國 立 交 通 大 學

電信工程研究所

碩 士 論 文

對三維度積體電路實體設計流程具可整合性和具網格分析適應性的溫度模擬器
3D-Adaptive-ADI: A Grid-Adaptive and Integrable Thermal Simulator for 3D IC Physical Design Flow Based on ADI Method

研究生　：李亭蓉

指導教授：李育民　教授

中 華 民 國 一〇〇 年 八 月

對三維度積體電路實體設計流程具可整合性和具網格分析適應性的溫度模擬器

3D-Adaptive-ADI: A Grid-Adaptive and Integrable Thermal Simulator for 3D IC Physical Design Flow Based on ADI Method

研 究 生：李亭蓉	Student：Ting-Jung Li
指導教授：李育民	Advisor：Yu-Min Lee

國 立 交 通 大 學
電信工程研究所
碩 士 論 文

A Thesis
Submitted to Institute of Computer and Information Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer and Information Science

August 2011

Hsinchu, Taiwan, Republic of China

中華民國一〇〇年八月

3D-Adaptive-ADI:
對三維度積體電路實體設計流程具可整合性和具網格分析適應性的溫度模擬器

學生：李亭蓉　　　　　　　　　　　　　　　　指導教授:李育民 博士

國立交通大學電機工程學系碩士班

摘　　要

經由提高晶片之間互連導線密度達到成本效益的三維度積體電路，被視為一個解決二維度平面積體電路的良好替代方案。三維度積體電路確實提供快速成長的電路系統許多益處，然而，由於主動電路層垂直堆疊造成較高的功率密度和較低的導熱特性，散熱問題成為其所面臨到的棘手挑戰之一。因此，相較於過往對於溫度只做封裝前的電路驗證是不夠的，對於未來高整合系統而言，溫度議題的處理需要被提早到實體設計階段做考量。基於這些原因，吾人使用提出的 3D-AADI 演算法於從平面規劃 (floor-plan) 階段到驗證 (verification) 階段，研發了一個具有網格分析適應性的溫度模擬器。

本篇論文提出的三維度積體電路溫度模擬器，不僅在效率和精準度之間取得平衡的低複雜度演算法，更可以建造出具適應性分析網格以避免因非關鍵區域造成的不必要時間浪費。此分析器包含四個程序分別是：熱趨勢的估計、初始適應性網格的建立、非規則性矩陣的建立、和3D-AADI非規則性矩陣的運算。由實驗結果顯示，此分析器在0.01%的收斂率下達到數百倍的速度提升，而因線性複雜度，隨著分析解析度越密加速更為顯著。此外，此分析器具有可以部分更新 (incremental) 的特性對應用於考慮溫度設計流程 (temperature-aware design flow) 中，是十分有效率的應用。

3D-Adaptive-ADI:
A Grid-Adaptive and Integrable Thermal Simulator for 3D IC Physical Design Flow Based on ADI Method

Student: Ting-Jung Li Advisor: Dr. Yu-Min Lee

Department of Electrical Engineering
National Chiao Tung University

ABSTRACT

3D ICs, which deal with cost-effective achievement by increasing the densities of interconnection between dies, are regarded as an attractive alternative solution for overcoming the bottlenecks on 2D planar ICs. 3D ICs offer the increased system a large number of advantages; however, one of critical challenges is heat dissipation due to higher accumulated power density and lower thermal conductivity of inter-layer dielectrics for vertical stacking layers of active tier. In this way, the management of thermal issues should be considered during physical design stages rather than only pre-packaging verification on the future highly integrated systems. For these reasons, we develop an adaptive thermal simulator applying our 3D-AADI algorithm for providing temperature distribution to 3DIC physical design flow from floor-plan level to verification level.

The proposed 3DIC thermal simulator, 3D-AADI, both utilizes low-complexity algorithm for achieving both efficient runtime and accuracy and constructs adaptive simulating granularity for avoiding unwanted runtime and resource consumption in non-critical position. Furthermore, 3D-AADI, including estimating heat trend, constructing initial adaptive grids, establishing non-uniform structure, and calculating 3D-AADI matrix processes, improves two orders of magnitude under 0.01% convergence in experimental results. As a result of linear complexity, the finer simulating granularity leads to the more speeding up. Due to the partial updating characteristics, 3D-AADI can not only be regarded as a reliable thermal simulator but also be applied to 3DIC design flow as a thermal-driven kernel.

