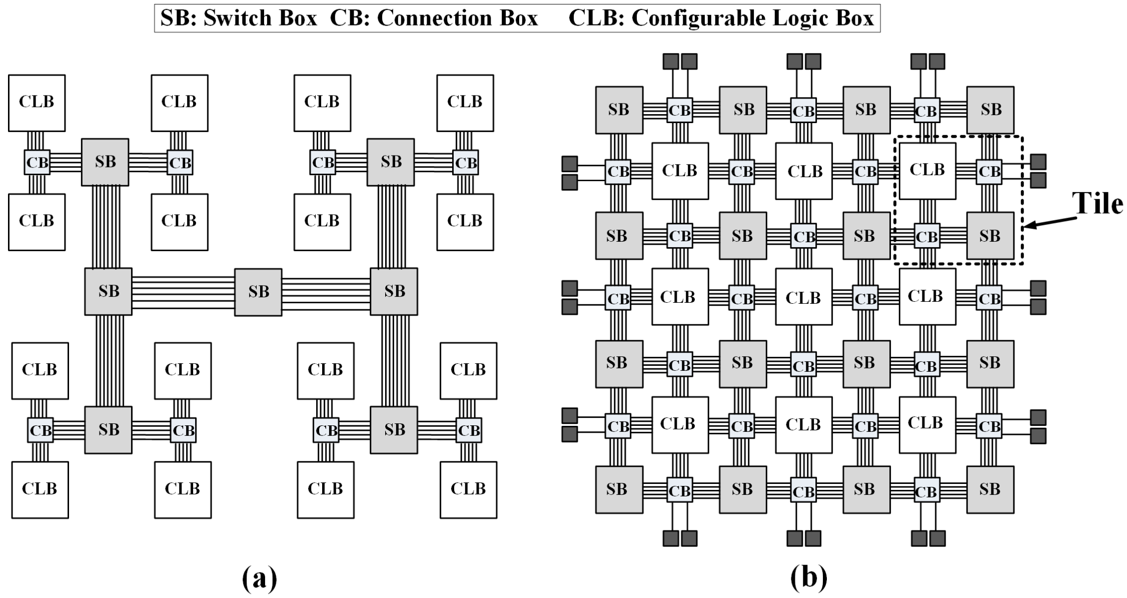

SB: Switch Box CB: Connection Box CLB: Configurable Logic Box
CLB
CB
SB
Tile
(a)
(b)

Figure 4.1: Illustration of the (a) hierarchical style and (b) island style FPGA

residing in different groups require the traversal of one or more levels of hierarchy. This routing style utilizes the locality of connections in FPGAs and has been used in several commercial FPGA families including Altera Flex 10K, Apex *etc.* . Another type of FPGAs is island style FPGA. In this architecture, configurable logic blocks (CLBs) and I/O blocks are arranged in a two dimensional mesh and routing fabrics, including connection boxes (CBs), switch boxes (SBs) and routing channels, are evenly distributed throughout the mesh (Figure 4.1(b)). As illustrated in the figure, the island-style FPGA has a regular pattern and can be regarded as the construction of identical tiles [61]. Each tile contains a CLB with its adjacent routing fabrics. Due to the regular pattern, island-style FPGAs exhibit a number of desirable properties including efficient connect between CLBs and routing tracks and high scalability. These properties make island-style the most commonly used architecture in contemporary commercial FPGA families including Stratix from Altera and Vertex from Xilinx. Due to the regular architecture, the island-style FPGA is usually used as the baseline to build 3D FPGAs.

As for stacking FPGAs in 3D, there are generally two different topologies:

1. The first one is developed using monolithic 3D ICs [60] where the computational units (such as configurable logic blocks) are separated from reconfigurable units (such as switch boxes *etc.*) and they are placed on different layers. This type of 3D FPGAs is illustrated in Figure 4.2(a). This type of 3D FPGAs can fully exploit the benefits of 3D integration with high density, low latency of TSVs. However, one problem with this topology is that it has low scalability if we would like to have several layers of logic blocks.

2. The second one develops the 3D FPGA by extending the switch boxes in traditional 2D FPGAs into 3D ones by adding TSVs to the switch boxes. And several 2D FPGA chips are bounded together using the 3D switch boxes. This type of 3D FPGAs is illustrated in Figure 4.2(b). Compared to the former type, this type of 3D FPGAs is easy to fabricate and the traditional P&R tools can be easily extended for designing this kind of 3D FPGAs. The extension of the traditional 2D switch boxes to 3D switch boxes will be described in Section 4.2.1.

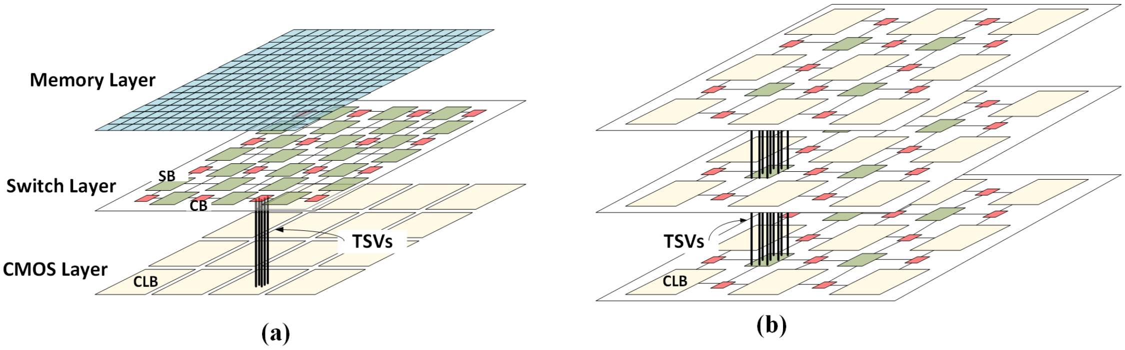


Figure 4.2: Illustration of the (a) monolithic 3D FPGA and (b) the chip bounded 3D FPGA

4.1.2 Design Tools for FPGAs

The design of FPGAs follows similar flow as ASICs. Generally, the design flow of FPGAs includes the following steps: (1) RTL design, (2) technology mapping, (3) clustering, (4) placement and (5) routing. Among these steps, RTL design is usually achieved using VHDL or verilog language and technology mapping (usually known as logic synthesis) selects certain gates to achieve the logic functions of the circuit. Clustering is a special step in FPGA design which clusters gates and maps the map the clusters to CLBs in FPGAs. A CLB is basically a hardware implementation (using look up tables) of a truth table or several truth tables. T-Vpack (which is a package of a 2D FPGA P&R tool, VPR [62]) is one of the popular open source tool for clustering. Following this, the physical positions of the clustered CLBs are determined during the placement step and the circuit is finally routed in the last step. Usually, the placement and routing of FPGAs are achieved in a single tool which is known as P&R tool. P&R tools are the kernel of the FPGA design since it has great impacts on the performance and power of FPGAs. Currently, P&R tools can be categorized into commercial tools and academic tools:

1. **Commercial Tools.** This kind of P&R tools include Quartus II from Altera, ISE and Vivado from Xilinx *etc.* . These tools are developed by certain companies and provide optimization of implementing circuits on the FPGA of their companies. The drawbacks of these tools are as follows: (1) they are inflexible in exploring the design space of FPGAs; (2) only 2D design tools are available since there are currently no true 3D FPGAs in the market; and (3) they are usually not free to be accessed.

2. **Academic Tools.** The most common P&R tool of this kind is VPR [62] developed by a group in University of Toronto. This tool targets 2D FPGAs and supports FPGAs with IP modules (*e.g.* DSPs) in the most recent version. TPR [63] is an extension of VPR which performs placement and routing for 3D FPGAs stacked following the second topology (as introduced in Section 4.1.1). Other popular P&R tools for 3D FPGAs include 3D MEANDER [64] which supports the interleave of 2D and 3D switch boxes. The academic tools usually focus on a general architecture of FPGAs (*e.g.* the island-style) and thus cannot be directly used on the contemporary commercial FPGAs which have a lot of specific IP modules and other blocks. However, these tools are much more flexible than the commercial tools, and are usually open sourced. Therefore, they are sufficiently good for academic research and design space exploration of 3D FPGAs.

TPR Tool: Since TPR [63] is easy to be acquired and widely used in the investment of the design of 3D FPGAs, we will develop our framework based on TPR. Here, we briefly introduce this academic tool. TPR takes the CLB-level netlist and the 3D FPGA architecture as inputs. In the first stage, the netlist is partitioned into different layers to minimize the total number of TSVs. Following this, TPR performs timing-driven partitioning-based placement successively from the top layer to the bottom layer. After the position of each CLB is determined, the tool uses Pathfinder Negotiated Congestion-delay algorithm [62] to assign routing elements to each net based on the distribution of routing resources. After routing is finished, the delay of the circuit is calculated based on Elmore Delay Model. Finally, TPR will output placement file, routing file and delay. Note that the delay of the circuit determines the maximum operating frequency of the circuit thus influencing the power of the chip.

4.1.3 Motivations

In this project, we will follow the second topology to build the 3D FPGA. This kind of 3D FPGAs has several unique challenges which need to be solved: (1) relatively larger TSV dimension will cause the increase in the footprint area; (2) switch boxes that provide inter-layer connections will use more transistors resulting in extra power and delay; and (3) the stacked structure will exacerbate thermal problem by increasing both the power density and the thermal resistance from the active layer to the ambient. The thermal problem can be further exacerbated when the leakage

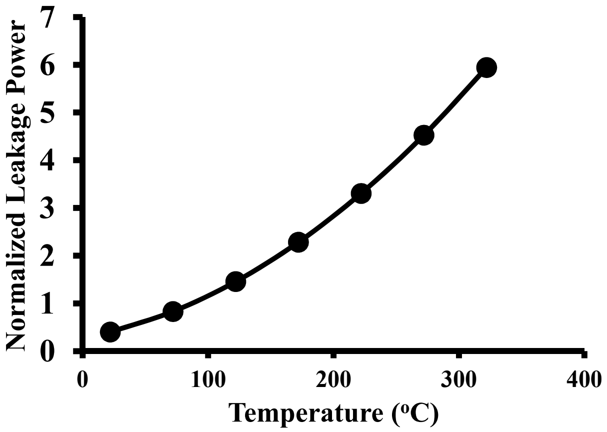


Figure 4.3: Trend of the leakage power with increasing temperature (with the data normalized to the leakage power at 85°C) [4]

power is taken into consideration. With the geometric and supply voltage scaling, leakage power becomes the primary contributor to the total power dissipation in FPGAs (which can account for up to 40% of the total power [65, 66]). Moreover, leakage power increases non-linearly with temperature as illustrated in Figure 4.3 [4]. The positive-feedback loop between the temperature and leakage power will lead to thermal runaway if 3D FPGAs are not sufficiently cooled.

The thermal problem of 3D FPGAs can be mitigated by using MF cooling (*e.g.* micro-channel cooling) as described in Section 1.1. However, applying miroc-channel cooling makes the design of 3D ICs even more complicated due to the several trade-offs introduced in Section 1.1. The case is even worse in 3D FPGAs. On one hand, the increase of vertical bandwidth (*i.e.* the number of TSVs) may not necessarily reduce the delay of 3D FPGAs. This is because although higher vertical bandwidth will reduce the latency of 3D nets, it will also cause congestion in 2D routing channels around TSVs due to the restriction of routing resources in 3D

FPGAs; this may increase the delay of the whole system if the affected 2D routing is in the critical path of the circuit. On the other hand, although increasing the vertical bandwidth will possibly reduce the use of switch boxes, the average number of transistors per switch box will increase since more transistors are required to support vertical connection and this will cause the rise of leakage power of a single switch box. Therefore, increasing the vertical bandwidth is not necessarily lead to the reduction of the power. Recalling the trade-off between the micro-channel density and the vertical bandwidth in 3D ICs (Section 1.1), the impacts of channel density on the delay and power dissipation of 3D FPGAs are nonmonotonic, while in 3D ASICs, the impacts of channel density on the delay and power dissipation are usually monotonic. This phenomenon makes it even harder to find a design of 3D FPGAs with micro-channel cooling which is optimal to both cooling and performance. Therefore, we propose a design-space exploration framework to achieve this (Section 4.2.2).

As described in Section 1.1, another challenge brought by MF cooling is that the cooling capability of the heat sink degrades along the flow direction thus may causing large in-layer temperature gradient and peak temperature if the placement of CLBs is not done properly. However, traditional thermal-aware placement techniques cannot be directly applied in 3D FPGAs. This is because the power of each block cannot be accurately obtained after the routing of the FPGA. In this chapter, we propose a cooling-aware placement technique to place and route 3D FPGAs such that the in-layer temperature gradient is significantly reduced (Section 4.3).

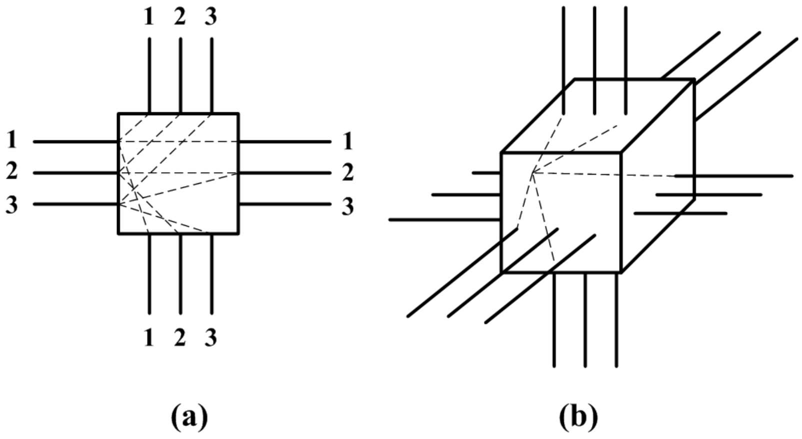


Figure 4.4: Illustration of the subset topology of the (a) 2D-SB and (b) 3D-SB

4.2 Design Space Exploration (DSE) of 3D FPGAs with Micro-Channel Cooling

4.2.1 Modeling of 3D FPGAs with Micro-channel Cooling

In this thesis, we focus on 3D FPGAs with stacked identical 2D island-style FPGA chips. This kind of FPGAs is illustrated in Figure 4.2(b). Some of the switch boxes (2D-SBs) in 2D FPGAs are extended to 3D switch boxes (3D-SBs) to support the inter-layer connection, which is achieved by fabricating TSVs between 3D-SBs in adjacent layers. In this project, we assume that each 3D-SB connects to the same number of TSVs.

Routing Fabric Modeling

In this project, we adopt the "subset-style" [63, 64] to design the switch box (as illustrated in Figure 4.4). In one switch box, an incoming routing track from one side is connected to routing tracks from other sides with the identical ID number.

Therefore, the flexibility (F_s) [64] of a 2D-SB equals three while a 3D-SB has $F_s = 5$ (including two TSVs connecting to the upper and lower layer respectively). Therefore, if both vertical and horizontal routing bandwidth in 3D FGPAs are equal to W_{route}, the number of transistors used in a 2D-SB is $6 \times W_{route}$ while the number of transistors in a 3D-SB is $15 \times W_{route}$. In practice, however, since TSVs occupy much more silicon area compared to 2D routing channels, we have to limit the number of TSVs connected by each 3D-SB (usually the number is smaller than the width of a 2D routing channel).

TSV Modeling

In 3D FPGAs, the property of TSVs will significantly affect the delay of the circuit. In order to model this impact, we use the following equations to characterize the property of TSVs [68]:

$$R_{TSV} = \frac{\rho_m H_{TSV}}{\pi r_{via}^2} \tag{4.1}$$

$$C_{TSV} = \frac{2\pi \epsilon_r \epsilon_0 H_{TSV}}{ln((r_{via} + t_{ox})/r_{via})} \tag{4.2}$$

In the above equations, ρ_m is the resistivity of the metal filling in a TSV, r_{via} is the radius of the metal filling region in the TSV, t_{ox} is the sickness of the insulate layer surrounding a TSV, and ϵ_r is the relative permittivity of SiO_2.

Area Modeling

The calculation of temperature requires accurate modeling of 3D FPGA area. In this project, we use model the FPGA area as the number of “minimum width transistor areas” [62]. The minimum width transistor area (MWTA) is defined as the area of the layout of the smallest transistor, plus the minimum spacing to other transistors located to the right and above it. In order to calculate the number of minimum width transistor areas, we first determine the number and size of transistors used in each programmable block (i.e. CLB, CB and SB). The schematic of each programmable block in our model is similar to the one described in [62]. For transistors with different sizes, their MWTA is calculated with:

$$\begin{aligned} \text{MWTA} &= \\ &0.5 + \frac{\text{Drive Strength}}{2 \times \text{Drive Strength of Minimum Width Transistor}} \end{aligned} \tag{4.3}$$

Following this, we can calculate the number of MWTAs for each programmable block and the "real" area is computed by multiplying the number of MWTAs with the area value of a certain technology node. Note that up to this stage, TSVs are not considered yet.

In our model, TSVs are assumed to be built between two 3D-SBs placed on adjacent layers. Therefore, the area of a 3D-SB should be added with the total silicon area occupied by TSVs connected to the 3D-SB. In order to simplify the analysis, in this work, we extend the "tile" of an island-style FPGA as introduced in Section 4.1.1 in the context of 3D FGPAs. By doing this, a tile can be categorized into 3D-tile or 2D-tile based on the different types of switch boxes constituting the tile. The area of a tile is the total area of all the programmable blocks including the TSVs in the tile.

Micro-Channel Density Modeling

Figure 4.5 shows a 3D FPGA with micro-channel cooling. We use the following physical parameters to characterize the structure of micro-channels: channel width (W_{ch}), silicon thickness between channels (W_{wall}), channel hight (H_{ch}) and

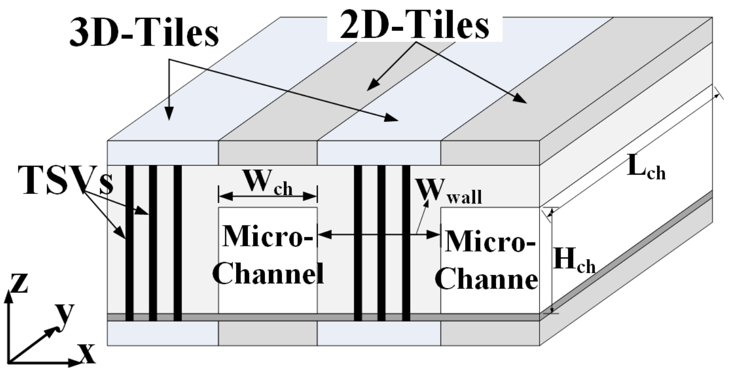


Figure 4.5: Illustration of a chip bounded 3D FPGA embedded with micro-channel based MF cooling

channel length (L_{ch}). We assume a micro-channel spans across the whole chip thus the length of the channel is equal to the length of the FPGA chip ($L_{ch} = L_{chip}$). The density of micro-channels is tuned by setting different silicon thickness values between channels. Without-loss-of-generality, we assume the thickness value can only be equal to or be a multiple of the channel width ($W_{wall} = \lambda W_{ch}$, $\lambda = 1, 2, 3, ...$). In this project, we will study how the micro-channel density affects the cooling capacity and performance of 3D FPGAs (by affecting the vertical FPGA bandwidth).

In this project, a 3D FPGA is modeled by the array of tiles. Therefore, we will capture the impact of micro-channel density on the distribution of different types of tiles. In order to do this, we first set all tiles in the 3D FPGA to be 3D-tile. Following this, we replace some 3D-tiles with 2D-tiles such according to the micro-channel density. Note that we assume a 2D-tile occupies the same area as a 3D-tile in a single layout. In this way, we can map the micro-channel distribution to the 3D FPGA architecture, thus characterizing different micro-channel densities by

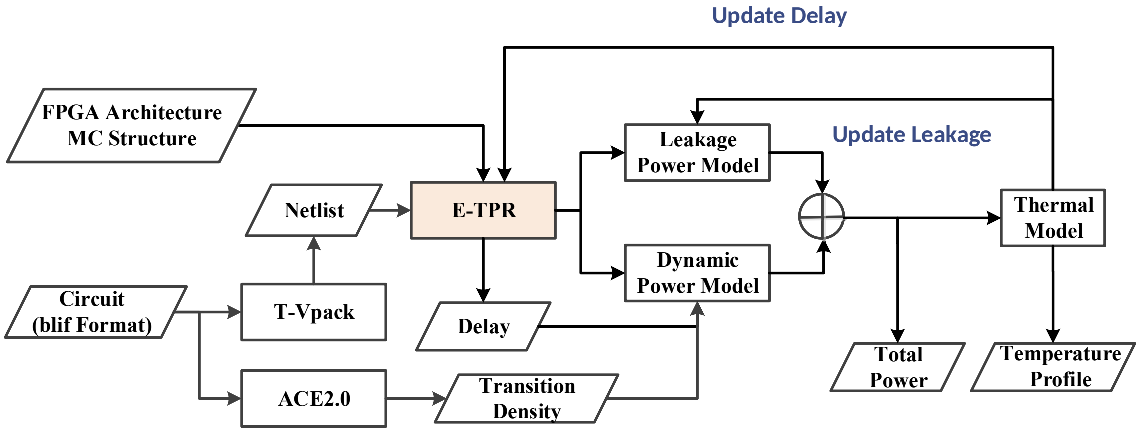


Figure 4.6: The illustration of our analysis framework

changing the distribution of 3D and 2D tiles in FPGAs. The distribution of different types of tiles will further influence the placement and routing in 3D FPGAs.

4.2.2 DSE Framework

In this section, we will introduce the method of studying the impact of micro-channel density on the performance, power and energy efficiency of 3D FPGAs. In this project, we use the operating frequency to evaluate the performance. Energy efficiency (E.E.) is defined by the following equation:

$$E.E. = Frequency^2/Power \tag{4.4}$$

Figure 4.6 shows the framework of our analysis methodology. The kernel of the framework is extended from TPR [63]. The modified TPR is denoted as E-TPR in our work. Originally, TPR only supports the 3D FPGA architecture with fully-vertical inter-connection (*i.e.* all the switch boxes in the 3D FPGA are 3D switch boxes). After the modification, we can use E-TPR to tune the distribution of 3D

SBs according to the given physical parameter of micro-channel and FPGAs (*e.g.* the width of micro-channels, the size of a CLB *etc.*).

A given circuit (in *.blif* format) is fed into T-Vpack (which is part of VPR [62], a P&R tool for 2D FPGA) and ACE2.0 [69] (an activity estimation tool) to generate a CLB-level netlist and transition density, respectively. Then, the CLB-level netlist, FPGA architecture and distribution of micro-channels are fed into E-TPR for placing and routing. Meanwhile, E-TPR also calculates the final delay of the circuit based on its embedded delay model. Following this, we take the placement file, routing file as well as the delay and transition density as the inputs to our power model to calculate the power dissipation profile. The power profile is used to calculate temperature profile which is then fed back into power model (Section 4.2.2.1) to update the leakage power. This "power-thermal" loop is necessary because of the positive feedback between leakage power and temperature. The "power-thermal" loop stops when the update of temperature profile is negligible.

4.2.2.1 Power Model

Our power model is based on the tile-structure of the 3D FPGA and we assume the power within a tile is distributed uniformly. Power dissipation of 3D FPGAs is the sum of dynamic power and static power (leakage power). The methodology of modeling the two types of power is introduced as follows.

(1) Dynamic Power

Dynamic power is generated from transition between signals. Signal transition will cause frequently charging and discharging capacitors and this forms the most significant contribution to the dynamic power in 3D FPGAs. This kind of dynamic power can be modeled using the following equation:

$$P = \sum_i C_i V_i^2 D_i f_i \tag{4.5}$$

where C_i, V_i, D_i and f_i are the total capacitance, swing voltage, signal transition density and operating frequency of source i. Another component of the dynamic power is the short-circuit power which is caused by signal switching. According to [70], while the short-circuit power accounts for a high percentage of the CLB power, it only contributes less than 10% to the power in the interconnect in an FPGA. In order to capture both factors of dynamic power and avoid complex computation, in our work, we model the dynamic power in two independent parts: Interconnect Power and CLB Power.

The calculation of **interconnect power** takes two steps. First, we compute the power for each routing segment (including horizontal routing segments on each layer and TSVs). A segment connects two terminals (a terminal is an input/output pin of a CLB or a pin of an SB). The equation to calculate the capacitor-charging-based dynamic power of segment i is shown as follows:

$$P_i = C_i V_i^2 D_i f_i \tag{4.6}$$

where C_i is the sum of input/output capacitance of the two terminals and the distributed capacitance of the segment; D_i is the signal transition density of the net to which the segment belongs and its value is calculated by ACE2.0 [69]; f_i is the operating frequency of the system and V_i is simply taken as the supply voltage.

Second, we project the power of each segment to the tile because the power of each tile is what really matters. In order to do this, we divide the segment power equally and assign it to the tiles containing the two terminals of segment, respectively. Note that, it is possible that both terminals of a segment reside in one tile. In this case, the whole power of the segment is assigned to this tile.