誌　　謝

本篇論文得以順利出爐，首先感謝我的指導教授李育民博士，老師在研究上給予的指導，是我在碩士學習過程中最大的收穫。因為在老師的指導下，我學到一個研究生該有的面對問題挑戰的態度和解決的思慮，這是我學習生涯中得到最大的寶藏。

在碩士的歲月中，感謝在我鑽研問題階段，時常找我討論概念、程式技巧給予指導的博士班培育學長，您對研究的執著和熱誠，令我感受到接受挑戰的喜悅和成就。還有感謝一群曾與我共同在實驗室打拼的夥伴，學長佳鴻、焯基、斯安、懷中、宗祐、同窗麒文、書含、巧翎、學弟志升、宗恆、翊瑄、威銘、嘉桐，你們給予的關心和幫助，豐富了我碩士的生活。

我將最赤誠的感謝，獻予支持我的父母兄長、摯友怡廷和雨欣，因為有你們的鼓勵關懷與支持，我才得以在人生的岔路中找到隱約的方向，論文中的字裡行間是獻給你們的一首感謝的歌，謝謝你們。

同時我將這份喜悅與成果獻給所有在乎我的人，謝謝。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Performance of chip systems has been successfully enhanced by downscaling the chip minimum feature size. This benefits systems to operate at a higher frequency for performing more computations per second. However, the shrinking scale of IC technology also results in higher power density, more metal layers with higher densities, and dielectric materials with lower thermal conductivities, which significantly impacts on both signal propagation delay and manufacturing reliability. Consequently, the advantages of 3D ICs become an attractive solution to relieve 2DIC bottleneck.

As the number of functions on one chip increases, more and more products have involved the stacking of dies to minimize system size. Nowadays, there are several three-dimension integration chip (3DIC) approaches that deal with cost-effective achievement by striking a balance between planar interconnects and vertical through-silicon vias (TSVs). 3D ICs are regarded as an attractive alternative solution for overcoming the wire-length bottleneck of interconnection and the physics of lithography on 2D planar ICs. three-dimension integration chips (3D ICs) offer the increased system a large number of advantages such as miniaturization in chip area, , reducing form factor, reducing wire length, reducing interconnect delays [1] [2], lower power consumption [3], supporting for mixed-technology chips, and heterogeneous SoC design.

1.1 Background on Thermal Simulation

However, there are several challenges such as TSVs, micro-bump, low temperature bonding, and thin-wafer handling of manufacturing, thermal management, design methodologies and tools of design, known-good-die and yield-improving of testing. One of the critical challenges that IC designers must consider is heat dissipation due to higher accumulated power density and lower thermal conductivity of inter-layer dielectrics [4] [5] for the vertical stacking of active tiers and interposer tiers. Furthermore, several previous studies of both computer aided design (CAD) tools and reference flow for 3D ICs [1] [6] indicate that the management of thermal issues plays an important role in the future generation of highly integrated systems due to the highly complex problem of heat dissipating on 3D ICs. That is to say, thermal analysis should be concerned not only on the verification level but also on the design level.

The algorithm of thermal simulators can be categorized into analytical and numerical methods. Analytical methods [7] [8] [9] are suitable for early design stages since the advantages on both avoiding the volume meshing calculation of entire substrate, and representing closed-form by modeling for temperature distribution. Hence, analytical methods are flexible to obtain the temperature distribution of certain user-specified regions without performing the thermal simulation for the entire chip.

The other category of thermal simulators is the numerical method, which applies the finite-difference method (FDM) or finite-element method to transform heat-transfer equations into thermal-circuit equations. For the constructed circuit network, several methods have been proposed for saving run-time. [10] analyzed the efficiency of simulating execution on transient state by model order reduction technique. [11] applied the multi-grid algorithm to improve the convergent rate of iterations and proposed a reduction scheme to reduce the runtime of dynamic thermal simulation. [12] utilized alternating-direction-implicit (ADI) method to separate the equivalent circuit system into different alternating subsystems by considering the property of linear complexity on each sub-system.

In recent years, great importance has been made in the area of 3DIC thermal analysis. Furthermore, because of the stacking material and techniques on 3D ICs, the extensive range of local heating and vertical dissipation of temperature distribution must be concerned. For designers of high-performance 3D ICs, it is essential to apply an accurate simulator by considering adjacent heating to enhance simulating reliability. In addition, because of the flexibility for handling the complicated structure, the numerical category is treated as the main stream on back-end design stages such as the post-layout thermal verification. However, several numerical approaches handling with simulation model on uniform slicing are performance-limited by the most critical position such as thermal or signal through silicon vias (TSVs) [12]. [12] consumed some runtime and hardware resource on analyzing uniform geometry for the much more non-critical positions such as substrate. In order to be more efficient, several approaches can solve adaptive simulating geometry. For example, [13] handled both uniform and non-uniform slicing due to the connecting rule between blocks; however, the characteristic also caused the simulating limitation to be suitable on the floor-plan stage rather than on the placement stage. This limitation of simulating number induces unsuitability on both high-performance and function-increasing 3D ICs. Although [14] modified the connecting rule by enhancing grid-based simulating mode, both [13] and [14] neglected the thermal impact for TSVs. On the other hand, techniques such as [11] [13] [15] and [16] have been developed to construct adaptive thermal modeling, but they are suitable to be verification tools rather than thermal-driven design kernel because of the incremental incapability. Indeed, great importance of thermal analysis has been made on both physical design and verification of 3D ICs [1] [6].

1.2 Our Contributions

To summarize, we are eager to develop an algorithm with the following advantages for both tape-out verification and high-performance physical design.

a). low-complexity algorithm for achieving both efficient runtime and accuracy.

b). referring estimator to provide a thermal reference for heat trend (where heat trend means estimation of power distribution, TSV cells, and thermal conductance using the compact thermal model).

c). adaptive simulating geometry for avoiding unwanted runtime and resource consumption in non-critical position.

d). partial updating analysis for local refinement and incremental characteristic.

In order to achieve these goals, this work derives several processes, including estimating heat trend, constructing initial adaptive grids, establishing non-uniform structure, and calculating 3D-AADI matrix. Although local-refined process of our whole flow (shown in Fig. 5.1) has not been addressed, the major contributions of this work are:

1. **Initial Heat Estimator:**

   We can develop a referring estimator to provide the heat trend as simulating reference by applying z-tile compact thermal model. According to the curve of grid-number and heat distribution, we set criteria and find out a suitable-analyzing resolution by taking chip-size, runtime, and accuracy into consideration.

2. **Initial Adaptive Simulation Grids:**

   We can strike a balance between the runtime and the accuracy. The simulating geometry can avoid being performance-restricted by the most critical position usually on hot spots and (thermal or signal) TSVs, since we develop the grid-construction processes depending on heat gradients to build adaptive geometry of simulation grids, where heat gradient means the gradient of the heat trend.

3. **Non-Uniform Meshing Construction and Calculation of 3D Adaptive ADI (3D-AADI):**

   We can construct adaptive simulating geometry and develop non-uniform scanning algorithm to handle non-uniform meshing which traditional ADI methods utilized on uniform spatial step size of each direction can not deal with. Besides, We are the first one who are eager for the property of linear complexity of ADI concept by concerning both a huge amount of computation and incremental updating during temperature-aware design on 3D chips.

4. **One of Good Solutions for Application on Thermal-Driven Design Due to Incremental Property:**

   3D-AADI thermal conductance matrix of non-uniform meshing can be locally updated on only the affected region rather than entirely re-construct and do LU-decomposition entirely. In this way, it is a good solution to be treated as a thermal kernel in temperature-aware design flow due to not only the non-uniform ability of node meshing but also the incremental data structures of the updated sub-circuit.

1.3 Organization of the Thesis

The rest of the paper is organized as follows. We first formulate our concerning problem in Chapter 2. Then we illustrate the overview of the whole 3D-AADI procedures in Chapter 3 and organize it into four parts which are initial heat estimator in section 3.1, adaptive simulation grids in section 3.2, non-uniform meshing construction on 3D-AADI in section 3.3, non-uniform meshing calculation on 3D-AADI in section 3.4. Besides, we discuss local refinement and incremental property in section 5.2 as the applications and future works. In the last chapter, we summarize the contributions, provide concluding remarks and analyze the experimental results about our 3D-AADI work.