The **CLB power** is calculated as follows: According to our FPGA model, a CLB contains Look Up Tables (LUTs), flip-flops, multiplexers, buffers and memory cells. Modeling the dynamic power in a CLB is difficult due to the complex sub-routing within the CLB. In our work, we use a simulation-based method to model the CLB power. First of all, we will calculate the average dynamic power of an active CLB. We use SPICE to simulate each component in a CLB with random input vector pairs at a certain frequency. The simulation gives the dynamic power of each component for all the pairs of input vector. The dynamic power of each component is then taken as the average power for all the input vector pairs of the component, with the assumption that each pair of input vectors has the same probability of occurrence. The dynamic power of a CLB ($Power_{CLB,0}$)is the sum of all the components constituting the CLB. Next, we take $Power_{CLB,0}$ as the input and scale it with the real frequency to get the real dynamic power of an active CLB. Finally, this power is added to the tile which contains the CLB.

(2) Static Power

The static power is caused by the leakage current in FPGAs and can be categorized into two types: gate leakage and source-to-drain leakage. Detailed modeling of the two types of static power is difficult and not accurate at small technology nodes. Therefore, in our work, we take the experiment value from [71] and calculate the leakage power for different temperatures based on an analytical extrapolation function. According to [71], at 85^oC, Typical High-Performance Stratix III has $3W$ static power for 300K logic blocks (similar to the tile in our model). Since the static power is linear to the number of logic blocks, we can calculate the static power for a single logic block, which is around $10\mu W$. This value is then taken as the static power of a 2D-tile at 85^oC in our 3D FPGA model. The static power of a 3D-tile is $\gamma \times 10\mu W$ where γ is a scaling parameter equal to the ratio of the number of minimum width transistor areas (without considering the area occupied by TSVs) between a 3D-tile and a 2D-tile. Following this, we use the following equation [4] to extrapolate the static power for different temperatures:

$$P_{stat}(T) = P_{stat}(T_0) \times (5.121(\frac{T}{T_0})^2 - 6.013\frac{T}{T_0} + 1.892) \tag{4.7}$$

Now we get a look-up-table for static power at different temperatures, which is then fed into the simulation framework.

4.2.2.2 Thermal Model

The thermal behavior of 3D FPGAs with micro-channel based MF cooling can be characterized by the thermal model introduced in Figure 2.1. Readers may refer to Section 2.1.1 for more details. This thermal model can be expressed with a thermal conductance matrix, G. After we get the profile of the power dissipation profile, $\boldsymbol{P}$, we can calculate the temperature with the following equation:

$$G\boldsymbol{T} = \boldsymbol{P} \tag{4.8}$$

During the simulation process, when a new temperature profile is calculated, the static power of each tile is updated according to the new temperature. Then the temperature is recalculated based on the updated total power dissipation. This is illustrated in Figure 4.6. This loop stops until the temperature is converged.

Our framework will also capture the impact of temperature distribution on the routing delay of 3D FPGAs. The resistance of metal is linearly dependent on its temperature with the following expression: $R(T) = R_0(1 + \eta(T - T_0))$. Here T_0 is the nominal temperature, R_0 is the corresponding metal resistance and η is the temperature coefficient of the material of routing wires. Based on this fact, we calculate the resistance of a certain routing segment according to the average temperature on this segment. The delay of this segment is then calculated using the temperature dependent resistance. Since the delay will influence the total power thus affecting the temperature distribution, we update the delay iteratively until it is converged.

4.3 ECO-Based P&R Framework for 3D FPGAs with MF Cooling

In the previous section, we propose a DSE framework to find the proper density of micro-channels for a 3D FPGA with micro-channel based MF cooling such that the performance and energy efficiency are co-optimized. However, this DSE framework does not consider how the CLBs are arranged in the 3D FPGA. In other words, the placement in the proposed DSE framework (Section 4.2) is totally unaware of the MF-cooling. As described in Section 1.1, such cooling-unaware placement techniques might result in significant large in-layer temperature gradient which may harm the reliability of the circuit. In this section, we will modify the DSE framework proposed in the previous section to achieve cooling-aware placement of 3D FPGAs thus fixing the temperature non-uniformity problem induced by the MF cooling. This placement algorithm is introduced as follows:

In the proposed DSE framework (Section 4.2), E-TPR is used to do the placement and routing for 3D FPGAs with micro-channel based MF cooling, even though the placement does not consider degradation of cooling capability downstream the flow of the coolant. In our new placement algorithm, we also use E-TPR to place and route the 3D FPGA. Afterwards, we make some changes to the placement according to the thermal requirement and the 3D FPGA is replaced with E-TPR. This placement prototype is called Engineering Change Order (ECO) placement. Compared to developing a P&R tool from scratch, the ECO based approach utilizes the existing tools and achieve the objectives efficiently.

4.3.1 ECO Placement Algorithm.

In this section, we will introduce our algorithm to make ECO modification to the original placement ("Baseline") to improve the temperature uniformity. Due to the fact that the cooling capability decreases along the flow direction (i.e. y direction as shown in Figure 4.7), we hope to allocate more power close to the inlet end and this is achieved by shifting high-power nodes toward the inlet. Figure 4.7(a)-(d) illustrates the flow of the algorithm. We first divide each layer of the 3D FPGA

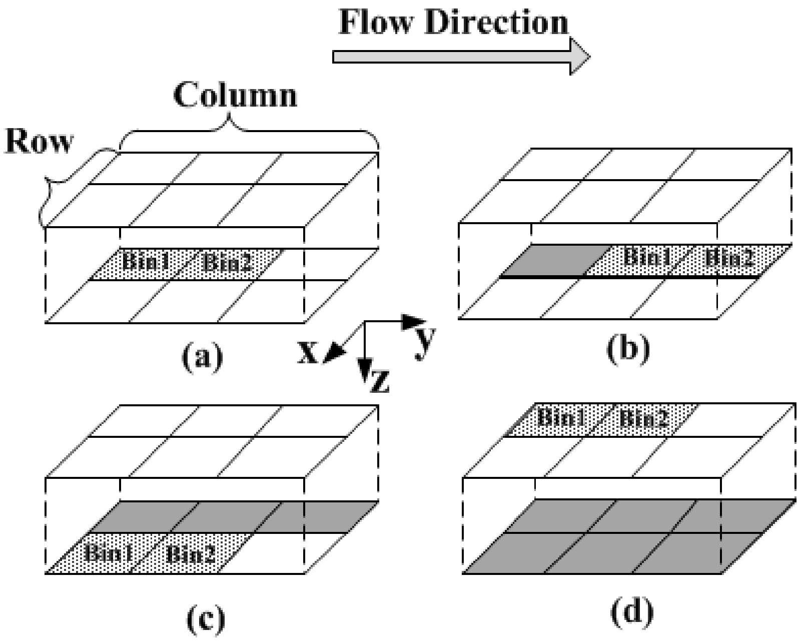


Figure 4.7: Illustration of the ECO placement applied on a 2-layer FPGA ((a) start processing the first two bins; (b) continue processing the following bins in the same column; (c) process bins in the next column; (d) process bins in the next layer)

into bins and each bin contains a number of CLBs. As depicted in Figure 4.7(a), bins located along the y direction form the *column*, while bins located along the x direction form the *row*. The bin size is $\lceil \frac{N_x}{\beta_x} \rceil \times \lceil \frac{N_y}{\beta_y} \rceil$, where N_x and N_y are the number of tiles in x and y direction, respectively; β_x and β_y are parameters tuning the number of tiles located in a bin. Nodes can only be shifted to the adjacent bin in the same column (in the same layer). Here, the node means the node in the netlist, which can be regarded as a gate. Shifting a node to a bin means implementing the gate using the CLB in the new bin. In our algorithm, we start from the bottom layer and process each column of bins from the inlet to the outlet. Details of the processing procedure will be introduced later in this section. Note that, if a CLB contains more than one node, all the nodes are shifted as a whole. Finally, we feed the new placement to E-TPR for rerouting.

When processing two adjacent bins (the bin close to the inlet is called $B1$ and the other bin is called $B2$ as shown in Figure 4.7), we shift a node from $B2$ to the CLBs in $B1$ if the power of that node is larger than the threshold (P_{th}). If there exist empty CLBs in $B1$, we can simply reallocate the node to these CLBs (i.e. implementing the function represented by the node using the empty CLBs). However, if all the CLBs are occupied, we have to choose appropriate CLBs and swap the node in that CLB with the one which we want to reallocate. In order to simplify the problem, we choose the target CLBs (in $B1$) according to their "available weight" (W_a) which is calculated as follows:

$$W_a = \alpha W_y + (1 - \alpha) W_P \tag{4.9}$$

where W_y is the normalized distance from the CLB to the border of $B1$ and $B2$. This term indicates the potential perturbation in wire-length when moving the nodes to this CLB (or swapping with the node in this CLB if applicable). Larger W_y implies larger perturbation in the wire-length. W_P is the normalized power of the nodes currently located in the CLB. If the CLB is empty, $W_P = 0$. This term exists because we always want to swap with a low-power node in the first place. W_a is thus calculated by the linear combination of W_y and W_P. Following this, we sort the CLBs in $B1$ in the ascending order of W_a and the target CLB is selected in this order. Besides, there are some restrictions on moving nodes across bins: (1) the incoming nodes cannot be swapped with nodes with power larger than P_{th}; (2) if a node in

B1 came from B2 during the previous process, it cannot be exchanged. When the processing of a pair of bins is completed, we continue to process the following pair of bins as shown in Figure 4.7(b). Now, the previous $B2$ becomes $B1$, and the previous $B1$ is marked "completed" and will not be revisited again. ECO algorithm completes when all the bins are processed. Therefore, the time complexity of this algorithm is $O(n)$ where n is the total number of nodes. Note that the results vary for different settings of P_{th} and β_y. P_{th} controls the number of nodes to be replaced and β_y determines the degree to which the original placement can be changed. We will discuss the impact of β_y on the results in the next section. Currently, we set P_{th} of each layer as the median of the cell power in that layer. We will investigate the approach generating the optimal P_{th} for the future work.

4.4 Experiment and Results

In this section, we first introduce the setup of the experiment including the parameters used to build the 3D FPGAs with micro-channel cooling. After this, we will first illustrate the efficiency of MF cooling by applying air-cooling and MF cooling to the fully-vertical-connected 3D FPGAs, respectively. Then, we describe the discoveries of the DSE framework (Section 4.2.2) and the guidelines for designing 3D FPGAs with micro-channel cooling. Finally, we will show and analyze the results after using our proposed framework (Section 4.3) to place and route 3D FPGAs.

Variable	Value	Unit	Description
T_{amb}	25	^oC	Ambient temperature
T_{ch}	25	^oC	Inlet temperature of micro-channel
v_f	1	m s^{-1}	Fluid velocity
k_{ox}	1.4	W m^{-1}K	Oxide thermal conductivity
k_{si}	149	W m^{-1}K	Silicon thermal conductivity
k_{cu}	401	W m^{-1}K	Copper thermal conductivity
k_f	0.6069	W m^{-1}K	Fluid thermal conductivity at 25^oC
R_{seg}	101	Ω	Resistance of one routing segment
C_{seg}	22.1	fF	Capacitance of one routing segment
V_{dd}	0.9	V	Supply voltage

Table 4.1: Electrical and thermal variables employed in the simulation

4.4.1 Setup of the Simulation

Our framework is evaluated with benchmarks from the Microelectronics Center of North Carolina (MCNC) benchmark suit. During our design space exploration, we study how the cooling method and the density of micro-channel influence the electrical properties of a 3D FPGA. By "electrical properties", we mean the operating frequency (Freq.), Power per CLB (P_{CLB}) and Energy Efficiency (E.E.), where Energy Efficiency is calculated with Equation 4.4. We define a new parameter, Micro-Channel Pitch (MC_{pitch}) to describe the micro-channel density:

$$MC_{pitch} = \frac{W_{wall}}{W_{ch}}. \tag{4.10}$$

For each micro-channel density, the vertical bandwidth is maximized by using the maximum number of 3D tiles. The range of MC_{pitch} is chosen from 1 to 11, which implies that the percentage of 3D-tiles among all tiles ranges from 50% to 92%. For the fully-vertical-connection, only air-cooling can be applied, while for other cases a hybrid cooling scheme (air-cooling and micro-channel-based fluidic cooling) is used. In this work, we assume the temperature limit is 85^oC [28]. If the maximum temperature exceeds this limit, the design is regarded as thermally infeasible. In this case, we will fix the thermal problem by scaling down the operating frequency. In the following, we summarize other parameters and variables for the 3D FPGA with micro-channel based MF cooling.

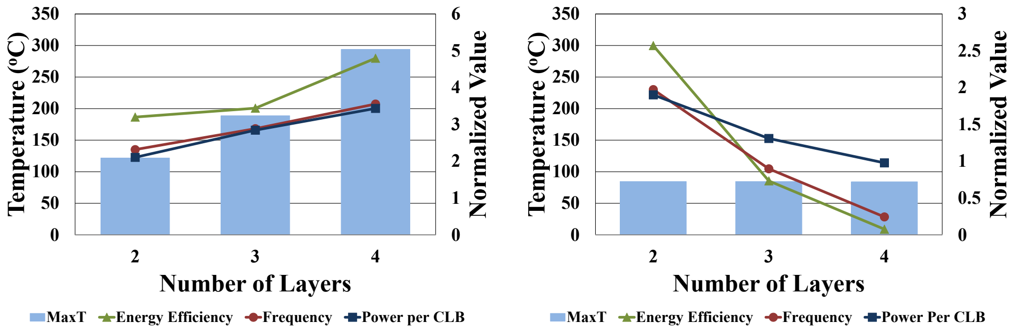
(1) The number of stacked layers of the 3D FPGA is {2, 3, 4}.

(2) Each 3D-SB has four TSVs connecting to the layer above and beneath it, respectively.

(3) The diameter and height of an TSV is $10\mu m$ and $150\mu m$, respectively. The resistance and capacitance of a single TSV is $32m\Omega$ and $223fF$, respectively.

(4) The width and height of a micro-channel is set as $50\mu m$ and $100\mu m$, respectively. According to [28], this will yield a pressure drop less than $1 \times 10^5 Pa$ and pumping power blow $1W$. The channel dimension is fixed during our simulation.

(5) each CLB is composed of a 4-input LUT, one flip-flop and corresponding memory elements. The width of horizontal routing channel is 30.



(a) (b)

Figure 4.8: Maximum peak temperature (MaxT), normalized frequency, normalized power per CLB and normalized energy efficiency for different number of layers (a) before fixing the thermal problem and (b) after fixing the thermal problem

(6) Table 4.1 lists the rest of the important electrical and thermal variables employed in this work.

In Section 4.4.2 and 4.4.3, we will discuss the results in detail.

4.4.2 Fully-Vertical-Connected 3D FGPAs

In fully-vertical-connected 3D FPGAs, all tiles are 3D-tile, thus micro-channels cannot be allocated under this scenario. Therefore, only air-cooling is applied. The results are shown in Figure 4.8(a). The bars represent the peak temperature over all benchmarks for different numbers of stacked layers while the lines represent the normalized frequency, energy efficiency and power per CLB, respectively. As illustrated by the figure, the performance, power and peak temperature increases with the number of stacked layers. The air-cooling scheme is not enough to provide sufficient cooling, thus the peak temperature of some benchmarks can exceed T_{limit}

even when only 2 functional layers are stacked. In addition, the peak temperature increases dramatically by increasing the number of layers in a 3D FPGA. Even though operating frequency and energy efficiency can be improved by stacking more layers in a 3D FPGA, these benefits cannot be realized with a thermally-infeasible design.

In this scenario, we can fix the thermal problem by scaling down the operating frequency, and the results are shown in Figure 4.8(b). It is interesting to discover from the figure that with the increase of the number of layers, the performance, power and energy efficiency decreases. This degradation can be reduced by using micro-channel-based fluidic cooling. Micro-channel structure imposes new constraints on 3D FPGA design and we will study the impact of micro-channel density on the electrical properties of 3D FPGAs in Section 4.4.3.

4.4.3 Design Space Exploration of 3D FPGAs with Micro-Channel Based MF Cooling

In this section, we will use our proposed DSE framework to study the impact of the density of micro-channels on the performance, power and energy efficiency of the 3D IC.

4.4.3.1 Before Fixing the Thermal Problem.

The result of applying the micro-channel based MF cooling is shown in Figure 4.9(a)-(c). As illustrate by the figure, the operating frequency increases with

	2 Layers	3 Layers	4 Layers
Freq.	15.0%	20.2%	4.8%
E.E.	15.1%	23.8%	6.0%

Table 4.2: Comparison to Best Cooling Design

	2 Layers	3 Layers	4 Layers
Freq.	24.3%	47.3%	80.3%
E.E.	35.5%	74.0%	124.0%

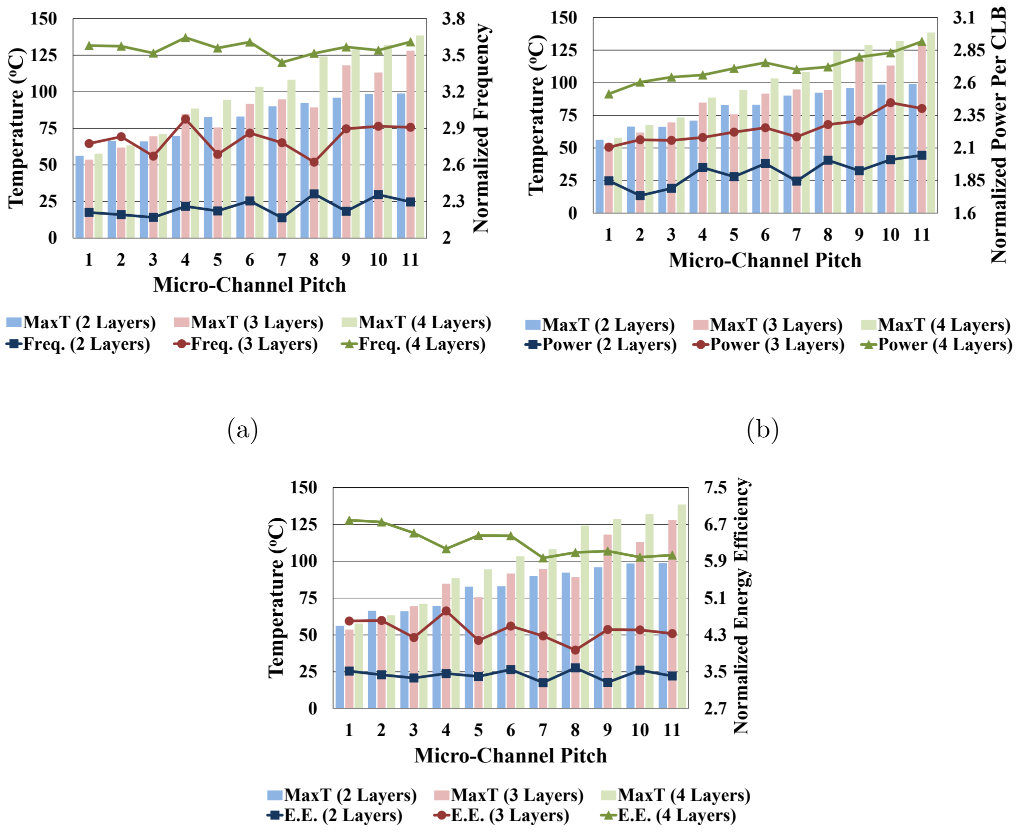
Table 4.3: Comparison to Best Bandwidth Design

the number of layers while it changes non-monotonically with the micro-channel density (which influence the vertical bandwidth). Authors of [64] also discovered this non-monotonic relationship which can be explained as follows:

(1) Although increasing vertical bandwidth will improve the latency in 3D nets, it can also cause congestion in 2D routing channels around the 3D-SB which may lead to the increase in delay of other nets.

(2) Since the distribution of routing resources changes by tuning the micro-channel density, the router may find completely new paths to connect CLBs. Since each routing channel has tracks of different lengths, these paths may have very significant variations in the RC parameters which are essential in calculating delay.

Since both power density and energy efficiency depend on the operating frequency, they also exhibit nonmonotonic behavior as shown in Figure 4.9(b)-(c).



(a)

(b)

(c)

Figure 4.9: Maximum peak temperature (MaxT) and normalized (a) operating frequency, (b) power per CLB (P_{CLB}), (c) energy efficiency for different number of layers and micro-channel density before fixing the thermal problem

Despite the non-monotonic behavior, there is an increasing tendency for the power per CLB when reducing the micro-channel density. This is because fewer micro-channels leads to higher temperature due to the poor cooling capability, thus causing the leakage power to increase. This phenomenon is illustrated by Figure 4.9(b). According to this figure, when the micro-channel density is fixed, increasing the number of stacked layers will increase the power per CLB. This can be explained as follows: reducing micro-channel density leads to lower mass flow rate of coolant which will reduce the cooling capacity of micro-channel heat sink and increase the on-chip temperature; increased on-chip temperature will in turn cause the rise of leakage power, thus increasing the power dissipation for each CLB. On the other hand, for a fixed micro-channel density, when another functional layer is added, the power dissipation per CLB will increase. This behavior is determined by three factors: (1) operating frequency may increase with the number of layers which leads to the increase of dynamic power per CLB; (2) increasing the number of stacked layers enables us to realize a certain benchmark using smaller number of CLBs; (3) increasing the number of layers causes the rise of on-chip temperature thereby increasing the leakage power per CLB due to the positive feedback between temperature and leakage power.

Energy efficiency is inversely proportional to the power while proportional to the frequency. Therefore, the energy efficiency enjoys a decreasing trend as the micro-channel pitch increases. As illustrated in Figure 4.9(c), this tendency becomes more obvious when the number of layers increases. By increasing the number of layers for a fixed micro-channel density, however, energy efficiency will increase since operating frequency is dominated.

4.4.3.2 After Fixing the Thermal Problem.

The peak temperature of the 3D FPGA is also illustrated in Figure 4.9. According to the figure, thermal violations can still occur even when micro-channel based MF cooling is applied, especially when the number micro-channel density is low. In order to fix the thermal problem, we scale down the operating frequency and the resulting frequency, power per CLB and energy efficiency are illustrated in Figure 4.10(a)-(c), respectively.

According to the figure, there is an obvious turning point for the frequency, power and energy efficiency as the micro-channel density decreases. And the drop of the electrical properties becomes sharper and occurs earlier when more layers are stacked in the 3D FPGA. This is because increasing the number of layers intensifies the thermal problem of 3D FPGAs and frequency should be scaled down to a much lower level in order to fix the thermal problem. The degradation of frequency meanwhile causes the reduction in power dissipation per CLB and energy frequency. According to the results shown in Figure 4.9 and 4.10, we can get the following guidelines for designing 3D FPGAs with micro-channel based MF cooling:

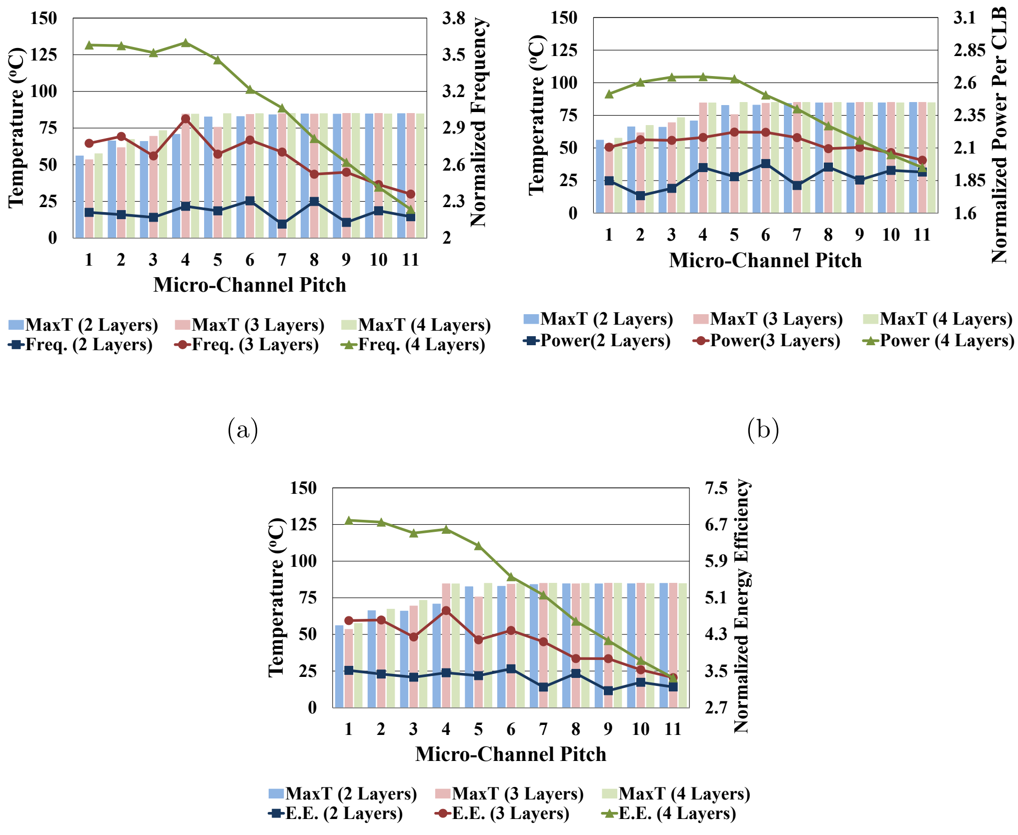
1. The performance of 3D FPGAs with higher vertical bandwidth will be “locked” due to the thermal restriction.