Chapter 2

Problem Formulation

The geometric information of device location and power dissipation on placement are known as input data. In fact, the input information we need is the material structure and the power distribution, so 3D-AADI thermal simulator can be applied to the stages from partition level to tape-out verification level. This simulator can provide temperature distribution profiles on each simulating tier including the active CMOS layers, interposer layers, and substrate layers.

Since the operating frequency of today's manufacturing technology node is much faster than the speed of heat dissipation, it is reliable to analyze temperature distribution on steady state [17] [9]. We extract the power library by power simulating tools, considering internal, switching and leakage power as the total power of each device. Based on [18], the heat sink thermal conductance of the primary and secondary heat transfer paths can be modeled as components and connected with thermal conductance into simulation meshing. Since the ambient air has a constant temperature, the primary and secondary heat sink models respectively connect the bottom and the top terminal with a constant voltage, room temperature node. Fig. 2.1 shows the compact thermal model of N layer 3D IC.

Furthermore, [15] showed the effects of various cooling techniques and indicated that through dummy thermal vias had a very effective cooling influence on temperature dissipation. In this way, TSVs must be considered in 3DIC thermal simulation. For the reason that the routing detail of interconnects on the given placement design is not complete, we build each interconnect layer and TSV alignment routing interposer layer into equivalent thermal model based on the weighted summation of each material thermal conductance. As a result of the package structures, the heat-transfer effect of air on vertical surfaces is strictly much less than that of primary

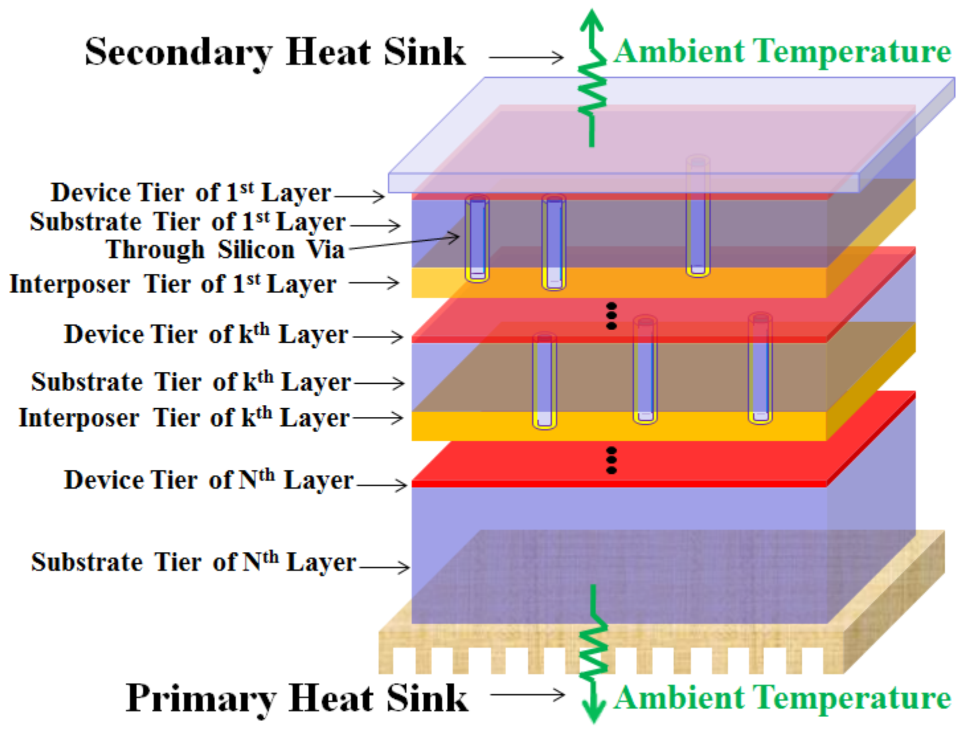

Fig. 2.1: The considering geometry of N layer 3D chip.

and secondary heat sinks on horizontal surfaces. For these reasons, the boundary of four vertical surfaces can be treated as adiabatic [8] [9].

Based on the law of energy conservation, thermal approaches analyze temperature distribution of heat transformation [19] by the following partial differential equations, Eq.(2.1).

$$\rho C_p \frac{\partial T(\overrightarrow{r},t)}{\partial t} = \nabla \cdot [k(\overrightarrow{r},t)\nabla T(\overrightarrow{r},t)] + g(\overrightarrow{r},t) \tag{2.1}$$

subject to the thermal boundary conditions

$$k(\overrightarrow{r},t)\frac{\partial T(\overrightarrow{r},t)}{\partial n_i} + h_i T(\overrightarrow{r},t) = f_i(\overrightarrow{r_{s_i}},t), \tag{2.2}$$

where ρ is the material density, C_p is the mass heat capacity, $T(\overrightarrow{r},t)$ and $k(\overrightarrow{r},t)$ are time- and space- dependent temperature and thermal conductivity of the material, $g(\overrightarrow{r},t)$is the generating rate of heat source, h_i is the heat transfer coefficient on the boundary surface of the chip,

$f_i(\overrightarrow{r_{s_i}}, t)$ is an arbitrary function on the boundary surface s_i, and $\frac{\partial}{\partial n_i}$ is the differentiation along the outward direction normal to the boundary surface s_i.

Since we apply the heat-transfer equation to steady-state condition, the left part of Eq.(2.1), which is not variable with time parameter is equal to zero shown on Eq.(2.3). The steady-state heat-transfer equation Eq.(2.4) can be derived as follows.

$$0 = \frac{\partial T(\overrightarrow{r}, t)}{\partial t} = \frac{1}{\rho C_p} \nabla \cdot [k(\overrightarrow{r}, t) \nabla T(\overrightarrow{r}, t)] + \frac{1}{\rho C_p} g(\overrightarrow{r}, t) \tag{2.3}$$

$$-\frac{1}{\rho C_p} g(\overrightarrow{r}, t) = \frac{k}{\rho C_p} \nabla \cdot [\nabla T(\overrightarrow{r}, t)] \tag{2.4}$$

In order to solve the temperature distribution with finite-difference method and finite-element method, we apply the central-finite-difference approximation and discretize Eq.(2.4) in the spatial domain to construct the given design into equivalent thermal conductance circuits shown in Fig. 2.2. After modeling this equivalent circuit, it can be stamped into matrix equation, Eq.(2.5), based on modified nodal analysis (MNA) method. The parameter $\mathbf{G}$ is the thermal conductance coefficient of corresponding grid by equivalent weighted summation, parameter $\mathbf{t}$ represents the concerning temperature vector of corresponding grids, and $\mathbf{p}$ is the heat source vector of corresponding grids.

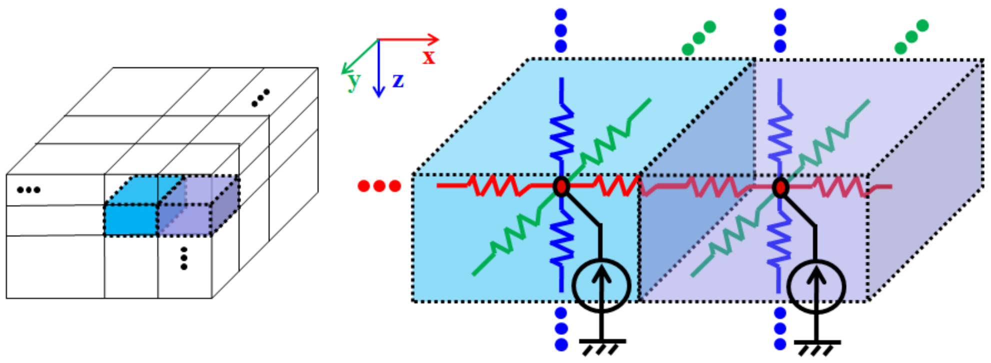

Fig. 2.2: The thermal circuit model of this numerical thermal simulation on steady state.

$$\mathbf{Gt} = \mathbf{p} \tag{2.5}$$

Chapter 3

The Framework of 3D-AADI Simulator

We approach a both adaptive thermal simulator and integrable thermal-aware kernel on 3D ICs. It considers three-dimension structure [6] practically and provides the whole 3DIC temperature distribution of steady state. In order to generate the adaptive simulating geometry depending on thermal gradients at the beginning, we utilize the characteristics of heat dissipation as our initial heat estimator. Furthermore, we develop our 3D-AADI algorithm that not only has the property of linear complexity based on the ADI concept but also applies it into non-uniform simulation grids on both construction and calculation. Since we do take incremental application, local refinement, and temperature-aware design into consideration, 3D-AADI algorithm can be applied to local refinement and incremental updating during design iterations. That is to say, this simulator can both provide adaptive temperature distribution profile depending on user-defined or estimator-suggesting granularity, and be suitable to be applied to 3DIC CAD flow.