2. There is an “unlocked window” close to the lower-vertical-bandwidth end where we can find the optimal design of a 3D FPGA.

3. The size of unlocked window shrinks as the number of stacked layers increases.

4. 3D FPGAs with different number of stacked layers have different optimal physical architecture.

Without the DSE framework, we can either design the 3D FPGA with the



(a)

(b)

(c)

Figure 4.10: Maximum peak temperature (MaxT) and normalized (a) operating frequency, (b) power per CLB (P_{CLB}), (c) energy efficiency for different number of layers and micro-channel density after fixing the thermal problem

largest micro-channel density (Best Cooling) or the smallest micro-channel density (Best Bandwidth). However, either design is optimal in performance or energy efficiency. With our proposed DSE framework, we can identify the optimal physical architecture of 3D FPGAs with different number of stacked layers. The improvement of our technique compared to the Best Cooling and Best Bandwidth are shown in Table 4.2 and 4.3 respectively.

Benchmarks	bigkey	des	dsip
No. of CLBs	81K	60K	60K
CLB Usage	60%	76%	65%

Table 4.4: Benchmark information

Benchmarks	bigkey			des			dsip		
	maxT	ΔT	Freq	maxT	ΔT	Freq	maxT	ΔT	Freq
ECO1	-0.06%	-16.30%	-3.40%	-0.03%	-18.50%	-0.70%	-0.03%	-6.70%	1.90%
ECO2	-0.21%	-30.20%	-5.40%	-0.15%	-32.30%	-6.90%	-0.03%	-23.60%	1.30%
ECO3	0%	-26.70%	-4.00%	0.15%	-30.80%	-4.10%	-0.48%	-38.20%	-7.60%

Table 4.5: Improvement for ECO1-ECO3 compared to the baseline case (B-MC)

4.4.4 Results of the ECO-Based P&R Framework

In this section, we will test the cooling-aware placement proposed in Section 4.3. We fix the number of stacked layers of the 3D FPGA to 3. Other settings are the same as introduced in Section 4.4.3.

We perform ECO placement with three different values of β_y: (1) $\beta_y = 10$ (ECO1), (2) $\beta_y = 5$ (ECO2) and (3) $\beta_y = 4$ (ECO3). Meanwhile, we fix the value of $\beta_x \times \beta_y$ in order to keep the candidate CLBs in each bin almost constant. Note that when β_y decreases, nodes are more free to move in the y direction, but more constrained in the x direction. The "baseline" of the experiment is the original P&R result of E-TPR and we evaluate the temperature of the baseline using micro-channel based MF cooling (B-MC). Peak temperature is significantly reduced when MF cooling is applied. However, without thermal-aware placement, the MF cooling cannot guarantee a thermally feasible design. On the other hand, our algorithm can always generate a thermally feasible design by choosing the appropriate β_y.

Even though one can achieve peak temperature reduction by scaling down the operating frequency, this method still ends up with significantly larger temperature non-uniformity compared to our algorithm. In order to demonstrate this, we scaled down the operating frequency of B-MC and ECO1-3 such that the peak temperature of each case is within T_{limit}. The increase in peak temperature (maxT), in-layer temperature difference (ΔT) and operating frequency (Freq) of each ECO compared to B-MC is shown in Table 4.5. As illustrated in the table, the peak temperature is almost the same for different techniques. However, our algorithm can achieve up to 38% improvement in temperature uniformity by choosing the appropriate β_y. The algorithm causes the reduction of operating frequency. However, compared to the improvement in temperature uniformity, the cost in frequency is insignificant. In some cases (e.g. ECO1 for *dsip*), our ECO placement even improves the routing and leads to a higher operating frequency.

4.5 Conclusion

In 3D FPGAs with MF cooling, the thermo-electrical trade-offs are more complex than those in 3D ASICs. In order to handle these trade-offs and determine the proper designs for 3D FPGAs as well as the MF cooling heat sinks, we propose a design space exploration framework. Compared with naive design methods (*e.g.* designs to include the maximum number of micro-channels *etc.*), our DSE framework can find more optimal designs with significant improvement in performance and energy efficiency. Moreover, according to the experimental results, we can also provide guidelines for designing 3D FPGAs with micro-channel based MF cooling. We also extend the proposed DSE framework to place and route 3D FPGAs with micro-channel based MF cooling such that the in-layer temperature gradient is reduced.

Chapter 5: Run-time Management for 3D CPUs

The 3D integration brings challenges to the run-time management for 3D CPUs. One of the major reasons is that the power noise and temperature can be coupled across layers. This means the activity of one layer in the 3D CPU may affect the performance and reliability of other layers through voltage and temperature coupling. In this Chapter, we will use the stacked memory-on-logic to study the inter-layer coupling effect and how this affects the performance and reliability of the 3D CPU. Following this, we propose to manage the operation of the processor during the run-time to maximize the CPU performance subject to the reliability constraint.

5.1 Background of Stacked DRAM on Multi-core Processors

Traditionally, DRAM memories and processors are implemented on separate chips and connected through an off-chip bus. This off-chip DRAM-processor bus limits the memory access speed in two ways: (1) The bus operating rate is quite low due to the long length of the bus (around 55mm [4]) which causes large resistance and capacitance. (2) The bus bandwidth is limited by the number of pins on the chip. Since the common off-chip bus bandwidth (usually 64 bits [4]) is smaller

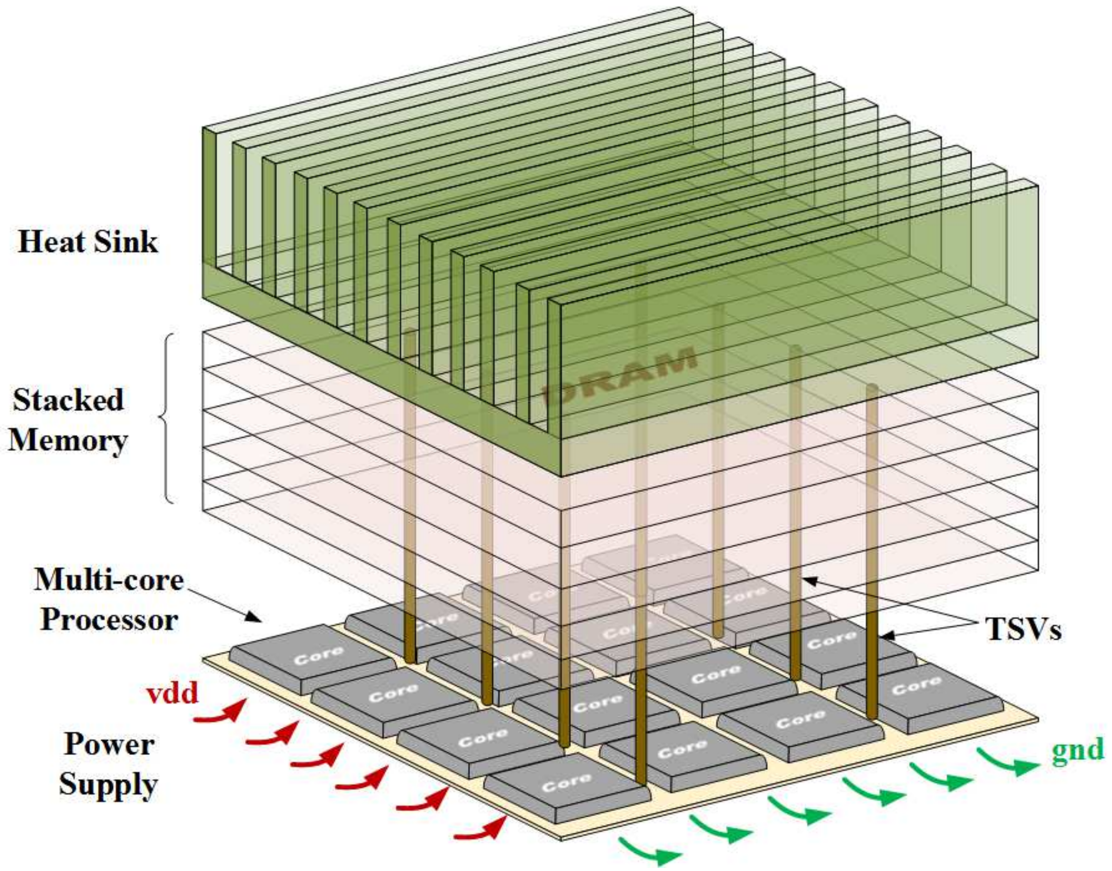


Figure 5.1: Illustration of the stacked memory-on-logic structure

than most contemporary cache line size, CPUs need to use multiple clock cycles to transfer only one cache line. Therefore, the off-chip buses become a bottleneck to the performance of CPUs. To make things worse, since DRAM cannot scale at the same rate as processors, the gap between processor clock rate and memory latency increases over time. One possible solution to this problem is to build deep on-chip cache hierarchies. However this method will occupy large amount of chip area and consumes more power.

On the other hand, stacking of DRAM on top of Multi-core Processors (MCPs) and bounding them with TSVs (as illustrated in Figure 5.1) enjoys a number of benefits: (1) the stacked memory has much higher DRAM-processor bus speed (around 2 GHz [72]) since TSVs are much shorter and less resistive than off-chip buses; (2) the "surface bounding" structure enables high density vertical connection which eliminates the limitation of I/O pins, thus increasing the memory bandwidth significantly; (3) different DRAM ranks is enabled to connect the processor through separate DRAM buses, thus allowing parallel access to different parts of DRAM as shown in the work of Meng *et al.* [12]; (4) faster main memory access helps reduce the size of cache hierarchies which leads to the reduction of footprint area and on-chip power consumption. According to [72], the stacked DRAM on MCP system can achieve over 90% improvement in performance compared to the 2D design.

In 2011, a new 3D stacked memory, Hybrid Memory Cube (HMC), was announced and was promised to be 15x faster than DDR3 [73]. This 3D memory uses TSVs to connect 4 to 8 dies of memory cell arrays which are stacked on top of each other. A certain number (usually 8 or 16) duplex differential serial links are used to connect the HMC with processors and other HMCs (in this way, the HMCs can be chained together to form larger storage capacity). Although the detailed architecture of HMCs is still confidential, there are several researches on the simulator which simulates the function of this new 3D memory [74,75]. Currently, HMCs have been used in SoCs with 2.5D technology, where HMCs are connected with processors through wires in the interposer. So far, the real stacked HMC on processor is not available in the market. One of the main obstacles is power noise and thermal coupling from the processor due to its high activity rate.

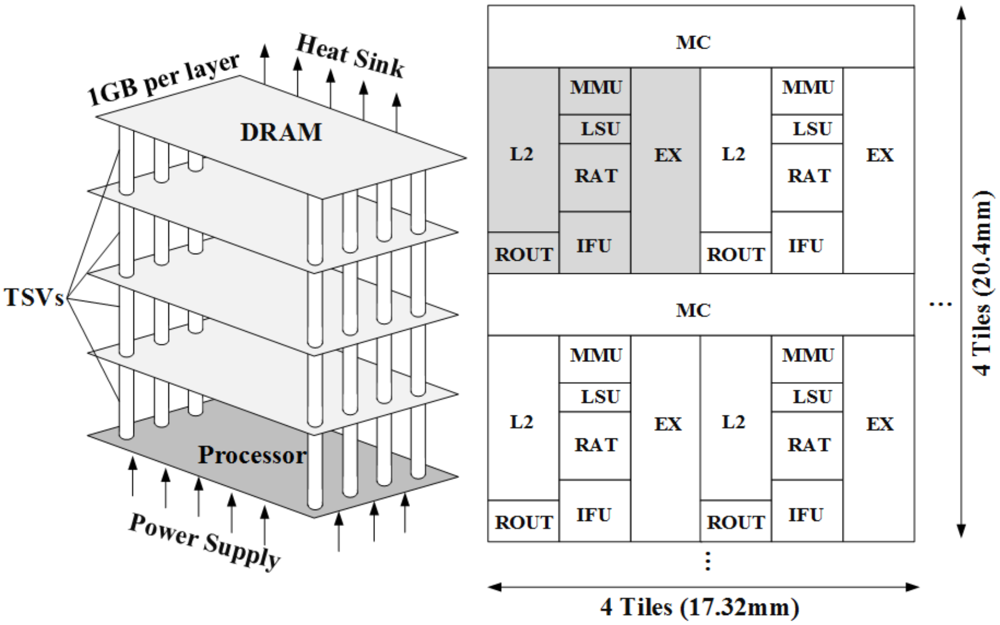


Figure 5.2: Illustration of (left panel) the 3D memory-on-logic Architecture and (right panel) the floorplan of the multi-core processor

5.2 3D Stacked Memory-on-Logic Architecture

In order to investigate the property and challenges of the stacked memory on logic structure, we will define a specific structure of the 3D CPU. The simulations in this chapter are all performed on this structure.

The stacked-memory-on-logic structure used in this chapter is illustrated in Figure 5.2. In this structure, four DRAM layers are connected vertically using TSVs and one multi-core processor layer is placed below the stacked memory layers. There are 16 cores on the processor layer. Figure 5.2(b) illustrates the floorplan of the multi-core processor. Each core in the processor consists of a load store unit (LSU), an execution unit (EX), an instruction fetch unit (IFU), a rename unit (RAT) and a memory management unit (MMU). The core is fabricated with 45nm technology node. A 8MB shared L2 cache is uniformly partitioned and placed adjacent to each core. 8 memory controllers (MCs) are implemented between the rows of cores with each MC connecting to two cores. The total area for the processor layer is $3.53cm^2$. Each layer of the stacked DRAM contains 1GB memory, and the area of each layer of DRAM is similar to the processor area [4]. The DRAM is connected to each MC through a 64B bus implemented with TSVs and each memory controller controls data usage in a certain range of address in the stacked memory. We list other important architectural parameters in Table 5.1. In this structure, we assume the power is supplied from the bottom to the processor layer and then is distributed to DRAM layers through P/G TSVs. The air-cooling heat sink is placed at the top of the chip which is close to the DRAM layers. Note that, here we choose the air cooling heat sink simply to illustrate the run-time management technique proposed in this chapter. If micro-fluidic cooling is used as introduced in Chapter 3, the proposed management technique will still work.

5.3 Inter-layer Coupling Effect

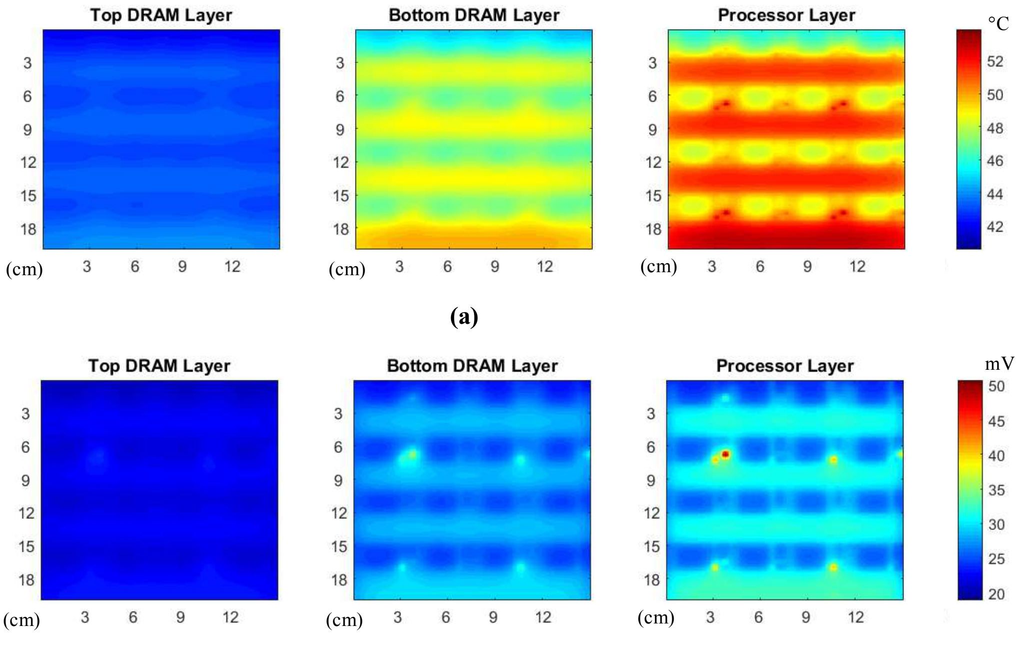
Figure 5.1 illustrates a 3D stacked memory on processor structure where the air-cooling heat sink is located at the top of the chip (which is close to the stacked DRAM) while the power is supplied from the off-chip circuit first to the processor layer and is then distributed to the DRAM layers through P/G TSVs. In this stacked structure, since the power dissipation in the processor layer is dominant, it

Name	**Value**
Branch Predictor	4K Entry 2-Level
Issue	Out of Order
Reorder Buffer	64 Entries
Fetch/Decoder/Issue Width	4
Functional Units	4 IALU, 1 IMULT, 2 FPALU, 1 FPMULT
BTB Size	1024 Entries
Private L1 I/D Cache	256 Sets per Core, 2-Way, 64B Block @ 2 cycle
Shared L2 Cache	512 Sets per Core, 2-Way, 64B Block @ 7 cycles
NOC Link Latency	3 cycles

Table 5.1: Architectural parameters of the multi-core processor

will significantly affect the temperature and supply voltage of the adjacent memory layers.

We have performed a simulation on the 3D structure as introduced in Section 5.2 by running BLACKSHOLE benchmark from the PARSEC benchmark suit [102]. All the 16 cores are used and the operating frequency is 3GHz. We assume the stacked DRAM totally dissipates 3W power which is distributed uniformly across the four layers. On the other hand, the power of the processor layer is computed using McPAT [76]. The air-cooling heat sink has an average heat flux of $23W/m^2K$. The settings of the 3D PDN are the same as in Table 3.5. The temperature map and voltage drop map of the top/bottom DRAM layer and the processor layer are shown in Figure 5.3. According to the figure, we can see that even though the DRAM dissipates very small power, there is still significant temperature and



(a)

(b)

Figure 5.3: The profile of (a) temperature and (b) voltage drop of different layers in the 3D CPU as introduced in Section 5.2

voltage variation in the layer, especially the layer closest to the processor (*i.e.* the bottom DRAM layer). This phenomenon demonstrates that the impacts of the activity of the processor layer will be coupled into the nearby DRAM layers and this will influence the performance of the DRAM (thus affecting the performance of the 3D CPU) and the reliability of the CPU. We will introduce these impacts in detail in the following subsections.

5.3.1 Impacts on Performance

The DRAM latency is affected by the supply voltage. In the stacked DRAM on processor structure, the memory is modeled with five different components: (1) MC queue latency, indicating the amount of time for a memory request waiting in

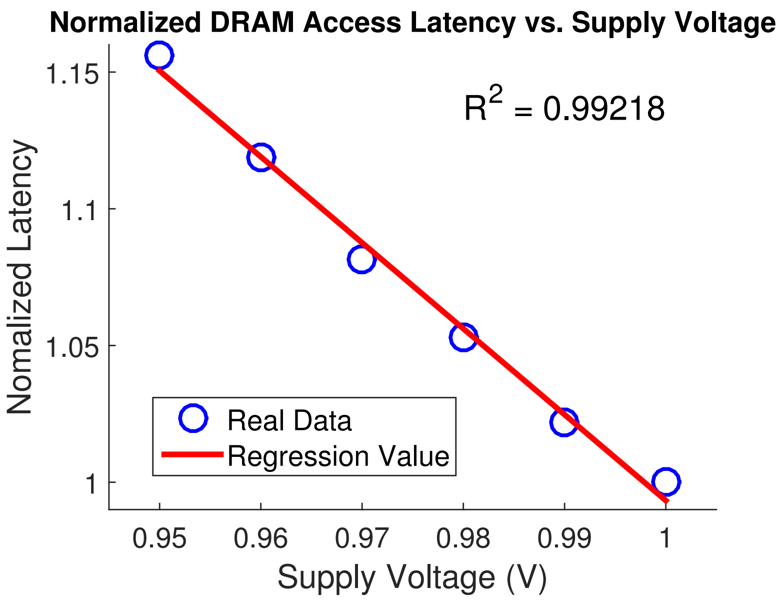


Figure 5.4: Normalized DRAM access latency vs. supply voltage

the memory controller queue; (2) MC decode latency, representing the time required to decode the memory addresses; (3) MC-DRAM latency: indicating the time taken to transfer a memory request through the MC-DRAM bus; (4) DRAM access latency, which is the time spent to access the DRAM cores; (5) data transfer latency, indicating the time to transfer the data from DRAM to the memory controller. [4] have studied the modeling techniques of DRAM memory.

The DRAM access latency in a 3D DRAM is estimated to be 32ns according to [4]. However, this value is acquired by assuming a perfect supply voltage in DRAM. In fact, the drop of supply voltage will increase the DRAM access latency [77]. In order to model this effect, we implement a DRAM cell in *Cadence Spectre*. The simulation is carried out at 90nm technology node, although we strongly feel that this trend would be similar at more state-of-the-art technology nodes. The standard supply voltage of DRAM is 1V with the minimum allowed supply voltage equal to

0.95V. We sweep the supply voltage from 1V to 0.95V and simulate the latency of reading one bit from the DRAM cell. Since reading signal "0" or "1" has distinct latency, we average the latency values of these two processes to represent the DRAM access latency. Simulation results illustrate the normalized DRAM access latency vs. supply voltage perfectly fits a linear model (Figure 5.4). Assuming the access latency is 32ns when the supply voltage is 1V, the access latency is then modeled as

$$T_{acc} = -99.2 \times V_{DD} + 131.2(ns) \tag{5.1}$$

Here, V_{DD} is the supply voltage ranging from 1V to 0.95V. Since lower DRAM access rate will increase the waiting time of a memory request in the MC queue, the MC queue latency calculated in Equation 5.1 will be scaled according to the actual DRAM access latency:

$$T_Q = \frac{T_{acc}}{T_{acc,0}} \times T_{Q,0} \tag{5.2}$$

where $T_{acc,0} = 32ns$, which is the nominal DRAM access latency when $V_{DD} = 1V$.

In practice, the high activity of the processor causes the supply voltage drop in the above memory layers. If the voltage drop is larger than the threshold, the memory cannot function correctly. Even if there is a little drop, the memory latency will be increased thus reducing the performance of the 3D CPU.

5.3.2 Impacts on Reliability

The coupling effect will also influence the reliability of memory. In general, there are two types of reliability issues: one is related to the soft error and the other is hard error (or called lifetime error). The soft error occurs because some bits stored in the DRAM cell flips so that the storage information becomes incorrect. Usually, soft errors can be mitigated using ECC techniques [78,79]. However, if the soft error occurs so frequently, the ECC process will degrade the performance or even fail to handle the problem. The reasons of soft error are various, including radiation [80] and power noise on the word line [81] *etc.* . In the 3D stacked DRAM on processor structure, the activity of the processor layer will cause the voltage nose on the word line in the DRAM which may cause the leakage of the DRAM cell. This effect leads to transient error in the memory which may require frequent refresh to mitigate, thus affecting the performance. Lu *et al.* has studied this effect and proposed a run-time management method to handle this problem [81]. In this chapter, we will focus on the hard error induced by the power/thermal coupling effect in the stacked DRAM on processor structure.

One of the most important hard failure mechanisms in the 3D stacked DRAM on processor structure is electromigration (EM) which usually happens in the metal interconnect. EM causes the metal atomic diffusion thus generating voids in the interconnect. A lot of work has shown that EM-induced voids appear frequently at the interface between wires and TSVs due to the drastic dimension mismatch between the two structures [39, 40, 82]. The EM effect is more severe in power delivery networks and P/G TSVs since the current in these structures is large and has uni-direction. EM effects cause the impedance of 3D PDNs to increase thus degrading the power delivery efficiency.

EM is affected by several factors including the current density, temperature, material properties *etc.* . In the 3D stacked DRAM on processor architecture, the power of the processor can influence the current and temperature due to the coupling effect, thus affecting the reliability of the 3D PDN. Several researches have investigated the EM-based TSV reliability issue and a number of related reliability models have been proposed [13,39–41]. In these models, the percentage of resistance increase [83] or the fixed amount of resistance increase [84] are two most common criteria for the failure of TSVs. Based on this, experiment has discovered that the mean time to failure (MTTF) of a TSV due to EM can be estimated as [85]:

$$MTTF = \frac{A}{j^n} \exp(\frac{E_A}{k_B T}) \tag{5.3}$$

where E_A is the activation energy for the EM process, j is the current density, k_B is the Boltzmann constant and T is the temperature. Usually, $\frac{1}{MTTF}$ is referred to as failure rate (λ) [86]. Several previous work estimated the Black's parameters for EM process of TSVs by analyzing of the experimental data. For example, Frank *et al.* [39] provided $E_A = 0.9 \pm 0.1eV$ and $n = 2 \pm 0.2$ for the TSVs of thin metal process. In this work, we use the $E_A = 0.82eV$ and $n = 2$ for simulation.