For temperature verification, Fig. 3.1 gives an overview. At the beginning, simulator parses 3DIC geometry, power library and material files as input information to build heat sources and thermal elements of simulation grids. Then, the first process for 3D-AADI is to provide heat-trend curve as estimator-defined suggestion or to initially discretize the chip into grids of user-given resolution. The second process of 3D-AADI is to establish the adaptive simulating geometry by referring the estimating heat gradients. The third is to construct non-uniform thermal circuits, and the fourth is to solve matrices of non-uniform meshing by 3D-AADI algorithm to provide the temperature distribution profiles as output results.

On the other hand, local refinement and incremental-kernel application are discussed as future works on section 5.2.

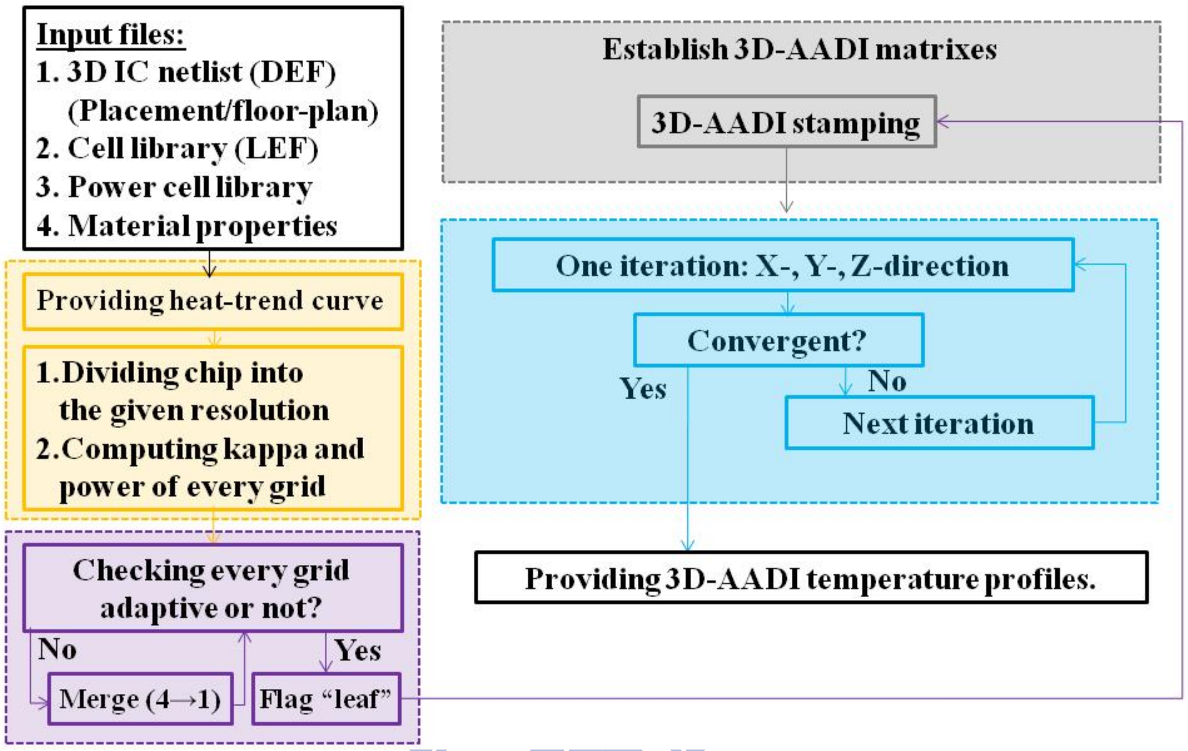
Input files:
1. 3D IC netlist (DEF) (Placement/floor-plan)
2. Cell library (LEF)
3. Power cell library
4. Material properties
Providing heat-trend curve
1.Dividing chip into the given resolution
2.Computing kappa and power of every grid
Checking every grid adaptive or not?
No
Yes
Merge (4→1)
Flag "leaf"
Establish 3D-AADI matrixes
3D-AADI stamping
One iteration: X-, Y-, Z-direction
Convergent?
Yes
No
Next iteration
Providing 3D-AADI temperature profiles.