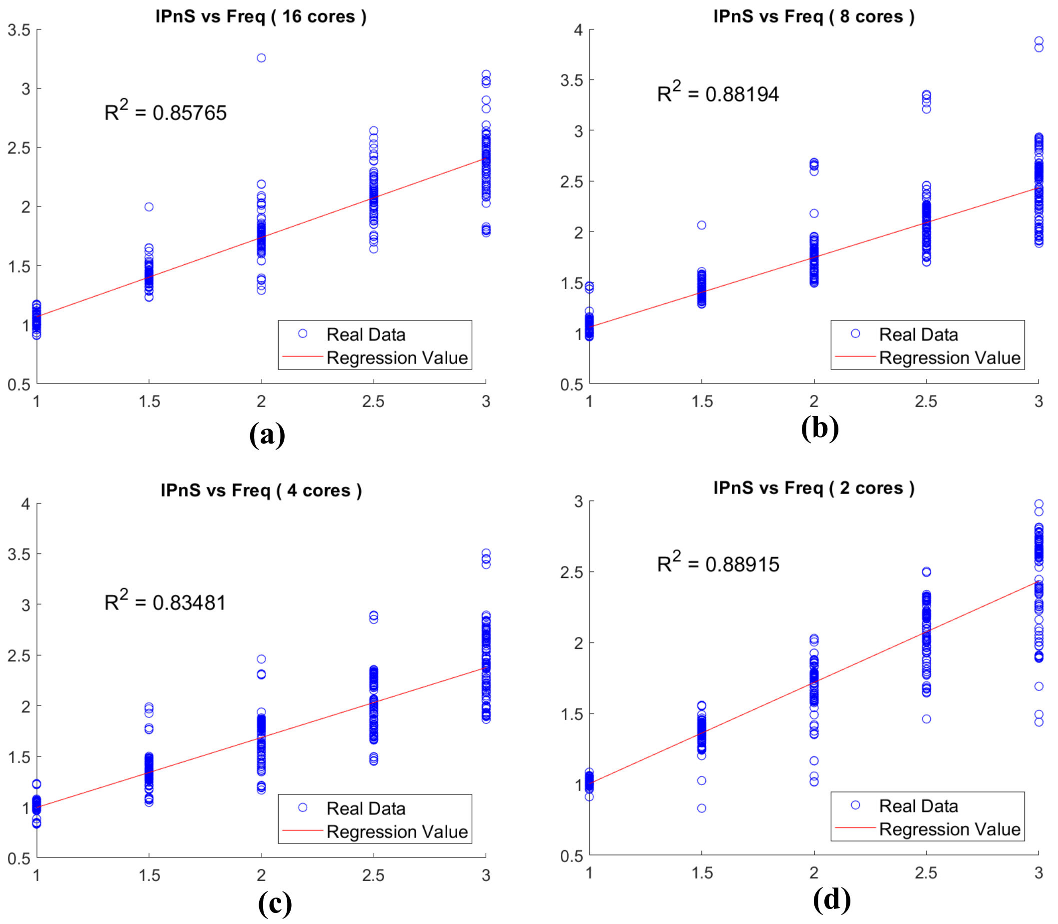
The reliability of a single TSV is defined as the probability that the TSV does not fail until a certain time. According to the previous work, reliability can be modeled as the cumulative distribution function (CDF) of a Weibull distribution, which is expressed as $F(t) = 1 - \exp(-(\frac{t}{\eta})^{\beta})$, where $F(t)$ refers to the probability that a TSV fails at time t, η is the lifetime parameter proportional to MTTF and β is the shape parameter of the Weibull distribution. Without loss of generality, let η =MTTF= $\frac{1}{\lambda}$. Then the reliability of a TSV at time t under a certain physical condition (θ) is expressed as:

$$R(t, \theta) = 1 - F(t, \theta) = \exp(-(\lambda t)^{\beta}) \tag{5.4}$$

The failure of a single P/G TSV (λ) will not affect the correct function of the system yet it will degrade the performance. When a large number of P/G TSVs fail, it will cause the failure of the whole system due to the large voltage drop in the DRAM layers. Moreover, it is the P/G TSVs on the bottom layer that are most vulnerable to the EM because they have the highest current density and temperature compared to TSVs on other layers. If the P/G TSVs at the bottom layer fail, the power delivery system will fail no matter the health condition of P/G TSVs on the upper layers. Therefore, we use the average failure rate of all the P/G TSVs at the bottom layer of the 3D CPU to evaluate the impact of EM-induced P/G TSV failure on the reliability of the whole system.

5.4 Impacts of Processor Activity on Performance, Power and Reliability

From the previous section, we know that in the stacked memory-on-logic structure, the activity of multi-core processors will influence the voltage noise and performance of the stacked memory as well as the reliability of the system. In this section, we will briefly investigate how the configuration of multi-core processors affects the performance and power of the processor as well as the reliability of the system. The configuration of the multi-core processor is determined by two dimensions: the number of active cores and the operating frequency. We use the 3D CPU



(a)

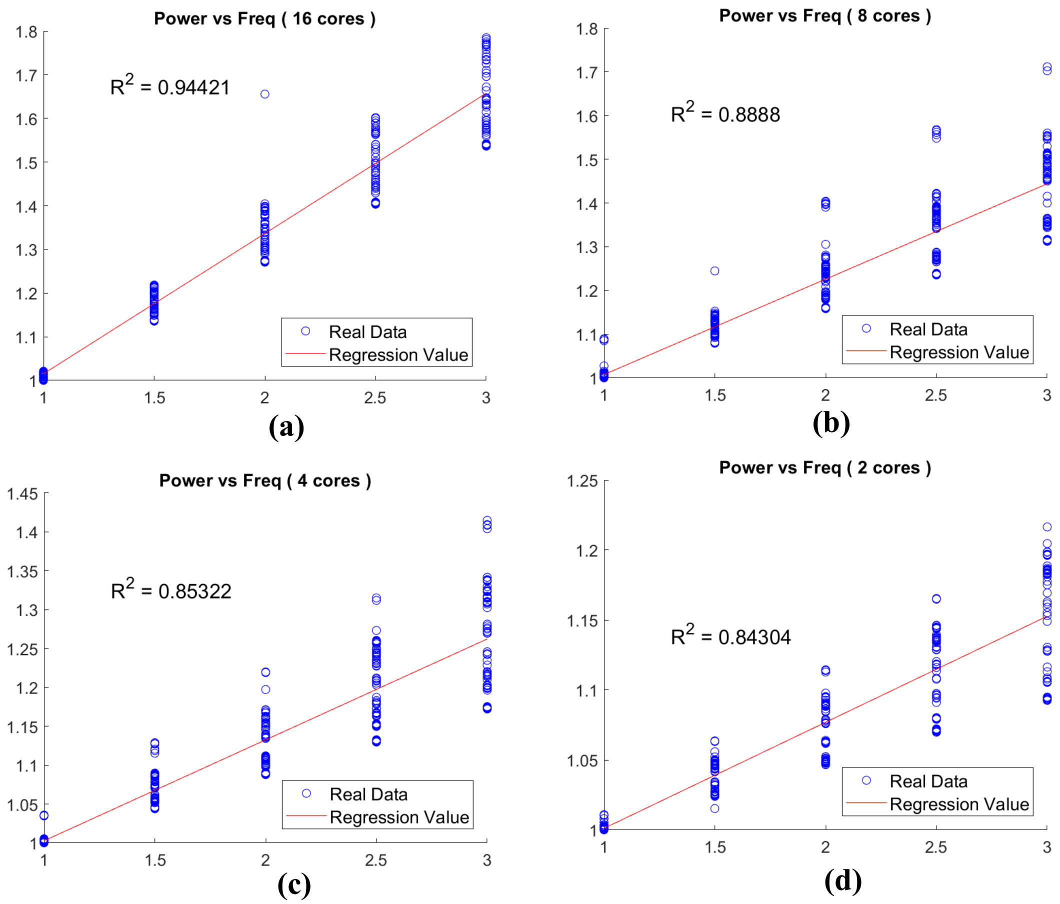
(b)

(c)

(d)

Figure 5.5: The change of normalized performance with frequency when the number of active cores is (a) 16, (b) 8, (c) 4 and (d) 2

structure introduced in Section 5.2 to run 15 benchmarks from SPLASH-II [103] and PARSEC [102] benchmark suits. Since some benchmarks can only run with the number of cores that is the power of two, we only consider using {2, 4, 8, 16} active cores in this section. If a core is not active, it is considered as clock gated. In addition, we assume all the active cores have the same power state (*i.e.* the same operating frequency), and there are five different frequency levels: {1, 1.5, 2, 2.5, 3} GHz. In the following part of this section, we will analyze the impacts of the



(a) (b) (c) (d)

Figure 5.6: The change of normalized total power with frequency when the number of active cores is (a) 16, (b) 8, (c) 4 and (d) 2

operating frequency and number of cores on the performance, power and reliability, respectively. The reliability here is influenced by the EM-induced P/G TSV failure, which is calculated as introduced in Section 5.3.2.

5.4.1 Influence of the Operating Frequency

Figure 5.5 (and Figure 5.6) illustrates the change of the normalized performance (total power) with the operating frequency for different number of cores. Each point in the figure represents the result for running one benchmark with a

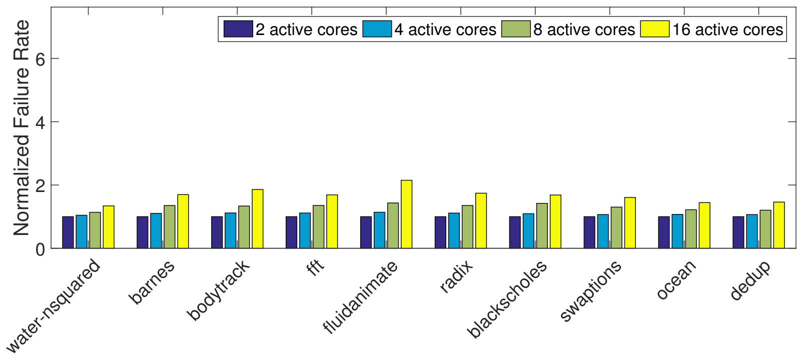


Figure 5.7: Normalized failure rate vs. the number of active cores for different benchmarks (clock frequency = 3GHz)

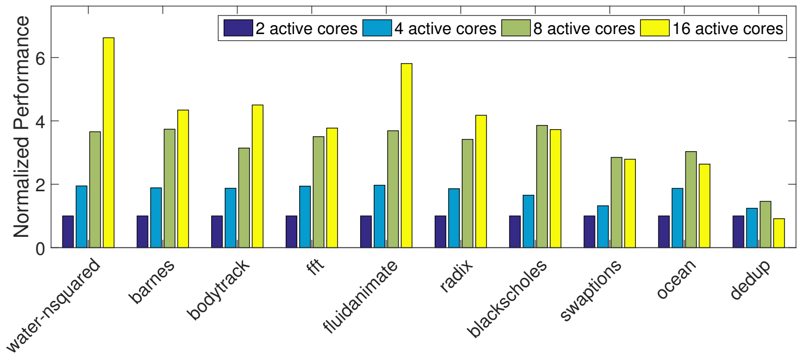
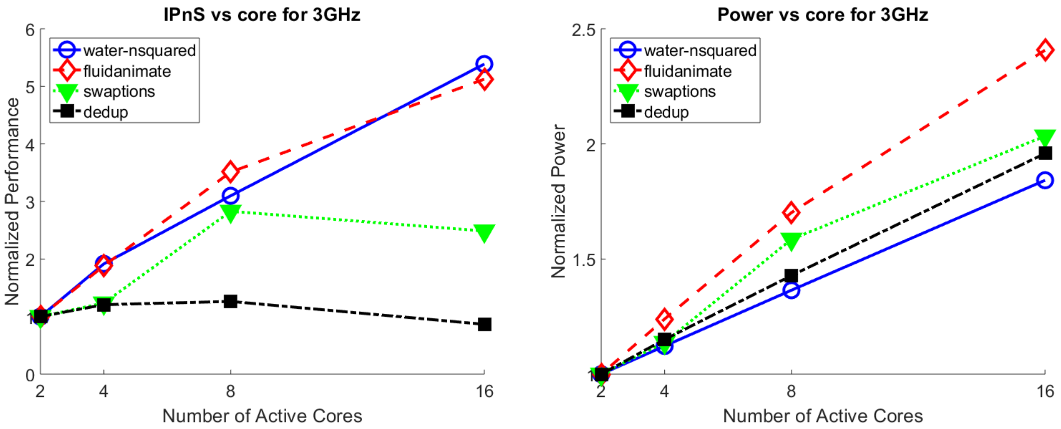


Figure 5.8: Normalized performance vs. the number of active cores for different benchmarks (clock frequency = 3GHz)

specific processor configuration. As illustrated by the figure, Even though there is some deviation, the performance and total power change linearly with the operating frequency (with high R-square values) when the number of active cores is fixed. This means when the number of active cores is fixed, reducing the frequency leads to the reduction in performance and power. Figure 5.7 and 5.8 illustrate the failure rate and performance, respectively, when executing different benchmarks with various frequency and 16 cores. According to this figure, reducing frequency leads to the



(a) **(b)**

Figure 5.9: The change of normalized (a) performance and (b) and total power for different number of active cores (when operating frequency is 3GHz)

reduction in both performance and failure rate. We can learn from this trade-off that we need to select the operating frequency properly in order to balance the performance and reliability.

5.4.2 Influence of the Number of Active Cores

Figure 5.9 illustrates the change of performance and total power for different number of active cores at 3GHz for 4 benchmarks. Each point in the figure represents the average value among all configurations with the corresponding number of active cores and frequency. From the figure, we can find that even though the power still changes linearly with the number of cores, the performance changes non-linearly with the number of cores. The impacts of tuning the number of active cores on performance and failure rate can also be illustrated by Figure 5.10 and 5.11, respectively, when executing different benchmarks with various number of active

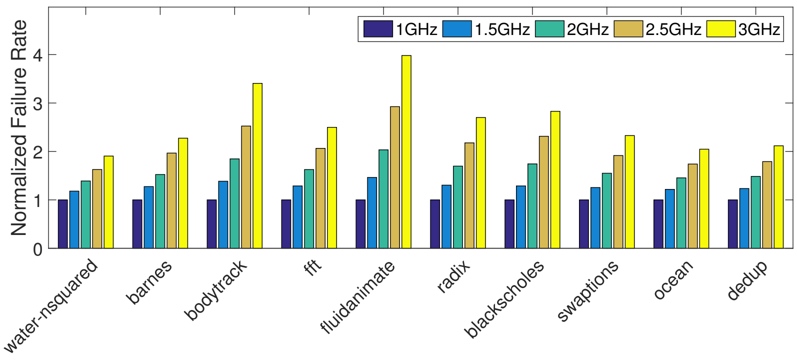


Figure 5.10: Normalized failure rate vs. clock frequency for different benchmarks (number of active cores = 16)

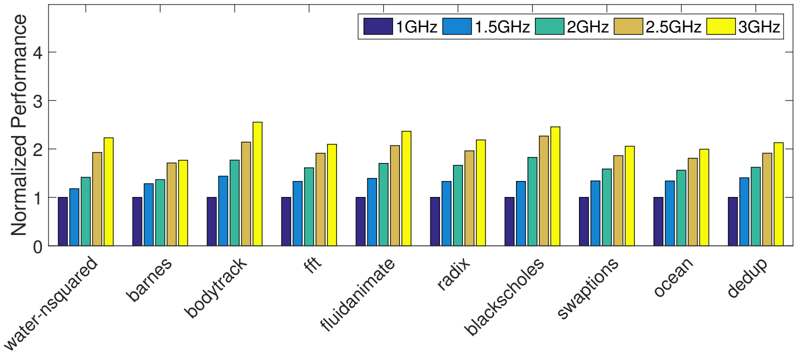


Figure 5.11: Normalized performance vs. clock frequency for different benchmarks (number of active cores = 16)

cores at 3GHz operating frequency. This figure illustrates that while reducing the number of active cores will reduce the failure rate, this action may not always lead to the decrease in performance. This phenomenon indicates that if we reduce the number of active cores, we can get better reliability while still maintaining high performance.

5.5 Q-learning Based Dynamic Reliability Management

In Section 2.6, we have introduced different types of dynamic management techniques. In this thesis, we will develop a learning-based dynamic management technique. In Section 5.5.1, we will introduce the on-line data collection methods used in learning-based management. In Section 5.5.2, we will describe a tabular-based learning based management technique – Q-learning technique and our proposed technique will be introduced in Section 5.5.3 and 5.5.4.

5.5.1 On-line Data Collection

During the run-time of programs, the learning agent keeps collecting data from the system and environment. In this section, we will introduce how data are collected (or estimated) during the run-time.

Performance detection and power estimation. The performance of the multi-core processor can be detected using performance counters integrated in each core of the processor. The power consumption profile of the multi-core processor can then be estimated with these performance counters [87].

Temperature estimation/detection. With the knowledge of the power profile, the temperature can be estimated using the following equation [88, 89]:

$$\frac{1}{R}T + C\frac{dT}{dt} = P, \tag{5.5}$$

where T is the temperature; R and C, respectively, are the matrices representing the chip's thermal resistance and capacitance which can be acquired through the off-line experiment; P is the power consumption. Alternatively, the temperature profile can be detected with integrated thermal sensors [90].

Failure rate estimation/detection. Basically, the failure rate of the multi-core processor is estimated with associated reliability models [40,91,92]. Sometimes, the DRM system needs to collect the information of the real degradation (*i.e.* the cumulative failure effect) to provide better management. This can be achieved with different types of aging sensors hard-wired in the system [93, 94]. Note that, any hard-wired sensors (e.g. for temperature or degradation detection) will occupy architecture resources (e.g. cache), thus increasing the cost of the chip and perhaps introduce more noise to the processor. Optimization of the design and placement of these sensors is beyond the scope of this thesis. However, users can decide the usage of hard-wired sensors according to their budget and other practical requirements.

5.5.2 Basics of the Q-learning Based Management Technique

Learning based methods will observe the behavior of the processor during the run-time and learn the optimal policy through trial and error interactions with the processor. Usually, this type of learning schematic is called **Reinforcement Learning (RL)**. Figure 5.12(a) illustrates how reinforcement learning methods are used for processor systems. Programs are executed on the processor. The learning agent selects the working modes for the processor at a fixed interval (called decision epoch). In the context of multi-core processor, a working mode is determined by the number of active cores and the performance state (*e.g.* the voltage and frequency

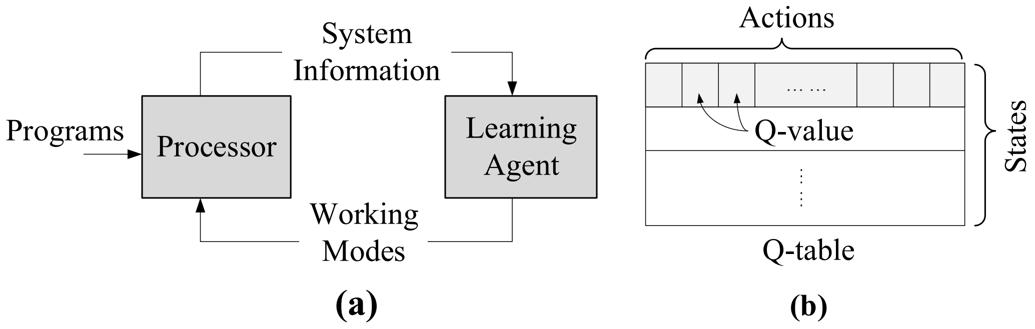


Figure 5.12: (a) The schematic of reinforcement learning based dynamic management method and (b) the structure of Q-table

etc.) of each core. By selecting working modes, the learning agent actually performs clock gating or DVFS for each active core. At each decision epoch, the learning agent also observes the information from the processor (*e.g.* the performance counters, temperature from thermal sensors *etc.*) which is evaluated for future management.

Among all learning-based dynamic management methods, Q-learning is one of the most widely used techniques. Q-learning method is originally designed to find the policy for Markov Decision Processes (MDPs) and has been proved to be able to achieve the optimal policy. In fact, the processor system for dynamic management is usually non-Markovian due to the long-range similarity of the system [95]. However, owing to its simplicity and robustness to endure noise, Q-learning based methods are still very popular in dynamic management. But the reinforcement learning methods in a non-Markovian environment is still an open problem [95]. Therefore, different types of Q-learning methods have been proposed.

In general, the Q-learning method maintains a look-up-table (called Q-table) in which the row represents the state (s, which is defined differently in different work) and the column represents the action (a) taken on certain states. A Q-table is illustrated by Figure 5.12(b). Taking an action on a state will result in a certain quality value (called Q-value) which is calculated according to the specific problem; and the Q-table (with size $|S| \times |A|$, where S and A represent the set of states and actions, respectively) is used to store the Q-values. Generally speaking, there are two types of Q-learning methods used for dynamic management, which will be introduced as follows.

5.5.2.1 Type A: Transition-driven methods

In this type of Q-learning methods, the state is defined as the working mode of the processor (*i.e.* the number of active cores and the power state of each core) while the action is defined as the transition from one state to another state. Q-values are used to evaluate such transitions. For example, Kim *et al.* [2] used this type of techniques to optimize the energy efficiency of multi-core processors subject to reliability, thermal and performance constraints. The transition of working modes is evaluated according to the change of performance, temperature and reliability *etc.* which are adopted during the run-time. This method defines a penalty term (PT) that penalizes the transition when the reliability or the temperature violation is too high to the performance is too low. In this case, PT is positive. The detailed definition of PT can be found in [2]. Then the PT of the corresponding transition is calculated using the following equation:

$$Q^{t+1}(s(t),a(t)) = (1-\alpha_Q(t))\times Q^t(s(t),a(t))+\alpha_Q(t)\times(PT(t+1)+\gamma_Q \min_a(\forall Q^t(s(t+1),a))) \tag{5.6}$$

Here, Q^t, $s(t)$ and $a(t)$ represent the Q-table, working mode and transition at time t, respectively. $PT(t+1)$ represents the penalty at time $t+1$. As illustrated by Equation 5.6, the new Q-value is updated by taking the linear combination between the current Q-value and the penalty (PT). Note that, the term $\min_a(\forall Q^t(s(t+1),a))$ evaluates the possible future reward (represented by the Q-value) if a certain transition is taken, and γ_Q is the parameter weighing the importance of such future influence on the current decision.

In this work, the initial state (*i.e.* the working mode) of the processor is selected randomly while the following working modes are chosen with the largest Q-value. The performance, temperature, reliability *etc.* are evaluated at the end of each decision epoch. According to [2], this method works perfectly when the program diversity is low. However, it mainly has two problems:

- When the diversity in a program is large this method may fail to provide efficient management. For example we have programs that iteratively perform CPU-bound and memory-bound tasks. Obviously, the processor configuration should be different for these two tasks. However, when applying this type of Q-learning method, the system will finally converge to a configuration that balance the performance of both tasks. In other words, the learned policy is suboptimal. This can be regarded as an example that the reinforcement learning method fails to learn the optimal policy in non-Markovian environment.

- The size of Q-table increases quadratically with the number of different processor configurations (*i.e.* working modes), which might occupy extremely large memory resources. To make things worse, following the schematic of this type learning method, the only way to reduce the table size is to decrease the number of working modes used for management. However, this will degrade the management capability.

5.5.2.2 Type B: Behavior-driven methods

Different from the previous type of Q-learning method, this type of methods defines the Q-table state (s) as the behavior of the processor. The action is also defined as the working mode. The behavior include all data that can be detected through sensors and embedded performance counters. Temperature, power, number of committed cycles within a certain period of time *etc.* are all behaviors which can be used to define the Q-table state [1]. Typically, a processor behavior (*e.g.* temperature) is continuous. Therefore, we need to quantize it (*i.e.* divide it into intervals) before using it to define the Q-table state. Examples of such type of Q-learning methods can be found in [1, 7]. In [1], Das *et al.* defined the Q-table state as the composition of the average thermal stress and the thermal cycling induced degradation. In [7], processor utilization was used as the Q-table state.

The Q-value update procedure of this type of methods is similar to the previous type. With the calculated penalty, the Q-value is updated using Equation 5.6 in some work. However, according to Das *et al.* [1], if we can properly predict the next Q-table state, we can directly set λ_Q in Equation 5.6 to zero. In this way, the Q-value is updated without considering the future influence. Compared to the previous type of methods (Type A), the Q-table size of this type is determined by both the number of working modes and the number of quantized processor behaviors. Therefore, the Q-table size can be controlled by choosing different granularity for quantizing the behaviors [1]. However, similar to the previous type, if the quantization of behaviors is too coarse, although we can save the memory space, the management efficiency will be degraded. Another problem of this type of methods is that the processor behavior will be affected by the configuration of processors. For example, the same behavior can result from the execution of two completely different pieces of code with different working mode. However, this type of methods will regard them as the same Q-table state and select the new working mode corresponding to this Q-table state. But, since the underline codes are different, the new working mode may lead the system to two different behaviors (and perhaps different Q-table states). Obviously, this may cause the learning agent to oscillate between policies thus causing suboptimal management.

5.5.3 Another Type: Phase-driven method

In this work, we propose another type of Q-learning based dynamic management method: **Phase-driven method**. This type of methods is motivated by the fact that the execution of tasks on processors is constituted of phases. A phase is defined as a set of intervals within the program that have similar behaviors while the behavior between phases can be very different. The phase behavior (*e.g.* the number of instructions per cycle, the power *etc.*) is a function of the way the code is being executed. Running a piece of code with different configurations of processors may lead to different behaviors, therefore using phase behavior to represent the phase will cause inefficient management (as introduced in the previous section). On the other hand, if we can track the change of proportions of code and use this to define the Q-table state, the management may become more efficient. Fortunately, such on-line phase detection method has been widely studied [5, 96]. In this way, we can utilize the existing phase detection technique in the Q-learning management method thus leading to our proposed technique. In the following part of this section, we will first introduce one popular phase detection technique proposed by Sherwood *et al.* [5] and then describe our Phase-driven Q-learning based method.

5.5.3.1 Phase Detection Technique

We use the method proposed by Sherwood *et al.* [5] to detect the phase in a program. This method is chosen because it is one of the most popular techniques that detect and track phases of programs according to the execution of the proportion of codes rather than the behavior of programs as most phase detection methods do [7, 95]. In this section, I will give a brief introduction to this technique.

This technique detects the program phase by tracking the basic block information during the run-time of the program. The basic block information is represented

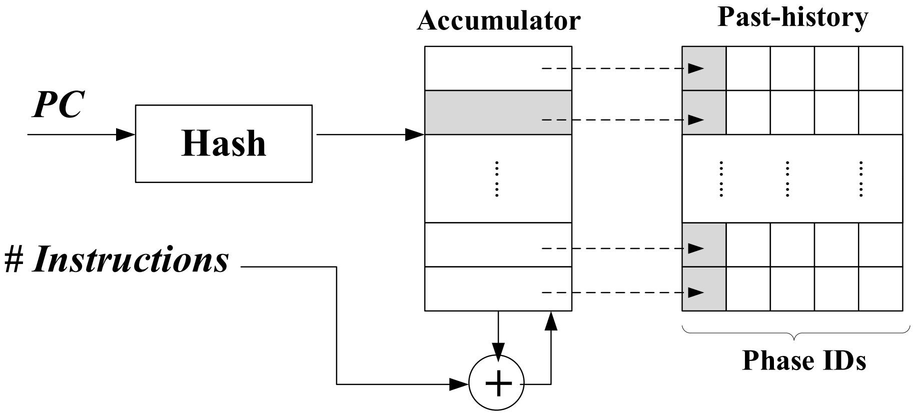


Figure 5.13: Illustration of the phase detection technique [5]

by the program counter ($\mathcal{PC}$) of each committed branch and the number of instructions ($\mathcal{I}$) between the current and the previous branches. As illustrated in Figure 5.13, the $\mathcal{PC}$ goes through a hash function and the result is used to index an accumulator table with a certain number of buckets (*e.g.* 32 buckets). Each bucket contains one counter which is incremented by the number of instructions since the last branch. The accumulating procedure is performed at the CPU speed. After a certain number of instructions (*e.g.* 10 million), the accumulated branch block information is used to form a footprint which is then used to classify phases. There are some techniques compressing the size of footprint that are introduced in [5]. After the footprint is formed, it is then used to compare with a set of historical footprints which are stored in a past-history-table. If the new footprint matches one entry of the table (*i.e.* the hamming distance between them is within a threshold) , this code interval will be classified into the phase represented by that entry. Otherwise, a new phase is discovered and a new entry in the past-history-table is created to store this new footprint.

5.5.3.2 Q-learning Procedure

In this section, we will introduce the Phase-driven Q-learning based dynamic reliability management (DRM) procedure using the phase detection technique introduced in the previous section.

The proposed DRM technique is designed to maximize the processor performance subject to the reliability constraints due to a wide range of failure mechanisms (*e.g.* TDDB, EM, TC *etc.*). Different from the two methods introduced in Section 5.5.2, in our technique, each "state" represents a phase (τ) detected by the system (*e.g.* using [5]) and each "action" represents a working mode (*i.e.* the combination of the number of active cores, frequency of each core *etc.*). Therefore, the Q-value of a certain entry of the Q-table ($Q(\tau, w)$) indicates the reward gained by executing the phase τ with the working mode w. A Q-table in our technique is illustrated in Figure 5.14(a). In the following part, we will elaborate the calculation of the reward and Q-value.

In this technique, both the detection of phases and the management of the processor are performed at the granularity of the decision epoch. At the beginning of the DRM (① in Figure 5.14), we start to process the first decision epoch. At this time, since the learning agent does not know the phase for this decision epoch, it randomly assigns a working mode (*e.g.* w_1). During the execution of the first decision epoch, the ID of the piece of code (*i.e.* the footprint) within this epoch is calculated; the average performance and failure rate within this period is also evaluated. Performance can be evaluated using the performance counters while the

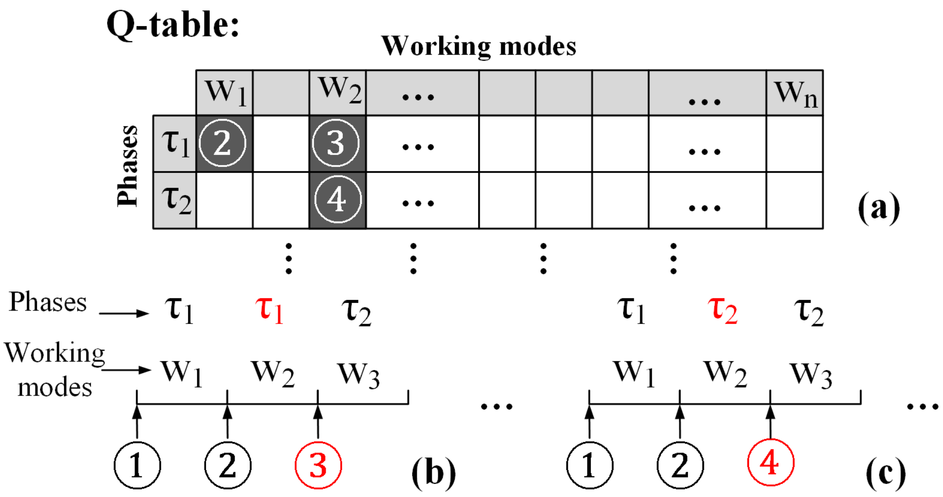


Figure 5.14: (a) Illustration of a Q-table in our technique; (b)(c) two possible cases of management flow (*i.e.* the actual phase for the second decision epoch is different)

failure rate is calculated using reliability models (*e.g.* [2]) according to specific DRM problems. At the end of the first decision epoch (② in Figure 5.14), the phase for this epoch can be detected (*e.g.* τ_1) and the corresponding Q-value (*i.e.* $Q(\tau_1, w_1)$) is thus updated using the performance and failure rate evaluated during this epoch. The reliability modeling and the update of the Q-value will be introduced as follows, respectively. At the end of the first decision epoch, the phase for the next decision epoch is assumed to be the same as the currently detected phase (*i.e.* τ_1) and the working mode is selected accordingly. When the decision epoch is shorter than a phase, this phase prediction method is efficient. However, our method can be replaced by any other predictors which may yield more accurate prediction [97]. At the end of the second decision epoch, the ID of the piece of code during this epoch is acquired thus we can determine the phase for this epoch. If it is a phase that was met before (*e.g.* τ_1), the learning agent locates the row for the phase in the Q-table and the corresponding Q-value is updated. This is the case illustrated in Figure 5.14(b). Otherwise, a new row in the Q-table is created and the corresponding Q-value is updated (as illustrated in Figure 5.14(c)). Note that, when the prediction of the phase is incorrect, there is a management overhead. However, when the length of a decision epoch is smaller than the phase, this overhead is insignificant compared to the performance improvement of our technique.

Reliability Modeling

Our method can support a wide range of failure mechanisms, such as EM, TDDB etc. The failure rate of a certain mechanism can be evaluated using the reliability model (*e.g.* [98]). And the failure rate of the processor is the combined effect of all the failure mechanisms:

$$\lambda_{tot} = \sum_{r=0}^{N_r} \lambda_r, \tag{5.7}$$

where N_r is the number of different failure mechanisms and λ_r is the failure rate of the r^{th} failure mechanism.

As introduced in [99], a processor usually requires a minimum lifetime (denoted as M^*). In order to meet this constraint, our proposed technique enforce the failure rate at any arbitrary time to be no larger than the failure rate constraint: $\lambda^* \propto \frac{1}{M^*}$. More details of the reliability modeling can be found in [99].

Reward Calculation and Q-value Update

At the end of each decision epoch, the average performance and the failure rate for this epoch can be evaluated. These two values are denoted as $IPS(\tau, w)$ and $\lambda(\tau, w)$, respectively. Then we can use the following equation to calculate the reward for this phase-working-mode pair, (τ, w):

$$r(\tau, w) = \begin{cases} IPS(\tau, w) - \theta(\lambda(\tau, w) - \lambda^*), & \lambda(\tau, w) > \lambda^* \\ IPS(\tau, w), & \lambda(\tau, w) \leq \lambda^* \end{cases} \tag{5.8}$$

In this equation, the reward is simply the performance if the failure rate is no larger than the constraint. Otherwise, the reward value is calculated as the performance subtracted by a penalty proportional to the slack between the failure rate constraint (λ^*) and the real estimated failure rate ($\lambda(\tau, w)$). After getting the reward value, we can update the corresponding Q-value ($Q(\tau, w)$) using the following equation:

$$\begin{aligned} &Q^{new}(\tau, w) = \\ &\begin{cases} r(\tau, w), & r(\tau, w) < 0 \\ (1-\eta)(Q^{old}(\tau, w)) + \eta(r(\tau, w)), & r(\tau, w) \geq 0 \end{cases} \end{aligned} \tag{5.9}$$

In this equation, $Q^{old}(\tau, w)$ and $Q^{new}(\tau, w)$ are the old and new Q-values, respectively. When the reward value is positive, the old Q-value is updated with the reward value multiplying a learning factor, η. However, if the reward value is negative, the new Q-value is automatically set negative which prevents the learning agent to select this phase-working-mode pair with large reliability penalty.

Selecting Working Modes

In this DRM technique, the working mode for the next decision epoch is selected according to the learning stage of the predicted phase. For each phase, there are three learning stages: Exploration Stage, Exploration-Exploitation Stage and Exploitation Stage. The learning stage is determined by the number of explored working modes for a phase. A working mode for a phase is explored indicates that the phase has been executed at least once with that working mode:

- **Exploration Stage:** We are at this stage when only a few working modes are explored. During this stage, the learning factor (η) stays constant as η_0 and the working mode is selected randomly while more weight is placed to selecting the unexplored ones.

- **Exploitation Stage:** When the number of explored working modes for a phase is large, we will enter this stage. During this stage, the working mode with the highest Q-value for this phase is selected (*i.e.* the learning parameter $\eta = 0$).

- **Exploration-Exploitation Stage:** This is a transition stage between the previous two stages. During this stage, we randomly choose between the policies for the exploration and exploitation stage to select the working mode for a phase. The probability of using the policy for the exploitation stage increases with the number of explored working modes for the phase. Meanwhile, η will decrease linearly from η_0 to 0 with the number of explored working modes increases.

It should be noted that in our proposed technique, different phases may be in different learning stages.

5.5.4 Enhancement modules to the method

In Section 5.5.2, we have introduced the basic algorithm of our phase-driven Q-learning based DRM technique. Another benefit of defining the Q-table state with phases is that we can exploit several enhancement techniques to further improve the management efficiency. In this section, we will propose four enhancement modules that can be integrated into the basic algorithm: (1) On-line Clustering Module, (2) On-line Population Module, (3) Thermal-aware Relearning Module and (3) Failure-rate Update Module. The four modules will be introduced in detail as follows.

5.5.4.1 On-line Clustering Module

According to the basic algorithm, the Q-table size is determined by the number of working modes ($|\mathcal{W}|$) and the number of different phases ($|\mathcal{T}|$). However, when $|\mathcal{T}|$ is large, there is a risk that our Q-table occupies a large memory space. Fortunately, experiments show that different phases may have similar distributions of performance and failure rate with respect to the working mode. This motivates us to cluster phases with similar such distributions to reduce the memory space. Further more, we can fix the size of the Q-table and adopt cache policy to deal with phase congestion. The proposed On-line Clustering Module is stated as follows:

1. **Defining the Q-table Size.** In the initialization stage, the Q-table is crated with a fixed size, $N_\tau \times |\mathcal{W}|$, instead of $|\mathcal{T}| \times |\mathcal{W}|$. Here, N_τ is a constant value indicating the number of rows of the Q-table and is determined according to the memory budget of the system. Usually, $N_\tau < |\mathcal{T}|$. Meanwhile, a look up table is built for mapping each phase to the row storing its Q-values in the Q-table. This look-up-table can be concatenated with the past-history-table as introduced in Section 5.5.3.1 (Figure 5.13). Note that, when $|\mathcal{W}| \gg 1$, this

method only needs roughly $\frac{N_\tau}{|\mathcal{T}|}$ of the original memory space.

2. **Clustering Phases.** For the phases stored in the Q-table, they can be clustered according to the similarity of performance and failure rate when running with each working mode. However, we cannot wait until all the working modes are explored to determine the similarity between phases. Therefore we propose a prediction technique. We select a set of "sampling working modes (SWM)" which are scattered in the working mode space and record the performance and failure rate for each phase (that stored in the Q-table) executed with the SWM during the Exploration Stage. After a number of decision epochs, the phases that satisfy the following two conditions are clustered: (1) they have similar performance and failure rate when executed with the SWM and (2) they have the same **current** optimal working mode. If the phases are clustered, we average the Q-values of each working mode across these phases. In order to further cluster this phase-cluster with other phases in the future, we generate the performance and failure rate values of the SWM for this phase-cluster by averaging the values of the related phases.

3. **Replace Policy.** If $N_\tau < |\mathcal{T}|$, when a new phase cannot find its space in the Q-table. We need to replace it with a phase in the Q-table which is least explored and update the look up table accordingly. This is similar to the least frequently used (LFU) cache.

Besides its benefits to saving the memory space, clustering phases will have complicated influence on performance. On one hand, when several phases are clus-

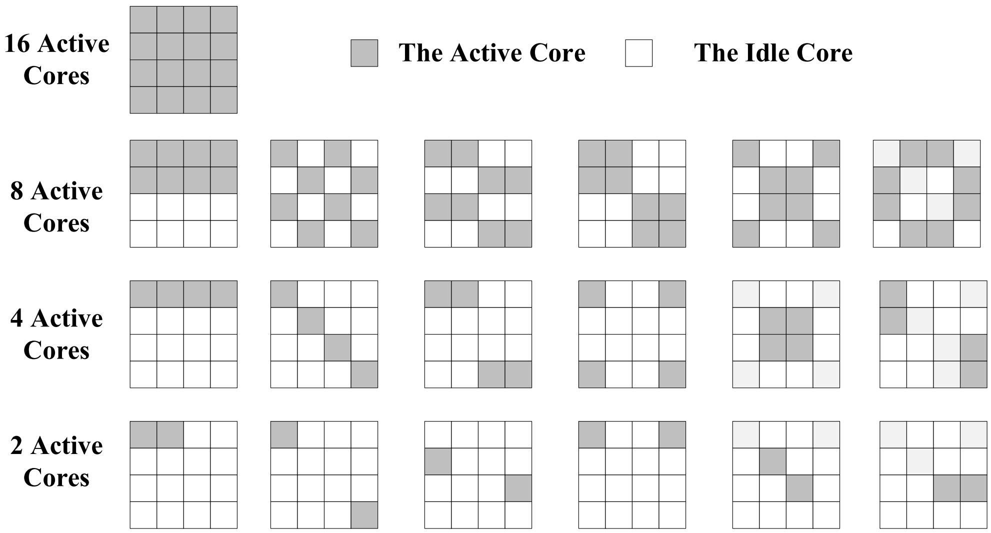


Figure 5.15: Illustration of different patterns of active cores (for a 16-core processor)

tered, we only learn one policy for the phase-cluster which is not guaranteed to be optimal for all phases in this cluster. Hence the performance may be degraded. On the other hand, on-line clustering can help reducing the Q-table size (in the "state" dimension) thus can speed up the process of learning the optimal policy for all phases/phase-clusters. And this will be beneficial to the performance.

5.5.4.2 On-line Population Module

Large Q-table size dose not only occupy the memory space, but also degrades the learning efficiency since more working modes should be explored before convergence. In order to increase the speed of learning the policy, we need to reduce the Q-table size. Section 5.5.4.1 introduces a method to cluster phases hence compressing the Q-table in the "state" dimension. However, the number of working modes can be very large, especially in the context of multi-core CPUs, and the phase-clustering method cannot solve this problem. Therefore, in this section, we propose an **On-line Population Module** to address the challenge brought by the large number of working modes.

This population method is motivated by the discoveries introduced in Section 5.4 and is used to speed up the exploration of working modes for each state (phase). According to the investigation and results in Section 5.4, when all the active cores have the same operating frequency, the performance and power changes roughly linearly with the frequency. In our proposed Q-learning technique, each Q-value is a linear function of performance and failure rate. Actually, the failure rate is only a penalty which will influence the Q-value only when the failure rate constraint is violated. Therefore, we assume the Q-value is a linear function with frequency for a fixed number of active cores even though the function between power and failure rate is complicated and definitely nonlinear. Based on this, we propose to build a **set** of linear models for each phase to predict the Q-value of working modes before they are explored. Here is the description of the algorithm:

- **Building the models.** At the run-time of the DRM process, the On-line Population algorithm analyses the Q-value for each phase. For a phase, if there are at least two **positive** Q-values such that their associated working modes have the same number of active cores (regardless of the distribution of the active cores) but different frequencies, a linear model of the Q-value with respect to the frequency is thus built for this phase and the specific number of active cores. There are two points to be noted: (1) this algorithm ignores the deviation between different active-core patterns (*e.g.* Figure 5.15);

(2) negative Q-values are not used to build the model since their associated working modes cause the reliability constraint violation.

- **Predicting the Q-value.** After building the model, we can predict the Q-value of the unexplored working modes for this phase with the same number of active cores. In this way, we increase the population speed of the Q-table. Since we only use positive Q-values to build the model, the predicted Q-values should also be positive. Note that, some of the predicted working modes will actually violate the reliability constraint (hence the corresponding Q-value is negative). In the next step, we will describe how this problem is addressed in our algorithm.

- **Update Q-values.** After the Q-value of a working mode is predicted, it is possible that this working mode is actually explored in the future such that the real reward (Equation 5.8) is calculated. In this case, the Q-value of this working mode will be updated. If the real reward is negative, the Q-value will be directly set negative as introduced in Equation 5.9.

5.5.4.3 Thermal-aware Relearning Module

Our proposed learning based DRM technique contains three stages. When the Exploitation Stage is reached, the Q-table will no longer be updated because we assume the optimal policy has been discovered. However, in practice, the environment may change, which necessitates restarting the learning process in order to provide efficient management. One of the most important factors is temperature.

Temperature is a relatively slow process compared to the management interval. The temperature can stay roughly constant for several seconds during which hundreds or even thousands of decision epochs may be processed and some phases may already entered the Exploitation Stage. At this stage, if the temperature is high, some high performance working modes will become “locked” (*i.e.* with negative Q-values) due to the exceeded failure rate. Following this, the management agent will tend to select low-performance working mode which leads to the decrease of temperature. The decreased temperature may reduce the failure rate, thus unlocking some high-performance working modes. However, since the phase is already in the Exploitation Stage, we cannot select those high-performance working modes since their Q-value is negative (Section 5.5.3.2). In this case, we will lose a chance to further improve the performance. In order to fix this problem, we propose a **Thermal-aware Re-learning Module** in this section to restart the learning procedure according to the temperature.

In this module, we integrate a temperature prediction model to the Q-learning based DRM technique. Thermal prediction techniques have been extensively studied in Dynamic Thermal Management (DTM) [27,100,101]. However, to our knowledge, there is no existing work that integrates the temperature prediction to reinforcement learning based dynamic management techniques (either DTM or DRM). Instead, the most common technique for the reinforcement learning based DTM is to define the state (*e.g.* the state of the Q-table) as a certain range of temperature and learn the optimal action for each state. As described in Section 5.5.2, such definition of the state may not provide efficient management. Therefore, in this work, we maintain the Q-table structure for the phase-driven based DRM technique (Section 5.5.2) and use an integrated temperature prediction model to predict the future temperature of each core based on the historical temperature.

Temperature Prediction Model

We use an autoregressive moving average (ARMA) model to predict the future temperature of cores using the observed temperature data in the past. ARMA models are widely used in many fields for forecasting the behavior of a time series from past values. ARMA models have been applied in DTM techniques to predict future temperatures with high accuracy [27]. In this work, we slightly simplify the ARMA-based thermal predictor such that the order of the moving average (MA) parts of the model is zero and only the autoregressive (AR) parts are included. Therefore, at decision epoch t_0, the temperature of a core at the $t_0 + q$ epoch is predicted as:

$$T(t_0 + q) = \sum_{i=0}^{N_p - 1} \phi_{(q,i)} T(t_0 - i) + \epsilon_{t_0}. \tag{5.10}$$

In this equation, $T(t_0 - i)$ is the temperature of the core observed at $t_0 - i$ epoch; N_p is the number of historical temperature data used for the prediction and $\phi_{(q,i)}$ is the weight of each historical temperature. ϵ_{t_0} is white noise which is taken as 0 in our work.

At the beginning of the simulation, the weights are initialized uniformly and the predicted temperature (*e.g.* T_{t_0+q}) is usually very inaccurate. When the real temperature data (*e.g.* $\hat{T}(t_0+q)$) is observed, we can calculate the error and update the weights by

$$\phi_{(q,i)}^{new} = \phi_{(q,i)}^{old} - \alpha_{pt} \times \frac{T(t_0+q) - \hat{T}(t_0+q)}{T(t_0-i)}, \tag{5.11}$$

where α_{pt} is the scaling factor which controls the learning speed. In this way, the thermal predictor adaptively modifies according to the observed temperature. According to the experimental results (Section 5.7.1.2), after a few epochs, the prediction error is within 1% when predicting the temperature of one epoch in the future.

Restart Learning

Before the management, we determine N_p and N_q. N_q is the number of decision epochs in the future where the core temperature is predicted; N_p is the number of decision epochs in the past where the temperature is used for prediction. For example, if we are at the end of the t_0 decision epoch, the core temperature of $\{t_0 - N_p + 1, t_0 - N_1 + 2, ..., t_0\}$ decision epochs is stored and used to predict the core temperature for the $\{t_0+1, t_0+2, ..., t_0+N_q\}$ decision epochs. After the temperature is predicted, we can evaluate the changing trend of the temperature. If we discover a significant drop in temperature (*i.e.* the following inequation is satisfied), we will restart the learning procedure by randomly exploring the working modes with negative Q-values.

$$\frac{1}{N_q}\sum_{j=1}^{N_q} T_{max}(t_0+q) < \max_{i\in[0,N_p-1]} T_{max}(t_0-i). \tag{5.12}$$

In this inequation, $T_{max}(x)$ represents the maximum core temperature among all cores at the x decision epoch.

5.5.4.4 Failure-rate Update Module

In our dynamic reliability management problem, the reliability constraint requires the **average** failure rate to be smaller than $\lambda^* = \frac{1}{M^*}$ (as introduced in Section 5.5.3.2). The average failure rate ($\bar{\lambda}$) is calculated using Equation 5.13 [26] supposing the actual failure rate for decision epoch i is λ_i.

$$\frac{1}{\overline{MTTF}} = \bar{\lambda} = \frac{1}{N}\sum_{i=1}^{N} \lambda_i. \tag{5.13}$$

However, during the process of management (Section 5.5.3.2), the working mode is selected and evaluated using the "transient" failure rate during each decision epoch. This means for eah λ_i, our algorithm enforces $\lambda_i < \lambda^*$. Although this guarantees that $\overline{MTTF} > M* = 1/\lambda^*$, the gap between $\overline{MTTF}$ and M^* may increase with time, thus leaving the potential to select high performance working modes to improve the throughput of the processor. Unfortunately, since the learning agent is unaware of the accumulative failure effect, it cannot make such aggressive selection.

In order to solve this problem and improve the processor performance, we propose to use aging sensors (as introduced in Section 5.5.1) to monitor the accumulative failure effect of cores. We develop a **Failure-rate Update Module** which uses the data from the aging sensors to update the failure rate constraint (as introduced in Section 5.5.3.2). After a number of decision epochs, the Failure-rate Update Module reads the aging data from the sensor (*i.e.* $MTTF_{sensor}$) and compares it with the reliability constraint M^*. There are three possibilities:

1. If $MTTF_{sensor} = M^*$, we do not update the failure rate constraint.

2. If $MTTF_{sensor} > M^*$, we increase the failure rate constraint (*i.e.* λ^*) and we will randomly reevaluate the phase-working-mode pairs with negative Q-values in the following decision epochs.