Fig. 3.1: An overview of this work (which is in the background-color blocks). Authors explain the left-middle orange block on section 3.1, the left-down purple block on section 3.2, the right-up gray block on section 3.3, and the right-middle blue block on section 3.4.

3.1 Initial Heat Estimator

As a quick estimator, we utilize 1D tiles which is composed of the compact thermal model (shown in Fig. 3.2) being our initial temperature estimator to obtain heat estimation. Concerning the accuracy of the compact thermal modeling, there are two reasons that the heat-dissipating influence on vertical is much more obvious than on horizontal. First, the primary heat-dissipating path, including the primary and the secondary heat sinks decided by chip package, is vertical. Second, the boundary of four vertical surfaces can be treated as adiabatic [8] [9]. However, different estimating resolution causes different error rate. For the reason that 1D thermal model regards lateral heat dissipation as adiabatic between grids, and puts total power on the center of each units, it indicates that the bigger grid-number makes both a larger amount of error due to ignoring more lateral heat transformation and a larger amount heat distributing information of each grid. On the other hand, smaller grid-number makes both a smaller amount of modeling error due to considering more lateral heat transformation and a larger amount temperature-like information of averaging feature. For these reasons, we can balance the trade-off between heat-distributing information and temperature-like information. In this way, plotting curves shown in Fig. 3.3 indicates us the suitable estimating grid-number resulting from ignoring suitable amount of lateral heat transformation.

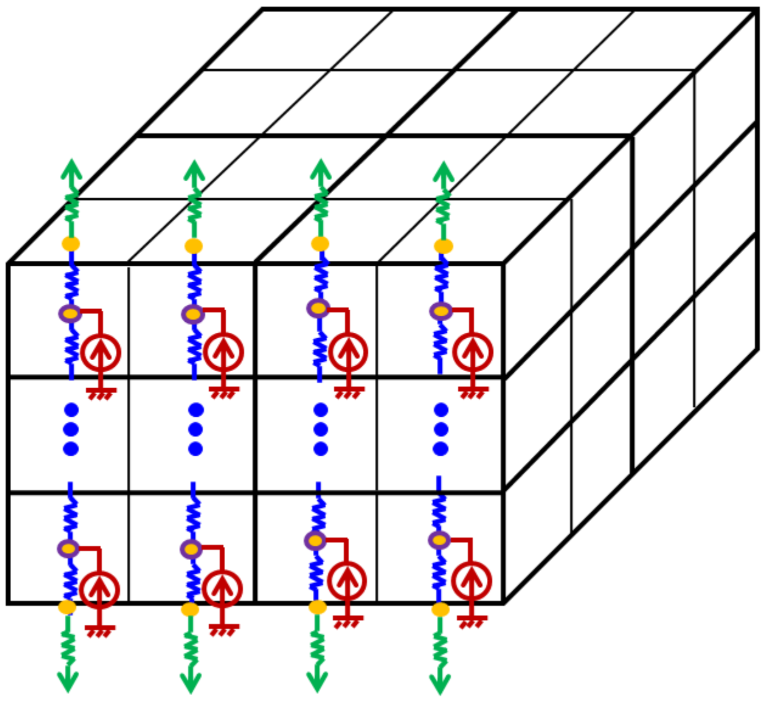

Fig. 3.2: The z-tile model of proposed initial estimator.

As soon as the simulator constructs the cell (including TSVs) information of power, material and position, curves (shown in Fig. 3.3 of grid number and heat difference display two parts, the linear region and the saturation region. The linear region for low grid number presents too much average property to concern accurate power distribution. On the other side, the saturation region for high grid number presents too much power distribution to concern lateral heat dissipation. In order to choose good suggesting resolution, we set criteria by abandoning the extreme and by obtaining enough heat information. On Eq.(3.1), we choose the integer level whose average-heat-difference changing rate is stable than its previous level,so we find the integer i such that average-heat-difference changing rate of $level_i$ is very close to that of $level_{i+1}$.