3. If $MTTF_{sensor} < M^*$, we decrease the failure rate constraint (*i.e.* λ^*). Note that, since the constraint becomes stringent, our learning procedure will automatically correct the Q-values according to Section 5.5.3.2.

5.6 Settings of the Experiment

5.6.1 Problem Description

In this section, we will use our proposed Phase-driven Q-learning based dynamic reliability management technique to tune the process of a 3D stacked memory-on-logic structure as introduced in Section 5.2. And we will use our method to address the reliability problem induced by the EM in P/G TSVs, which is one of the main reliability problems in this 3D CPU structure. The degradation of P/G TSVs will degrade the power delivery in 3D CPUs and harm the performance and reliability of the whole system. This reliability model is introduced in Section 5.5.4.4

DRM Problem Statement: The DRM problem for this experiment is formally stated as follows: Given the programs executed in the 3D CPU, we will determine the working mode for the 3D CPU at each decision epoch in order to maximize the performance of the CPU (*i.e.* total number of instructions divided by

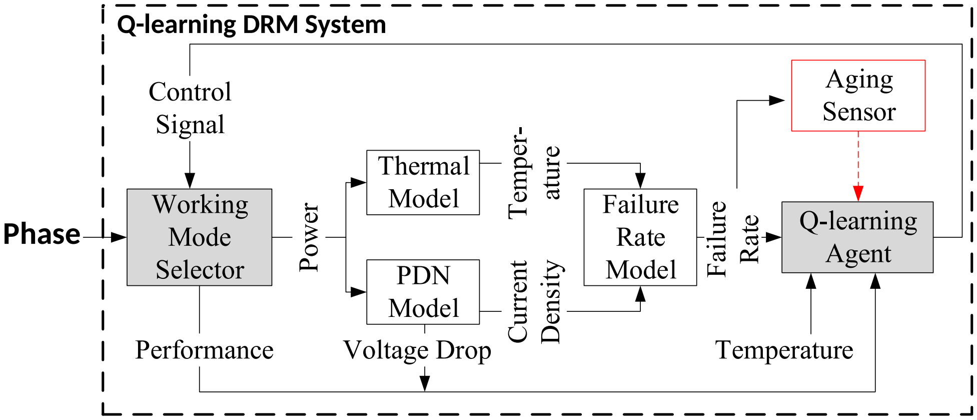


Figure 5.16: Illustration of the simulation flow for one decision epoch

the total execution time) subject to the reliability constraint for the 3D PDN.

5.6.2 Simulation Mechanism and Platform

5.6.2.1 Simulation Framework

Our proposed algorithm relies on the existing technique [5] to detect phases. In this experiment, we assume the phase can be correctly detected and the footprint of each phase is already generated. The simulation framework in this experiment was built based on this assumption. Figure 5.16 illustrates the simulation flow for one decision epoch. In this figure, “Phase” represents a phase detected in one decision epoch. In practice, the phase is detected using phase detection methods (*e.g.* [5]), however, in this experiment, we use “hyper phases”, which are generated from existing benchmarks, to represent. The generation of hyper phases will be discussed in Section 5.6.2.2.

The dark rectangles in the figure illustrate the interaction between the Q-learning agent and the 3D CPU. When a new decision epoch comes, the Q-learning agent controls the working mode selector to assign a working mode for the upcoming phase (as stated in Section 5.5.3.2, at this point the Q-learning agent does not know the actual phase for this epoch). The assigned working mode and the upcoming phase (*i.e.* hyper phase) are imported to a series of models to evaluate the performance, temperature and failure rate *etc.* during this epoch. Here, the simulation framework uses three important models: the thermal model, the PDN model and the failure rate model. Details of the PDN and reliability model can be found in Section 2.2 and 5.3.2, respectively.

The evaluated data, together with the information of the hyper phase, are fed into the Q-learning agent to update the Q-table for future management. The kernel of the Q-learning agent can be either our proposed technique or the existing methods (*e.g.* [1, 2]). The red rectangle in the figure indicates a module that simulates the aging sensor [93, 94]. This module takes the information of failure rate of each decision epoch to calculate the average failure rate of the system. Note that, if the **Failure-rate Update Module** is disabled, this average failure rate is unseen by the Q-learning agent while working as a “hidden” parameter to evaluate the reliability of the system. If the **Failure-rate Update Module** is enabled, on the other hand, the average failure rate will be required by the learning agent to perform the management as introduced in Section 5.3.2.

5.6.2.2 Constructing the Hyper Phase

In this experiment, the hyper phase is constructed using existing benchmarks. We took 15 benchmarks from PARSEC [102] and SPLASH-II [103] benchmark suits. For each benchmark, we use the phase detection method [5] to find several pieces of code within the benchmark; and we totally collected 680 different pieces of code thus giving us 680 "hyper phases". Then we evaluate the performance and power of each piece of code using Multi2Sim [104] and McPAT [76] with different working modes. The performance/power *etc.* information of these pieces of coded constitute the golden data of the phase, which is used in the framework (Figure 5.16) for evaluating the failure rate, updating the management policy and analyzing the management efficiency *etc.* . In the actual system, these data can be collected/generated from embedded sensors, performance counters or any user-defined models *etc.* .

After the hyper phases are generated, they are used to generate programs with different diversity. The diversity of a program is determined by the number of different hyper phases used to construct the program (denoted as N_L). *e.g.* Program A is constructed with 30 different hyper phases (*i.e.* $N_L = 30$) and Program B is constructed with 100 different hyper phases (*i.e.* $N_L = 100$). Then we say Program B is more diverse than Program A.

5.6.2.3 Setting the Working Modes.

In this experiment, our testing vehicle contains a processor with 16 cores. We assume all these cores belong to the same power domain while each core can be clock gated separately. Since they are in the same power domain, all the active cores have the same operating frequency which has five levels: {1,1.5,2,1.5,3}GHz. And we only consider the power-of-two number of cores to be active simultaneously. This is because that in most cases, execution with a power-of-two number of cores is more efficient than otherwise. Besides the number of active cores and operating frequency, we also consider different "patterns" for each number of active cores, which represents the distribution of active cores. Some patterns are shown in Figure 5.15. In this figure, we can see that when all the cores (*i.e.* 16 cores) are active, there is only one pattern. When the number of active cores is smaller than 16, there are several patterns. There are some relationships between patterns:

- **Complementary:** As illustrated by the last two patterns for 8 active cores. The active cores of one pattern are exactly the idle cores of the other one. When two complementary pattern (at the same operating frequency) are used alternatively, we can balance the failure rate between cores.

- **Symmetry:** Symmetry includes axial symmetry, centrosymmetry and rotation symmetry. For example, the third and the last pattern for 4 active cores are rotation symmetry. Symmetric patterns (at the same frequency) have exactly the same performance, however, the impacts on cores' failure rate are different due to different usage of cores.

- **Displacement:** As shown in the figure, the last pattern for 2 active cores is a displacement of the first pattern for 2 active cores. Similar to the symmetric relationship, if a pattern is a displacement of another pattern (at the same frequency), the performance behavior of the two patterns are the same while the failure effect is different.

In our experiment, we considered these patterns of active cores and finally selected 25 different patterns for illustration. Considering the 5 frequency levels, we have 125 working modes in total. This setting of working modes also enables us to use the **On-line Population Module**. Considering this enhancement module may not be available if the setting of working mode changes, in this experiment, we will analyze the result both with and without the **On-line Population Module**.

5.6.3 Existing Techniques – The Baseline

In this experiment, our proposed technique will be evaluated against two existing Q-learning based DRM techniques: KimTech [2] and DasTech [1]. In this section, we will briefly introduce the two baseline techniques and the parameters used in the experiment.

5.6.3.1 KimTech

Kim *et al.* [2] proposed a Q-learning based technique which belongs to the **Type A** of the methods (Section 5.5.2.1). Both the state and action of the Q-table are defined as the working modes, so the Q-table size in this technique is $|\mathcal{W}|^2 = 125 \times 125$. The working mode for the first decision epoch is assigned randomly and the following working modes are selected such that the transition from the current working mode has the minimum Q-value. At each decision epoch, the Q-value is updated following Equation 5.6 in which $\alpha_Q = 0.2$ and $\gamma_Q = 0.1$. Both α_Q and γ_Q are constant through the simulation. PT in Equation 5.6 is defined as follows:

$$PT(t+1) = PT_{perf}(t+1) + PT_{frate}(t+1). \tag{5.14}$$

In this equation, $PT_{perf}(t+1) = \frac{perf(t)-perf(t+1)}{R_{perf}}$ which evaluates the penalty of performance; R_{perf} is an estimated range of performance. $PT_{perf}(t+1) > 0$ indicates the performance reduces compared to the previous epoch (thus being penalized). $PT_{frate}(t+1) = \frac{frate(t+1)-frate(t)}{R_{rate}}$ which evaluates the penalty of reliability; R_{frate} is an estimated range of the failure rate. $PT_{frate}(t+1) > 0$ indicates the failure rate increases compared to the previous epoch (thus being penalized).

5.6.3.2 DasTech

Das *et al.* [1] proposed a Q-learning based technique which follows the **Type B** of the methods (Section 5.5.2.2). In this technique, the state of the Q-table is defined as the failure rate (λ). Since λ is continuous, we quantized the whole range of failure rate (*i.e.* $(0, \lambda^*]$ where λ^* is the failure rate constraint) into 125 intervals; any failure rate that is larger than λ_i while no larger than λ_{i+1} belongs to the i^{th} interval; any failure rate that is larger than the failure rate constraint belongs to the last (*i.e.* 125^{th}) interval. The action of the Q-table for DasTech is defined as the working mode, therefore the Q-table size is also 125×125. The working mode for the first decision epoch is assigned randomly while the working mode for the following epoch is selected with the maximum Q-value for a certain state. The Q-value is updated using the following equation:

$$Q^t(\lambda, w) = (1 - \alpha_Q)Q^{t-1}(\lambda, w) + \alpha_Q r(perf(t), \lambda(t)). \tag{5.15}$$

In this equation, α_Q is the learning parameter which decays with the exploration of the Q-table [1, 7]. $r(perf(t), \lambda(t))$ is the reward calculated using the current performance $(perf(t))$ and failure rate $(\lambda(t))$. To be specific, $r(perf(t), \lambda(t))$ is calculated as follows:

$$r(perf(t), \lambda(t)) = \begin{cases} perf(t) - \eta_{Das}^0 \times (\lambda^* - \lambda(t)), & \lambda(t) \geq \lambda^* \\ perf(t), & \lambda(t) < \lambda^* \end{cases} \tag{5.16}$$

5.7 Experimental Results and Analysis

In this section, we will perform experiments using the framework introduced in Section 5.6.2.1 to evaluate our proposed technique. Our proposed technique will be compared against the result of two existing techniques: KimTech and DasTech as introduced in Section 5.6.3.1 and 5.6.3.2, respectively. In Section 5.7.1, we will perform several experiments to evaluate our basic algorithm and the four enhanced modules. In Section 5.7.1, we will investigate the impacts of the phase diversity by running the simulation with programs of different phase diversity. In Section 5.7.2, we will investigate the influence of the phase diversity of programs on the management efficiency of our proposed technique.

5.7.1 Evaluating Enhanced Modules

As introduced in Section 5.5.4, besides the basic algorithm, we also propose four enhancement modules: (1) On-line Clustering Module, (2) On-line Population Module, (3) Thermal-aware Relearning Module and (4) Failure-rate Update Module. Therefore, we will perform different experiments to evaluate the efficiency of each module. The programs used in this experiment are constructed with $N_L = 125$ different phases. This setting guarantees the Q-table size in our technique is the same as KimTech and DasTech when the **On-line Clustering Module** is disabled (with Q-table size equal to 125×125). Since the total number of hyper phases is 680, we generated 10 different programs with $N_L = 125$. These programs are simulated separately and the average performance and reliability across these programs are reported.

5.7.1.1 Exp-1: Evaluating the On-line Clustering Module

In this experiment, we enable the **On-line Population Module** but keep the **Thermal-aware Relearning Module** and the **Failure-rate Update Module** disabled. Then we will evaluate our proposed technique with and without enabling the **On-line Clustering Module**. When this module is enabled, three different Q-table sizes are experimented: $N_\tau = \{100, 75, 50\}$ (N_τ is defined as in Section 5.5.4.1). Please note that the diversity level of the programs under experiment is $D_L = 125$ and originally the Q-table has $N_\tau = 125$. The learning parameters are defined as follows: $\theta = 3 \times 10^{-10}$ and $\eta_0 = 0.5$. Moreover, we assume the *Exploration-*

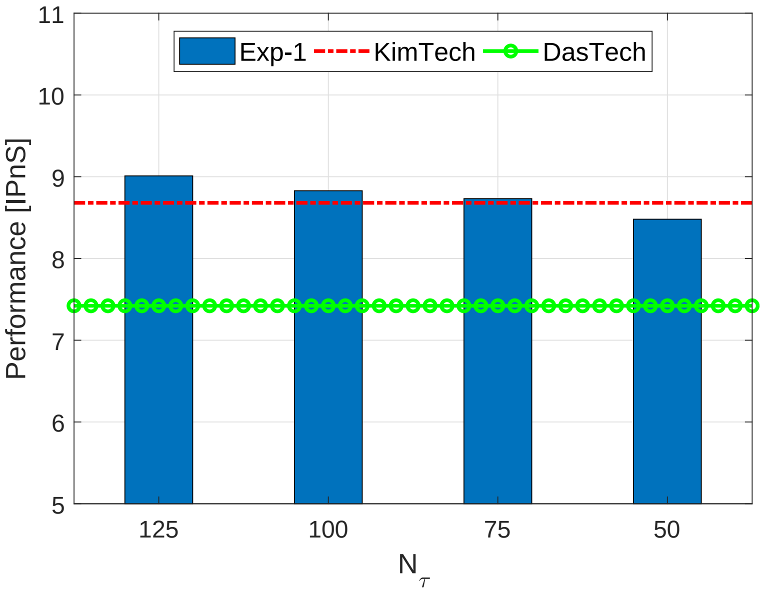


Figure 5.17: The performance (IPnS) of KimTech [6], DasTech [7] and our proposed technique with enabling the **On-line Clustering Module** (**Exp-1**) for different sizes of Q-table (indicated by N_τ)

Exploitation Stage is reached when the ratio of explored working modes for a phase reaches 40% while the *Exploitation Stage* starts when this ratio reaches 85%. The reliability constraint is set such that the minimum MTTF (M^*) due to the EM-induced failure mechanism is no less than 5 years (*i.e.* 1825 days). The thermal constraint is set as $85°C$ [28].

We use the average throughput of the 3D CPU (denoted as the number of instructions per nano-second, IPnS) through the run-time of the programs as the criterion to evaluate the management efficiency of different techniques. The results are shown in Figure 5.17. In this figure, $N_\tau = 125$ represents the case when the Q-table is not compressed (*i.e.* the **On-line Clustering Module** is disabled). The red and green lines in the figure represent the performance for KimTech and

DasTech, respectively. According to the figure, we can discover that in this experiment, KimTech works better than DasTech, and the performance of our proposed technique is a little better than the performance of KimTech. When the Q-table is not compressed ($N_\tau = 125$), our technique outperforms KimTech by 4% and DasTech by 21%. When the Q-table is compressed, the memory space can be saved. For example, when $N_\tau = 50$, 60% of the memory space for the Q-table will be saved. In this experiment, the performance degrades a little as the Q-table size is compressed. However, the performance degradation is very small compared to memory savings. When the memory saving is 40% (*i.e.* $N_\tau = 75$), the performance reduction is only 3%. Please also note that the Q-table size for DasTech and KimTech is the same as that for our technique with $N_\tau = 125$. Therefore, the results also demonstrate that by enabling the **On-line Clustering Module**, our technique can achieve similar or even better performance than KimTech and DasTech with a much smaller Q-table.

In this experiment, the **On-line Population Module** is enabled. The result with this module disabled will be discussed in Section 5.7.1.4.

5.7.1.2 Exp-2: Evaluating the Thermal-aware Relearning Module

In this experiment, we will evaluate the performance of **Thermal-aware Relearning Module**. Before that, we will first test the accuracy of our proposed thermal predictor (Section 5.5.4.3). We perform simulation on a program with $D_L = 125$. During the running of the program, we use 20 historical temperature data to predict the temperature of **one** epoch in the future with the auto-regression model introduced in Section V-B (*i.e.* $N_q = 1$, $N_p = 20$). The predicted temperature and the real temperature (which is acquired from the thermal model in the simulation framework as introduced in Section 5.6.2.1) are compared. In Figure 5.18, the real (solid blue line) and predicted (dashed red line) peak temperature are plotted. As illustrated by the figure, the predicted peak temperature matches perfectly with the real peak temperature. We also vary N_q (as defined in Section 5.5.4.3) and the corresponding prediction error for each N_q is evaluated. The result is shown in the inset table in Figure 5.18. According to the result, even when we predict the temperature for 20 epochs into the future, the error is still very small (<1%). Therefore, the thermal predictor proposed in this work is accurate enough to be used in the *Thermal-aware relearning Module*. However, users can modify this predictor or apply other predictors to this module for more accurate thermal prediction.

In order to further illustrate the effect of the **Thermal-aware Relearning Module**, we perform the simulation without clustering the phase (*i.e.* $N_\tau = 125$) and compare the temperature and performance of the processor with/without enabling the **Thermal-aware Relearning Module**. Figure 5.19(a)(b) illustrates the transient temperature and performance of the processor, respectively. This relearning module adjusts the working mode selection policy according to the change of temperature, hence the working mode can be selected more aggressively. This leads to the following results: (1) the temperature increases a little (Figure 5.19(a)) but it is still less than the thermal constraint ($85°C$). This phenomenon also indicates that the reliability constraint can be violated even if the temperature does not ex-

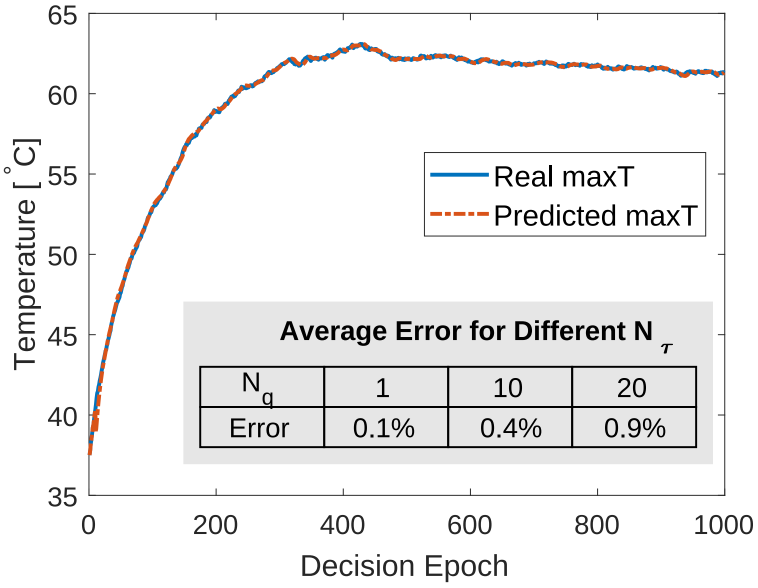
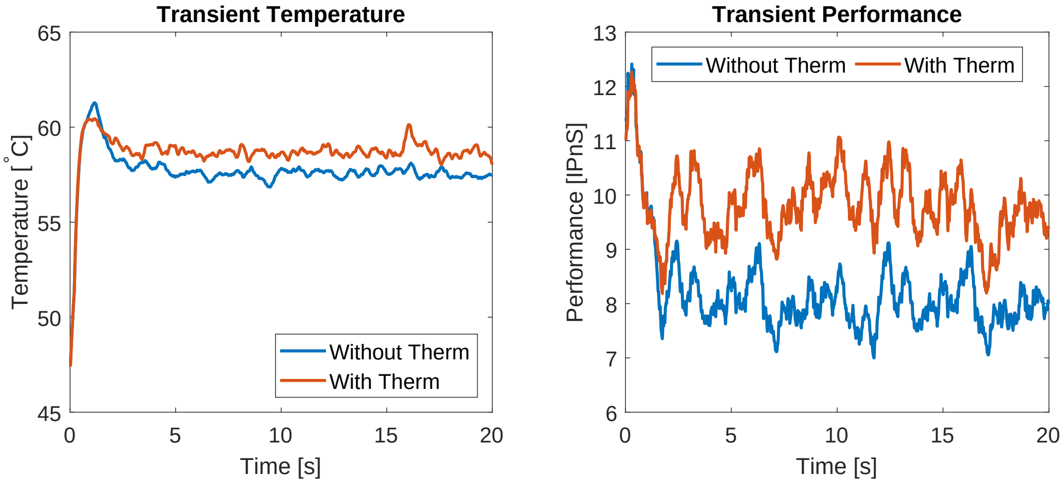


Figure 5.18: Accuracy of the thermal prediction (the inset table shows the average error of prediction for different number of epochs in the future to be predicted)



(a)

(b)

Figure 5.19: (a) The transient temperature and (b) the transient performance (IPnS) of the processor with/without enabling the **Thermal-aware Relearning Module** when the Q-table size is not compressed (**with/without Therm** indicates our technique with/without enabling the **Thermal-aware Relearning Module**)

ceed the safety line. Therefore DTM alone cannot achieve efficient DRM. (2) The performance of the 3D CPU can be maintained at a high level (as illustrated by Figure 5.19(b)). (3) The failure rate increases while the reliability constraint is still met..

In the next step, we evaluate the performance of the **Thermal-aware Relearning Module** with the influence of phase clustering. In order to achieve this, we further enable the **Thermal-aware Relearning Module** on the basis of **Exp-1** (Section 5.7.1.1). All the other settings of the experiment are the same as in **Exp-1**. The results are illustrated in Figure 5.20. In this figure, the blue bars represent the results from **Exp-1** while the red bars represent the results from **Exp-2** (*i.e.* with the **Thermal-aware Relearning Module** enabled). As illustrated by the figure, no matter how the Q-table is compressed, enabling the thermal module is beneficial in improving the performance. As illustrated by the figure, when the Q-table size compression ratio is 0%, 20%, 40% and 60% (corresponding to $N_\tau = 125, 100, 75$ and 50, respectively), the performance improvement is 1.09x, 1.05x, 1.07x and 1.10x, respectively compared to the case when the thermal module is disabled.

It should be noted that although the performance decrease as the Q-table compression ratio increases when the **Thermal-aware Relearning Module** is disabled, it is not the case when this module is enabled. According to red bars in Figure 5.20, the performance when $N_\tau = 100, 75$ and 50 is roughly the same and the performance at $N_\tau = 75$ is even a little better than $N_\tau = 100$. This phenomenon can be explained by the fact the compressing the Q-table has two opposite influence on the performance (Section 5.5.4.1). By enabling the relearning procedure, the total

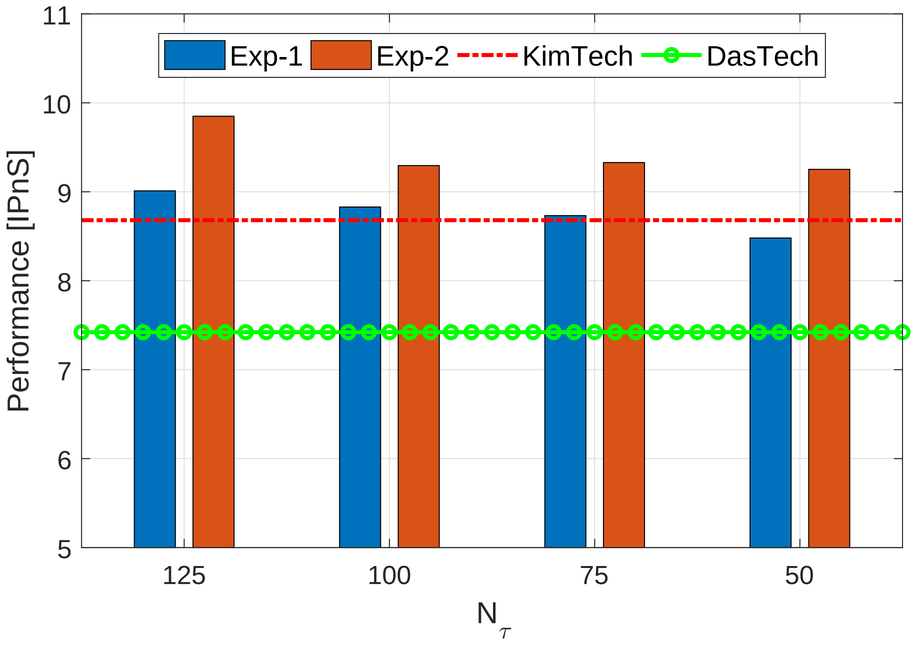


Figure 5.20: The performance (IPnS) of KimTech [6], DasTech [7] and our proposed technique for **Exp-1** and **Exp-2** with different sizes of Q-table (indicated by N_τ)

effect of compressing the Q-table may be different. Please note that the **On-line Population Module** is enabled in this experiment. The result with this module disabled will be discussed in Section 5.7.1.4.

5.7.1.3 Exp-3: Evaluating the Failure-rate Update Module

The reliability constraint of the DRM problem is enforced such that the average MTTF ($\overline{MTTF}$) should be no less than a limit (M^*). In this work, we use Equation 5.13 to calculate the $\overline{MTTF}$ while in practice this value can be collected from embedded aging sensors. In this section, we will investigate the $\overline{MTTF}$ of each technique, study how the enhancement modules affect the $\overline{MTTF}$ in our technique and finally evaluate the performance of the **Failure-rate Update Module**.

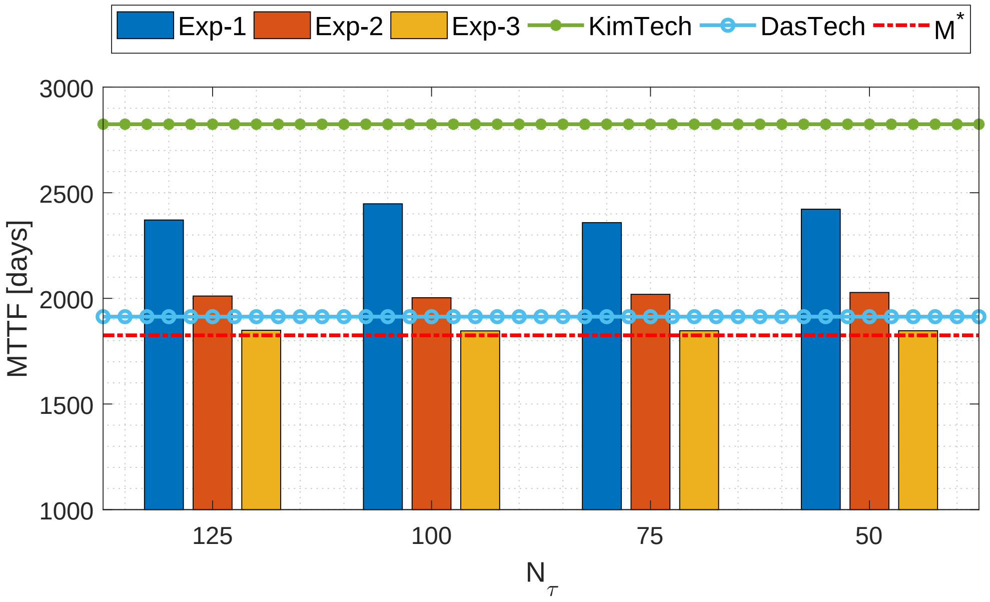


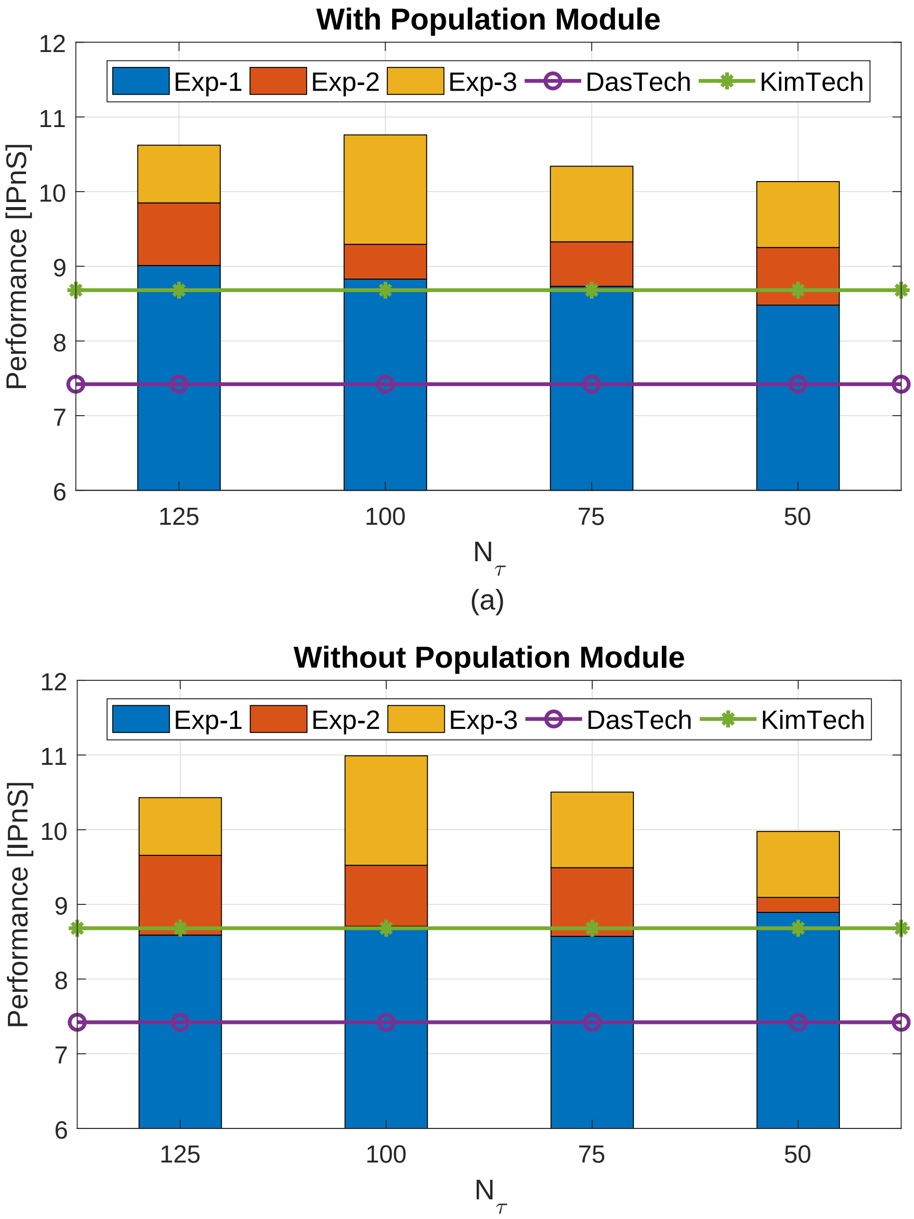
Figure 5.21: The Mean time to failure ($\overline{MTTF}$) of the processor of KimTech [6], DasTech [7] and our proposed technique with enabling different enhancement modules (**Exp-1** to **Exp-3**)

The $\overline{MTTF}$ of KimTech, DasTech and our proposed technique in **Exp-1** (Section 5.7.1.1) and **Exp-2** (Section 5.7.1.2) are shown in Figure 5.21. The MTTF constraint (M^*) is illustrated using a dotted red line. The $\overline{MTTF}$ for KimTech and DasTech are illustrated by the green and light blue lines in the figure, respectively. As illustrated by the figure, the reliability constraint is maintained in each of the techniques. For KimTech, there is a large slack between $\overline{MTTF}$ and M^*. For DasTech, even though $\overline{MTTF}$ is very close to M^*, its performance is very poor (as shown in Figure 5.17). For both technique, we can tune the parameters to gain better results. However, this action is completely empirical and the result is not guaranteed. For our technique, we can see from the figure that the slack between $\overline{MTTF}$ and M^* is reduced significantly after enabling the **Thermal-aware Re-**

learning Module. This is due to the aggressive selection of working modes and the performance is thus improved as analyzed in Section 5.5.4.3. However, there is still slack between $\overline{MTTF}$ and M^* even after the **Thermal-aware Relearning Module** is enabled.

In this experiment, we further enable the **Failure-rate Update Module** on the basis of **Exp-2** (Section 5.7.1.2) and demonstrate that this module can help improve the performance by exploiting the reliability slack. The $\overline{MTTF}$ of our technique at different Q-table compression levels is shown by the yellow bars in Figure 5.21. According to the figure, the $\overline{MTTF}$ is just slightly above M^*. With the reduction in the reliability slack, the performance can be further improved. This can be illustrated by Figure 5.22(a). The performance of KimTech, DasTech and our technique in **Exp-1**, **Exp-2** and **Exp-3** are all shown in this figure. As shown in the figure, after enabling the **Failure-rate Update Module**, the improvement in performance for different Q-table size ($i.e.$ $N_\tau = \{125, 100, 75, 50\}$) compared to the results from **Exp-2** are 1.08x, 1.16x, 1.11x and 1.10x, respectively. It is interesting to note that the performance when $N_\tau = 100$ is better than the performance when the Q-table is not compressed. The reason of this phenomenon has been explained in Section 5.7.1.1.

The performance improvement introduced by applying the **Failure-rate Update Module** relies on the usage of aging sensors. However, implementing aging sensors on chip will introduce more cost and noise to the circuit. Otherwise we can achieve this with reliability models at the risk of inaccurate modeling. However, this is beyond the topic of our work. In our technique, we just provide an interface



(a)

(b)

Figure 5.22: The performance (IPnS) of KimTech [6], DasTech [7] and our proposed technique (a) with and (b) without enabling the **On-line Population Module** for different experiments (**Exp-1** to **Exp-3**)

to use accumulative aging information for management and users can decide how to use it according to their practical requirement.

From **Exp-1** to **Exp-3**, we keep the **On-line Population Module** enabled. In the **Exp-4** (Section 5.7.1.4), we will discuss the result if this module is disabled.

5.7.1.4 Exp-4: Evaluating the On-line Population Module

In this section, we will discuss the impact of the **On-line Population Module**. Compared with other enhancement modules, this module is special because it can only be used under the condition that all the active cores are of the same operating frequency. And different reliability models may affect the performance of this module. Therefore, in this section, we perform the Exp-1, Exp-2 and Exp-3 again with the **On-line Population Module** disabled and compared the results with KimTech and DasTech. The performance are shown in Figure 5.22(b).

According the result, we can conclude that the **On-line Population Module** does have impacts on the performance, especially when both the **Thermal-aware Relearning Module** and **Failure-rate Update Module** are disabled. If we focus on the blue bars in Figure 5.22(a) and (b), we can see that when the Q-table is not compressed or just compressed a little (*e.g.* $N_\tau = 125$, 100 and 75), the population module improves the performance. On the other hand, when the compression ratio of the Q-table is large (*e.g.* $N_\tau = 50$), enabling the population module degrades the performance. We think one reason behind this phenomenon is the prediction error of the **On-line Population Module**. As introduced in Section 5.5.4.2, the population module predicts the Q-value for unexplored working modes for each “state” of the Q-table. When the Q-table is not compressed, each state represents one phase, this prediction is accurate. However, when the compression level is high, each state of the Q-table represents several phases. The prediction of Q-values for this kind of states will be less accurate, thus degrading the performance.

With the enabling of other modules, the impacts of **On-line Population Module** seems irregular. This is because the reinforcement learning based management is a stochastic process (*e.g.* randomly selecting working modes at the Exploration Stage) and enabling each module will have an influence on this process (*e.g.* affecting the selection of working modes). However, from Figure 5.22, we can conclude that at a certain compression level for the Q-table, enabling the **Thermal-aware Relearning Module** and **Failure-rate Update Module** will cause significant improvement in performance while the reliability constraint is still maintained.

5.7.2 Investigate the Impacts of Phase Diversity

In Section 5.7.1, we only use programs with $D_L = 125$ to evaluate our proposed technique and the enhancement modules. However, the phase diversity will also influence the management efficiency. If the phase diversity is high (*i.e.* D_L is large), we either need a large Q-table to perform the management or we can use the phase clustering technique to maintain a small Q-table at the risk of performance degradation. In order to investigate the impacts of phase diversity, in this experiment, we

Phase Diversity (D_L)	30	125	250	350
DasTech [1]	8.5	7.4	6.5	7.2
KimTech [2]	8.3	8.7	7.3	8.0
Proposed Tech (with Pop).	11.8	10.6	8.2	8.6
Impr. DasTech	1.29x	1.43x	1.26x	1.19x
Impr. KimTech	1.42x	1.22x	1.12x	1.08x

Table 5.2: The average performance (IPnS) of the processor under different techniques with different phase diversity levels (**Impr. DasTech** and **Impr. KimTech** represent the improvement of our technique compared to DasTech [1] and KimTech [2], respectively) with the **On-line Population Module** is enabled

perform **Exp-3** using programs with different D_L ($D_L = \{30, 120, 250, 350\}$). The Q-table size is kept constant as $N_\tau = 125$. If $D_L > N_\tau$, we need to use **On-line Clustering Module** to cluster phases. We compare our results with KimTech and DasTech and the results are reported in Table 5.2 and 5.3. In Table 5.2, we enable the **On-line Population Module** while in Table 5.3, this module is disabled. All the other enhancement modules are enabled.

According to the result, when both **Thermal-aware Relearning Module** and **Failure-rate Update Module** are enabled, the impact of **On-line Population Module** on the performance is very small (by comparing the results from Table 5.3 and 5.2). This is similar to the discovery in Section 5.7.1.4. Another discovery from the result is our proposed technique can achieve significant performance improvement compared to the existing techniques (*i.e.* KimTech and DasTech). When

Phase Diversity (D_L)	30	125	250	350
DasTech [1]	8.5	7.4	6.5	7.2
KimTech [2]	8.3	8.7	7.3	8.0
Proposed Tech (without Pop).	11.8	10.7	8.4	8.5
Impr. DasTech	1.29x	1.45x	1.29x	1.18x
Impr. KimTech	1.42x	1.23x	1.15x	1.06x

Table 5.3: The average performance (IPnS) of the processor under different techniques with different phase diversity levels (**Impr. DasTech** and **Impr. KimTech** represent the improvement of our technique compared to DasTech [1] and KimTech [2], respectively) without enabling the **On-line Population Module**

the phase diversity is low (*e.g.* $D_L = 30$), we can achieve 1.29x and 1.42x improvement in performance compared with DasTech and KimTech, respectively, when the **On-line Population Module** is enabled. Even in the case when $D_L = 350$, our technique can still achieve 8% and 19% improvement in performance compared to DasTech and KimTech, respectively. Note that, no matter what D_L is, the Q-table size of our technique is the same as DasTech and KimTech. In practice, during a short period of time, the number of different phases in programs is not very large. Therefore our proposed technique can provide more efficient management than the existing techniques such as KimTech and DasTech.

5.8 Conclusion

In this chapter, we have studied the thermal and power coupling effect in 3D CPUs (stacking memory-on-logic) and discovered the activity of processor will affect the performance of memory and the reliability of TSVs through power and temperature coupling. Afterwards, we build an overall model to capture this influence. Then we propose a phase-driven Q-learning based DRM technique which can tune the activity of the processor to maximize the 3D CPU performance subject to the reliability constraint. Our proposed dynamic management method combines the reinforcement learning method with on-line phase detection techniques (*e.g.* [5]) and can provide efficient management compared to existing learning based techniques. We evaluate our proposed technique by solving a DRM problem on a 3D CPU (Section 5.2). According to the result, when the number of phases is smaller than the number of working modes, our proposed technique can achieve over 1.36 improvement in performance with 60% memory savings.

Chapter 6: Conclusion and Future Work

The 3D integration technology is promising to keep the Moore's Law by reducing the interconnect delay and power as well as increasing the logic density of the chip. However, the benefits of this new technology come with challenges. Temperature and power integrity are two major problems due to the stacking structure of 3D ICs. Moreover, the TSV (a unique structure in the 3D IC which enables interlayer communication) introduces new failure modes thus making the reliability issue in 3D ICs more complicated. In this thesis, we have proposed to address the 3D IC challenges both in design stage and in run-time.

For the design stage solution, we first adopt the micro-fluidic cooling to enhance the heat removal capability in 3D ICs. Then, we discover the MF cooling structure brings several new challenges to the physical design of 3D ICs. Therefore, we propose co-optimization techniques for the physical design of 3D ASICs and FPGAs. For the runtime management, we demonstrate the main challenge is the strong power/thermal coupling across layers in the 3D IC. Afterwards, we propose a phase-driven learning based dynamic management technique to maximize the performance of the 3D CPU subject to some reliability constraints.

In Chapter 3, we propose a hierarchical quadri-partitioning based placement framework to handle the trade-offs during the physical design stage of 3D ASICs with MF cooling. Our framework is first used to perform cooling aware placement of gates and signal TSVs with the existence of micropin-fin based MF cooling (1^{st}-level Co-design Problem) and it can achieve significant reduction of the in-layer temperature gradient compared to the traditional placement method. We also extend the framework to co-design the gate-level floor plan, the assignment of micro-channels and the allocation of P/G TSVs for 3D ASICs with micro-channel based MF cooling. Compared to the traditional sequential physical design flow (*i.e.* placing gates and signal TSVs, allocation micro-channels and designing the 3D PDN are performed in the successive order), our proposed technique is able to guarantee the feasibility of the design (*i.e.* no conflicting between TSVs and micro-channels, no violation of temperature *etc.*) with a little increase in wire-length.

In Chapter 4, we have studied the influence of MF cooling to the 3D FPGA and investigated design methodologies for 3D FPGAs with micro-fluidic cooling. Compared with 3D ASICs, the thermo-electrical trade-offs in 3D FPGAs with MF cooling is more complex. In order to handle these trade-offs and determine the proper designs of FPGAs as well as the MF cooling heat sinks, we propose a design space exploration framework. Compared with naive design methods (*e.g.* designs to include the maximum number of micro-channels *etc.*), our DSE framework can find more optimal designs with significant improvement in performance and energy efficiency. Moreover, according to the experimental results, we can also provide guidelines for designing 3D FPGAs with micro-channel based MF cooling. We also extend the proposed DSE framework to place and route 3D FPGAs with micro-channel based MF cooling such that the in-layer temperature gradient is reduced.

In Chapter 5, we have studied the thermal and power coupling effect in 3D CPUs (stacking memory-on-logic) and discovered the activity of processor will affect the performance of memory and the reliability of TSVs through power and temperature coupling. Afterwards, we build an overall model to capture this influence. Then we propose a phase-driven Q-learning based DRM technique which can tune the activity of the processor to maximize the 3D CPU performance subject to the reliability constraint. Our proposed dynamic management method combines the reinforcement learning method with on-line phase detection techniques (*e.g.* [5]) and can provide efficient management compared to existing learning based techniques. We evaluate our proposed technique by solving a DRM problem on a 3D CPU (Section 5.2). According to the result, when the number of phases is smaller than the number of working modes, our proposed technique can achieve over 1.36 improvement in performance with 60% memory savings.

6.1 Future Work

6.1.1 Signal Integrity

In Chapter 3 and 5, we have considered the power integrity in 3D IC and investigated the design-time and runtime methods to maintain the power integrity and reliability induced by power integrity issues. However, besides power integrity, we also need to ensure the signal voltage noise is maintained within a certain margins.

Due to its high transition frequency, the signal voltage can be coupled between

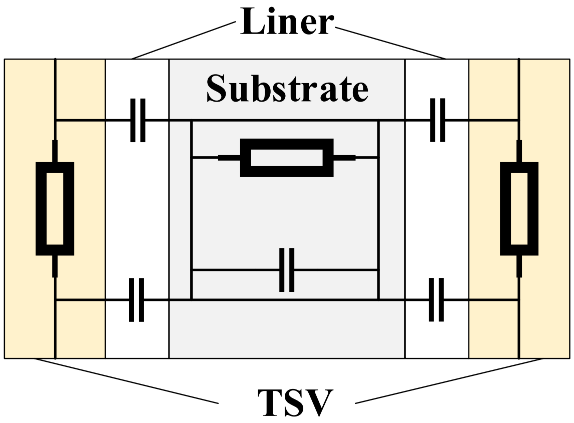


Figure 6.1: TSV-TSV coupling circuit model [8]

switched devices. All the coupling effects will cause leakage/short circuit currents which possibly result in digital glitches that affect circuit behavior or cause incorrect computations. In 3D ICs, the TSV structure provides another source for signal coupling and compared to planar wires, TSVs can be more problematic due to the fact that they are much larger and surrounded by a much thinner dielectric layer [105, 106].

Figure 6.1 shows a circuit model of coupling between two TSVs. There are various paths for signals to leak through TSVs. Through the thin insulation layer enclosed the metal vias, the signals can be easily coupled into the silicon substrate, from where the voltage noise can be coupled into other TSVs or transistors on other layers (since the silicon substrate is conductive). To make things worse, the distance between two signal TSVs has little impact on such leakage since the liner capacitance is independent of the distance between TSVs. In addition, due to the small dimension of signal TSVs, it is not practical to use TSV shielding techniques (as proposed in [8]) to mitigate the voltage noise. Another challenge is the high temperature and thermal gradient in 3D ICs can lead to significant inaccuracy in delay and clock skew thus affecting the signal integrity [107]. This make the signal integrity mitigation in 3D ICs even more complex.

However, there are still potentials for maintaining the signal integrity. If we use fine-grained design and integration technologies (rather than global placement methods investigated in this thesis), we can tune the dimension and distance between signal TSVs (and P/G TSVs) to achieve the reduction in signal coupling through TSVs.

6.1.2 Deep Learning Based EDA Techniques

Nowadays, deep learning is currently pervasively applied to fields such as computer vision, natural language processing, bioinformatics *etc.* [108]. This revolutionary technology has been changing people's life and our society. Deep learning methods have strong power in solving complex problems involving large amounts of data which are hard to be solved using traditional methods. Applying deep learning techniques to the electronic design automation (EDA) [109, 110] is a quite new yet promising idea, due to the following reasons:

- The number of devices in a chip keeps increasing either through technology scaling or 3D integration. This significantly expands the solution space thus making it very time consuming if using traditional methods to perform placement, routing *etc.* [109]. On the other hand, deep learning is promising to provide an efficient solution to such large-sized problems.

- At the advanced technology nodes, the design of integrated circuits should consider factors from multiphysical domains: temperature, IR drop, reliability, crosstalk *etc.* . This thesis is driven by this factor. However, with the increase of the dimension of multiphysics, traditional modeling methods may lead to suboptimal and inefficient solutions [110]. However, deep learning is good at extracting features from huge amounts of data and building models for complex relationships.

- The IC design involves a number of different stages (*e.g.* logic synthesis, placement, routing *etc.*). Traditionally, different stages are optimized separately. When the circuit gets more complex, this will cause a problem that the optimization at an early stage (*e.g.* logic synthesis) may not improve or even degrade the final results (*e.g.* the performance of the circuits). However, it is hard to predict the quality of results (QoR) at that early stage using current methods. On the other hand, deep learning is believed to be promising to solve this problem by efficiently modeling the QoR [110].

Compared to applications like computer vision, self-driving *etc.* , the application of deep learning in EDA is just proposed and there are many possibilities. Therefore, we think it is an interesting topic to investigate in future.

6.1.3 Value-Driven Neural Network ASICs

Nowadays, neural network chips or AI chips have attracted substantial interest both in industry and academia. Neural network is highly computation intensive

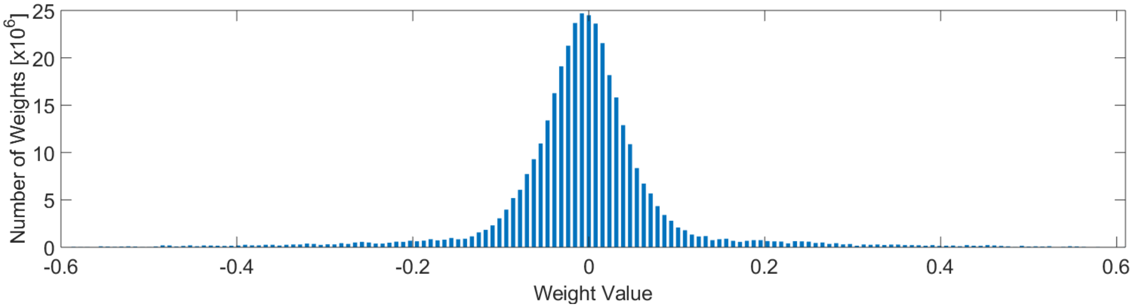
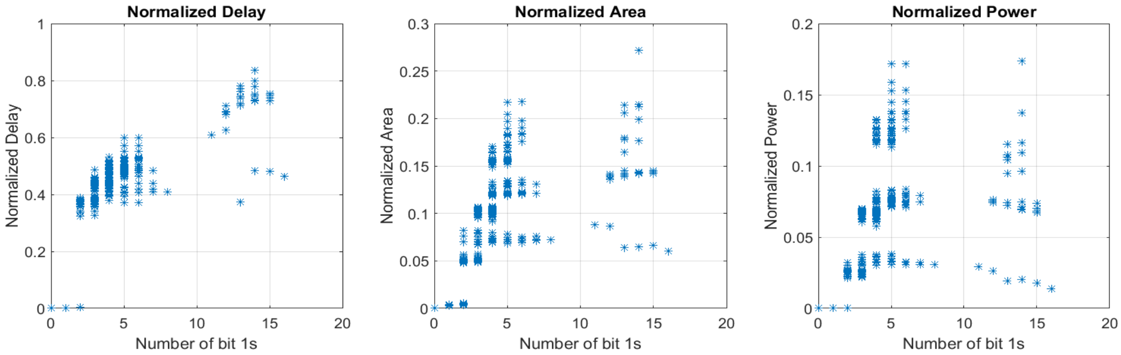


Figure 6.2: The distribution of weights for GoogleNet [9]

and memory intensive, thus making it inefficient to run on traditional CPUs or even GPUs. Therefore, people tend to AI chips for processing neural network in a faster and more efficient way [111,112]. Another benefits of AI chips is that the size of an AI chip can be small enough to fit in wearable devices thus enabling local neural network computation which brings benefits to IoT systems.

Due to the fast evolution of the neural network, current AI chips are designed as general as possible. However, as the neural network algorithms become more mature, in the near future, there will be another direction of AI chips that are built specific to a certain application (just like today's ASIC). If this is true, we can utilize the information of the parameters of the neural network to design the AI chip. This is expected to have better performance and energy efficiency.

One possible design method is to fix one input of the multiplier to a constant value (according to the synaptic weight of the neural network), and use this kind of value-driven optimized multiplier to replace the general multiplier. This idea is motivated by the fact that if we use fixed-point data to represent the weight neural network parameters, these parameters will only take a few distinct values



(a) (b) (c)

Figure 6.3: The (a) delay, (b) area, and (c) power for 16-bit fixed-point multipliers with different fixed input value (evaluated by the number of 1's) normalized to the data of the general multiplier

(as illustrated by Figure 6.2 for GoogleNet [9]). Therefore, we can optimize the multiplier according to these values (by fixing one input of the multiplier to a weight value). We have performed SPICE simulation for some of this kind of multipliers using *Cadence Spectre* and compared their energy and delay with general multipliers (as illustrated by Figure 6.3). As illustrated by the figure, significant improvement can be acquired in performance, area and power.

Despite the advantages of the value-driven method, there are several challenges. For example, how to design the controller and memory system to mitigate the possible overhead brought by the one-to-one mapping between the weight and the multiplier. In addition, since the number of weights is large while the area of an AI chip is limited, it requires research efforts on how to determine the number/size of multipliers as well as other modules (*e.g.* adders, buffers *etc.*) to balance the performance, energy efficiency and resource utilization.

Bibliography

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

[2] Taeyoung Kim, Xin Huang, Hai-Bao Chen, Valeriy Sukharev, and Sheldon X-D Tan. Learning-based dynamic reliability management for dark silicon processor considering em effects. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 463–468. IEEE, 2016.

[3] Zhimin Wan, Yoon Jo Kim, and Yogendra K Joshi. Compact modeling of 3d stacked die inter-tier microfluidic cooling under non-uniform heat flux. In *ASME 2012 International Mechanical Engineering Congress and Exposition*, pages 911–917. American Society of Mechanical Engineers, 2012.

[4] Caleb Serafy, Ankur Srivastava, and Donald Yeung. Continued frequency scaling in 3d ics through micro-fluidic cooling. In *Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2014 IEEE Intersociety Conference on*, pages 79–85. IEEE, 2014.

[5] Timothy Sherwood, Erez Perelman, Greg Hamerly, Suleyman Sair, and Brad Calder. Discovering and exploiting program phases. *IEEE micro*, 23(6):84–93, 2003.

[6] Taeyoung Kim, Xin Huang, Hai-Bao Chen, Valeriy Sukharev, and Sheldon X-D Tan. Learning-based dynamic reliability management for dark silicon processor considering em effects. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*, pages 463–468. IEEE, 2016.

[7] Anup Das, Geoff V Merrett, Mirco Tribastone, and Bashir M Al-Hashimi. Workload change point detection for runtime thermal management of embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(8):1358–1371, 2016.

[8] Caleb Serafy and Ankur Srivastava. Tsv replacement and shield insertion for tsv–tsv coupling reduction in 3-d global placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(4):554–562, 2015.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabi-

novich, et al. Going deeper with convolutions. Cvpr, 2015.

[10] Wm A Wulf and Sally A McKee. Hitting the memory wall: implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995.

[11] Caleb Serafy, Bing Shi, Ankur Srivastava, and Donald Yeung. High performance 3d stacked dram processor architectures with micro-fluidic cooling. In *3D Systems Integration Conference (3DIC), 2013 IEEE International*, pages 1–8. IEEE, 2013.

[12] Jie Meng, Katsutoshi Kawakami, and Ayse K Coskun. Optimizing energy efficiency of 3-d multicore systems with stacked dram under power and thermal constraints. In *Proceedings of the 49th Annual Design Automation Conference*, pages 648–655. ACM, 2012.

[13] Xin Zhao, Yang Wan, Michael Scheuermann, and Sung Kyu Lim. Transient modeling of tsv-wire electromigration and lifetime analysis of power distribution network for 3d ics. In *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, pages 363–370. IEEE, 2013.

[14] Jun So Pak, Joohee Kim, Jonghyun Cho, Kiyeong Kim, Taigon Song, Seungyoung Ahn, Junho Lee, Hyungdong Lee, Kunwoo Park, and Joungho Kim. Pdn impedance modeling and analysis of 3d tsv ic by using proposed p/g tsv array model based on separated p/g tsv and chip-pdn models. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 1(2):208–219, 2011.

[15] Runjie Zhang, Kaushik Mazumdar, Brett H Meyer, Ke Wang, Kevin Skadron, and Mircea Stan. A cross-layer design exploration of charge-recycled power-delivery in many-layer 3d-ic. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2015.

[16] Tiantao Lu, Zhiyuan Yang, and Ankur Srivastava. Electromigration-aware placement for 3d-ics. In *Quality Electronic Design (ISQED), 2016 17th International Symposium on*, pages 35–40. IEEE, 2016.

[17] Huy N Phan and Dereje Agonafer. Experimental analysis model of an active cooling method for 3d-ics utilizing multidimensional configured thermoelectric coolers. *Journal of Electronic Packaging*, 132(2):024501, 2010.

[18] Rujia Wang, Lei Jiang, Youtao Zhang, and Jun Yang. Sd-pcm: Constructing reliable super dense phase change memory under write disturbance. *ACM SIGARCH Computer Architecture News*, 43(1):19–31, 2015.

[19] Ravi Kandasamy, Xiang-Qi Wang, and Arun S Mujumdar. Transient cooling of electronics using phase change material (pcm)-based heat sinks. *Applied Thermal Engineering*, 28(8-9):1047–1057, 2008.

[20] Bing Dang, Muhannad S Bakir, Deepak Chandra Sekar, Calvin R King Jr, and James D Meindl. Integrated microfluidic cooling and interconnects for 2d and 3d chips. *IEEE Transactions on Advanced Packaging*, 33(1):79–87, 2010.

[21] Muhannad S Bakir, Calvin King, Deepak Sekar, Hiren Thacker, Bing Dang, Gang Huang, Azad Naeemi, and James D Meindl. 3d heterogeneous integrated

systems: Liquid cooling, power delivery, and implementation. In *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE*, pages 663–670. IEEE, 2008.

[22] Avram Bar-Cohen, Ankur Srivastava, and Bing Shi. Thermal-electrical co-design of three-dimensional integrated circuits: challenges and opportunities. *Computational Thermal Sciences: An International Journal*, 5(6):441–458, 2013.

[23] David B Tuckerman and RFW Pease. High-performance heat sinking for vlsi. *IEEE Electron device letters*, 2(5):126–129, 1981.

[24] Radu Teodorescu and Josep Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 363–374. IEEE Computer Society, 2008.

[25] Shuguang Feng, Shantanu Gupta, Amin Ansari, and Scott Mahlke. Maestro: Orchestrating lifetime reliability in chip multiprocessors. In *International Conference on High-Performance Embedded Architectures and Compilers*, pages 186–200. Springer, 2010.

[26] William Song, Saibal Mukhopadhyay, and Sudhakar Yalamanchili. Architectural reliability: Lifetime reliability characterization and management ofmany-core processors. *IEEE Computer Architecture Letters*, 14(2):103–106, 2015.

[27] Ayse Kivilcim Coskun, Tajana Simunic Rosing, and Kenny C Gross. Utilizing predictors for efficient thermal management in multiprocessor socs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1503–1516, 2009.

[28] Bing Shi, Ankur Srivastava, and Peng Wang. Non-uniform micro-channel design for stacked 3d-ics. In *Proceedings of the 48th Design Automation Conference*, pages 658–663. ACM, 2011.

[29] Yue Zhang, Calvin R King, Jesal Zaveri, Yoon Jo Kim, Vivek Sahu, Yogenda Joshi, and Muhannad S Bakir. Coupled electrical and thermal 3d ic centric microfluidic heat sink design and technology. In *Electronic Components and Technology Conference (ECTC), 2011 IEEE 61st*, pages 2037–2044. IEEE, 2011.

[30] Mohamed M Sabry, Arvind Sridhar, Jie Meng, Ayse K Coskun, and David Atienza. Greencool: An energy-efficient liquid cooling design technique for 3-d mpsocs via channel width modulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(4):524–537, 2013.

[31] Bing Shi, Caleb Serafy, and Ankur Srivastava. Co-optimization of tsv assignment and micro-channel placement for 3d-ics. In *Proceedings of the 23rd ACM international conference on Great lakes symposium on VLSI*, pages 337–338. ACM, 2013.

[32] Zhiyuan Yang and Ankur Srivastava. Physical co-design for micro-fluidically cooled 3d ics. In *Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2016 15th IEEE Intersociety Conference on*, pages 1373–1380. IEEE, 2016.

[33] Arvind Sridhar, Alessandro Vincenzi, Martino Ruggiero, Thomas Brunschwiler, and David Atienza. 3d-ice: Fast compact transient thermal modeling for 3d ics with inter-tier liquid cooling. In *Proceedings of the International Conference on Computer-Aided Design*, pages 463–470. IEEE Press, 2010.

[34] Nauman H Khan, Syed M Alam, and Soha Hassoun. Power delivery design for 3-d ics using different through-silicon via (tsv) technologies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(4):647–658, 2011.

[35] Michael B Healy and Sung Kyu Lim. Power delivery system architecture for many-tier 3d systems. In *Electronic Components and Technology Conference (ECTC), 2010 Proceedings 60th*, pages 1682–1688. IEEE, 2010.

[36] Jie Gu and Chris H Kim. Multi-story power delivery for supply noise reduction and low voltage operation. In *Proceedings of the 2005 international symposium on Low power electronics and design*, pages 192–197. ACM, 2005.

[37] Paul Falkenstern, Yuan Xie, Yao-Wen Chang, and Yu Wang. Three-dimensional integrated circuits (3d ic) floorplan and power/ground network co-synthesis. In *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, pages 169–174. IEEE, 2010.

[38] Meeta S Gupta, Jarod L Oatley, Russ Joseph, Gu-Yeon Wei, and David M Brooks. Understanding voltage variations in chip multiprocessors using a distributed power-delivery network. In *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*, pages 1–6. IEEE, 2007.

[39] Thomas Frank, Stephane Moreau, Cedrick Chappaz, Lucile Arnaud, Patrick Leduc, Aurelie Thuaire, and Lorena Anghel. Electromigration behavior of 3d-ic tsv interconnects. In *Electronic Components and Technology Conference (ECTC), 2012 IEEE 62nd*, pages 326–330. IEEE, 2012.

[40] Jiwoo Pak, Sung Kyu Lim, and David Z Pan. Electromigration study for multiscale power/ground vias in tsv-based 3-d ics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(12):1873–1885, 2014.

[41] Steven L Wright, Paul S Andry, Edmund Sprogis, Bing Dang, and Robert J Polastre. Reliability testing of through-silicon vias for high-current 3d applications. In *Electronic Components and Technology Conference, 2008. ECTC 2008. 58th*, pages 879–883. IEEE, 2008.

[42] Tiantao Lu, Zhiyuan Yang, and Ankur Srivastava. Post-placement optimization for thermal-induced mechanical stress reduction. In *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*, pages 158–163. IEEE, 2016.

[43] Moongon Jung, Joydeep Mitra, David Z Pan, and Sung Kyu Lim. Tsv stress-aware full-chip mechanical reliability analysis and optimization for 3d ic. *Com-*

munications of the ACM, 57(1):107–115, 2014.

[44] John H Lau. Tsv manufacturing yield and hidden costs for 3d ic integration. In *Electronic Components and Technology Conference (ECTC), 2010 Proceedings 60th*, pages 1031–1042. IEEE, 2010.

[45] Peter Spindler, Ulf Schlichtmann, and Frank M Johannes. Kraftwerk2a fast force-directed quadratic placement approach using an accurate net model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(8):1398–1411, 2008.

[46] Jason Cong, Guojie Luo, and Yiyu Shi. Thermal-aware cell and through-silicon-via co-placement for 3d ics. In *Proceedings of the 48th Design Automation Conference*, pages 670–675. ACM, 2011.

[47] Jason Cong, Guojie Luo, Jie Wei, and Yan Zhang. Thermal-aware 3d ic placement via transformation. In *Design Automation Conference, 2007. ASP-DAC'07. Asia and South Pacific*, pages 780–785. IEEE, 2007.

[48] Brent Goplen and Sachin Spatnekar. Placement of 3d ics with thermal and interlayer via considerations. In *Proceedings of the 44th annual Design Automation Conference*, pages 626–631. ACM, 2007.

[49] Guojie Luo, Yiyu Shi, and Jason Cong. An analytical placement framework for 3-d ics and its extension on thermal awareness. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(4):510–523, 2013.

[50] Brent Goplen and Sachin Sapatnekar. Efficient thermal placement of standard cells in 3d ics using a force directed approach. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 86. IEEE Computer Society, 2003.

[51] Min Pan, Natarajan Viswanathan, and Chris Chu. An efficient and effective detailed placement algorithm. In *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, pages 48–55. IEEE Computer Society, 2005.

[52] George Karypis and Vipin Kumar. hmetis 1.5: A hypergraph partitioning package. Technical report, Technical report, Department of Computer Science, University of Minnesota, 1998. Available on the WWW at URL http://www. cs. umn. edu/metis, 1998.

[53] Dae Hyun Kim, Krit Athikulwongse, and Sung Kyu Lim. Study of through-silicon-via impact on the 3-d stacked ic layout. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(5):862–874, 2013.

[54] Dae Hyun Kim, Saibal Mukhopadhyay, and Sung Kyu Lim. Tsv-aware interconnect length and power prediction for 3d stacked ics. In *Interconnect Technology Conference, 2009. IITC 2009. IEEE International*, pages 26–28. IEEE, 2009.

[55] Dennis J-H Huang and Andrew B Kahng. Partitioning-based standard-cell global placement with an exact objective. In *Proceedings of the 1997 interna-*

tional symposium on Physical design, pages 18–25. ACM, 1997.

[56] Christoph Albrecht. Iwls 2005 benchmarks. In *International Workshop for Logic Synthesis (IWLS): http://www. iwls. org*, 2005.

[57] Jason Cong and Guojie Luo. A 3d physical design flow based on open access. In *Communications, Circuits and Systems, 2009. ICCCAS 2009. International Conference on*, pages 1103–1107. IEEE, 2009.

[58] Amos R Omondi and Jagath Chandana Rajapakse. *FPGA implementations of neural networks*, volume 365. Springer, 2006.

[59] Seul Jung and Sung su Kim. Hardware implementation of a real-time neural network controller with a dsp and an fpga for nonlinear systems. *IEEE Transactions on Industrial Electronics*, 54(1):265–271, 2007.

[60] Mingjie Lin, Abbas El Gamal, Yi-Chang Lu, and Simon Wong. Performance benefits of monolithically stacked 3-d fpga. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 26(2):216–229, 2007.

[61] Ian Kuon, Russell Tessier, and Jonathan Rose. Fpga architecture: Survey and challenges. *Foundations and trends in electronic design automation*, 2(2):135–253, 2008.

[62] Vaughn Betz, Jonathan Rose, and Alexander Marquardt. *Architecture and CAD for deep-submicron FPGAs*, volume 497. Springer Science & Business Media, 2012.

[63] Cristinel Ababei, Pongstorn Maidee, and Kia Bazargan. Exploring potential benefits of 3d fpga integration. In *International Conference on Field Programmable Logic and Applications*, pages 874–880. Springer, 2004.

[64] Kostas Siozios, Alexandros Bartzas, and Dimitrios Soudris. Architecture-level exploration of alternative interconnection schemes targeting 3d fpgas: A software-supported methodology. *International Journal of Reconfigurable Computing*, 2008, 2008.

[65] Jason Helge Anderson and Farid N Najm. Active leakage power optimization for fpgas. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(3):423–437, 2006.

[66] Tim Tuan, Sean Kao, Arif Rahman, Satyaki Das, and Steve Trimberger. A 90nm low-power fpga for battery-powered applications. In *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, pages 3–11. ACM, 2006.

[67] Aman Gayasen, Vijaykrishnan Narayanan, Mahmut Kandemir, and Arifur Rahman. Designing a 3-d fpga: switch box architecture and thermal issues. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(7):882–893, 2008.

[68] Ioannis Savidis, Syed M Alam, Ankur Jain, Scott Pozder, Robert E Jones, and Ritwik Chatterjee. Electrical modeling and characterization of through-silicon

vias (tsvs) for 3-d integrated circuits. *Microelectronics Journal*, 41(1):9–16, 2010.

[69] Julien Lamoureux and Steven JE Wilton. Activity estimation for field-programmable gate arrays. In *Field Programmable Logic and Applications, 2006. FPL'06. International Conference on*, pages 1–8. IEEE, 2006.

[70] Li Shang, Alireza S Kaviani, and Kusuma Bathala. Dynamic power consumption in virtex-ii fpga family. In *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, pages 157–164. ACM, 2002.

[71] Peter cheung. 2008. modern fpga architecutres. (jun 2008). retrieved august 30, 2015. from http://www.ee.ic.ac.uk/.

[72] Gian Luca Loi, Banit Agrawal, Navin Srivastava, Sheng-Chih Lin, Timothy Sherwood, and Kaustav Banerjee. A thermally-aware performance analysis of vertically integrated (3-d) processor-memory hierarchy. In *Proceedings of the 43rd annual Design Automation Conference*, pages 991–996. ACM, 2006.

[73] Joe Jeddeloh and Brent Keeth. Hybrid memory cube new dram architecture increases density and performance. In *VLSI Technology (VLSIT), 2012 Symposium on*, pages 87–88. IEEE, 2012.

[74] John D Leidel and Yong Chen. Hmc-sim: A simulation framework for hybrid memory cube devices. *Parallel Processing Letters*, 24(04):1442002, 2014.

[75] Dong-Ik Jeon and Ki-Seok Chung. Cashmc: A cycle-accurate simulator for hybrid memory cube. *IEEE Computer Architecture Letters*, 16(1):10–13, 2017.

[76] 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 multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 469–480. IEEE, 2009.

[77] Donghyuk Lee, Yoongu Kim, Gennady Pekhimenko, Samira Khan, Vivek Seshadri, Kevin Chang, and Onur Mutlu. Adaptive-latency dram: Optimizing dram timing for the common-case. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, pages 489–501. IEEE, 2015.

[78] Timothy J Dell. A white paper on the benefits of chipkill-correct ecc for pc server main memory. *IBM Microelectronics Division*, 11, 1997.

[79] Chin-Long Chen and MY Hsiao. Error-correcting codes for semiconductor memory applications: A state-of-the-art review. *IBM Journal of Research and Development*, 28(2):124–134, 1984.

[80] Robert Baumann. Soft errors in advanced computer systems. *IEEE Design & Test of Computers*, 22(3):258–266, 2005.

[81] Tiantao Lu, Caleb Serafy, Zhiyuan Yang, and Ankur Srivastava. Voltage noise induced dram soft error reduction technique for 3d-cpus. In *Proceedings of the*

2016 International Symposium on Low Power Electronics and Design, pages 82–87. ACM, 2016.

[82] Pulkit Jain, Tae-Hyoung Kim, John Keane, and Chris H Kim. A multi-story power delivery technique for 3d integrated circuits. In *Proceedings of the 2008 international symposium on Low Power Electronics & Design*, pages 57–62. ACM, 2008.

[83] T Frank, C Chappaz, P Leduc, L Arnaud, F Lorut, S Moreau, A Thuaire, R El Farhane, and L Anghel. Resistance increase due to electromigration induced depletion under tsv. In *Reliability Physics Symposium (IRPS), 2011 IEEE International*, pages 3F–4. IEEE, 2011.

[84] RG Filippi, P-C Wang, A Brendler, K Chanda, and JR Lloyd. Implications of a threshold failure time and void nucleation on electromigration of copper interconnects. *Journal of Applied Physics*, 107(10):103709, 2010.

[85] James R Black. Electromigrationa brief survey and some recent results. *IEEE Transactions on Electron Devices*, 16(4):338–347, 1969.

[86] Jayanth Srinivasan, Sarita V Adve, Pradip Bose, and Jude A Rivers. The impact of technology scaling on lifetime reliability. In *Dependable Systems and Networks, 2004 International Conference on*, pages 177–186. IEEE, 2004.

[87] William Lloyd Bircher and Lizy K John. Complete system power estimation using processor performance events. *IEEE Transactions on Computers*, 61(4):563–577, 2012.

[88] Jun Yang, Xiuyi Zhou, Marek Chrobak, Youtao Zhang, and Lingling Jin. Dynamic thermal management through task scheduling. In *Performance Analysis of Systems and software, 2008. ISPASS 2008. IEEE International Symposium on*, pages 191–201. IEEE, 2008.

[89] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. Hotspot: A compact thermal modeling methodology for early-stage vlsi design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, 2006.

[90] Yufu Zhang, Bing Shi, and Ankur Srivastava. A statistical framework for designing on-chip thermal sensing infrastructure in nano-scale systems. In *Proceedings of the 19th international symposium on Physical design*, pages 169–176. ACM, 2010.

[91] MN Chang, Y-H Lee, SY Lee, K Joshi, CC Ko, CC Chiu, and K Wu. A new insight into beol tddb lifetime model for advanced technology scaling. In *2015 IEEE International Electron Devices Meeting (IEDM)*, pages 7–4. IEEE, 2015.

[92] Jayanth Srinivasan, Sarita V Adve, Pradip Bose, and Jude A Rivers. Lifetime reliability: Toward an architectural solution. *IEEE Micro*, 25(3):70–80, 2005.

[93] John Keane, Xiaofei Wang, Devin Persaud, and Chris H Kim. An all-in-one silicon odometer for separately monitoring hci, bti, and tddb. *IEEE Journal of Solid-State Circuits*, 45(4):817–829, 2010.

[94] Prashant Singh, Eric Karl, Dennis Sylvester, and David Blaauw. Dynamic nbti management using a 45 nm multi-degradation sensor. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(9):2026–2037, 2011.

[95] Ying Tan, Wei Liu, and Qinru Qiu. Adaptive power management using reinforcement learning. In *Computer-Aided Design-Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 461–467. IEEE, 2009.

[96] Shruti Padmanabha, Andrew Lukefahr, Reetuparna Das, and Scott Mahlke. Trace based phase prediction for tightly-coupled heterogeneous cores. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 445–456. ACM, 2013.

[97] Canturk Isci, Gilberto Contreras, and Margaret Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 359–370. IEEE Computer Society, 2006.

[98] Lin Huang, Feng Yuan, and Qiang Xu. Lifetime reliability-aware task allocation and scheduling for mpsoc platforms. In *Proceedings of the DATE*, pages 51–56. EDAA, 2009.

[99] Zhiyuan Yang, Caleb Serafy, Tiantao Lu, and Ankur Srivastava. Phase-driven learning-based dynamic reliability management for multi-core processors. In

Proceedings of the 54th Annual Design Automation Conference 2017, page 46. ACM, 2017.

[100] Inchoon Yeo, Chih Chun Liu, and Eun Jung Kim. Predictive dynamic thermal management for multicore systems. In *Proceedings of the 45th annual Design Automation Conference*, pages 734–739. ACM, 2008.

[101] Heba Khdr, Thomas Ebi, Muhammad Shafique, Hussam Amrouch, and Jörg Henkel. mdtm: Multi-objective dynamic thermal management for on-chip systems. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 330. European Design and Automation Association, 2014.

[102] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

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

[104] Rafael Ubal, Byunghyun Jang, Perhaad Mistry, Dana Schaa, and David Kaeli. Multi2sim: a simulation framework for cpu-gpu computing. In *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, pages 335–344. ACM, 2012.

[105] Caleb Serafy, Bing Shi, and Ankur Srivastava. A geometric approach to chip-scale tsv shield placement for the reduction of tsv coupling in 3d-ics. *Integration, the VLSI Journal*, 47(3):307–317, 2014.

[106] Chang Liu, Taigon Song, Jonghyun Cho, Joohee Kim, Joungho Kim, and Sung Kyu Lim. Full-chip tsv-to-tsv coupling analysis and optimization in 3d ic. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 783–788. IEEE, 2011.

[107] Amir H Ajami, Kaustav Banerjee, and Massoud Pedram. Non-uniform chip-temperature dependent signal integrity. In *VLSI Technology, 2001. Digest of Technical Papers. 2001 Symposium on*, pages 145–146. IEEE, 2001.

[108] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[109] Andrew B Kahng. Machine learning applications in physical design: Recent results and directions. 2018.

[110] Andrew B Kahng. New directions for learning-based ic design tools and methodologies. In *Design Automation Conference (ASP-DAC), 2018 23rd Asia and South Pacific*, pages 405–410. IEEE, 2018.

[111] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM Sigplan Notices*, 49(4):269–284, 2014.

[112] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 243–254. IEEE, 2016.