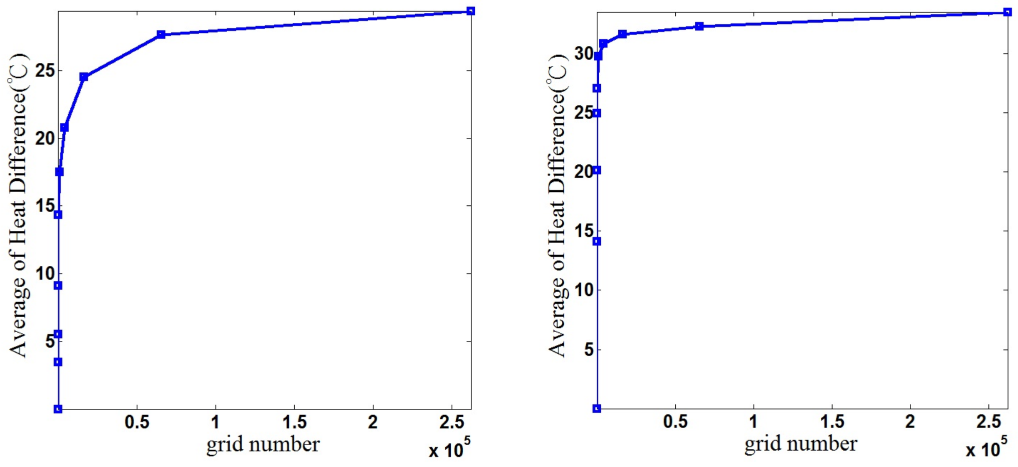

Fig. 3.3: The z-tile curves built by initial heat estimator for test chip 1 (on left one) and test chip 2 (on right one). The x-scale presents planer grid number which is $grid_x$ by $grid_y$ of every simulating z-tier. The y-scale presents the average of heat difference comparing with average temperature of the entire chip.

$$1 < \frac{(AvgErr_i - AvgErr_{i-1})/AvgErr_{i-1}}{(AvgErr_{i+1} - AvgErr_i)/AvgErr_i} < 1 + \varepsilon \tag{3.1}$$

where $AvgErr_i$ is the average of heat difference of i resolution reference by comparing with chip average temperature, ε is a small and positive real number. Then, we divide the whole chip into grids of suggesting level, which can be user-given beforehand or estimator-defined by Eq.(5.1). In this way, the initial estimator applying 1D z-tile method for obtaining the trend of thermal dissipation is reliable to be the reference approximation on proper simulating grid size.

3.2 Adaptive Simulation Grids

Although there have been several approaches that applied finite-difference method to analyzing thermal dissipation, the runtime and the accuracy of their simulator are performance-limited by the most critical position usually on hot spots and (thermal or signal) TSVs. We develop our adaptive structure of simulation grids to eliminate the limited performance on spatial size of critical simulated elements.

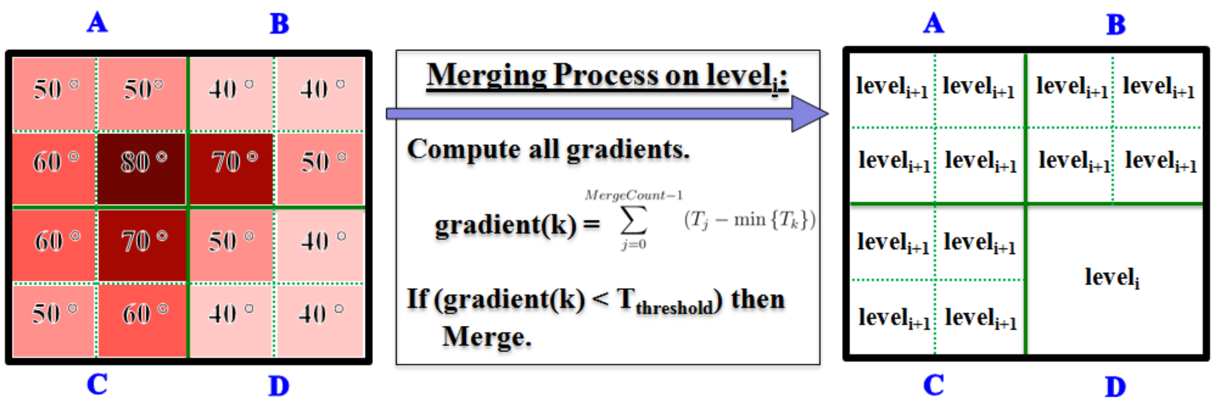
A
B
50° 50° 40° 40°
60° 80° 70° 50°
60° 70° 50° 40°
50° 60° 40° 40°
C
D
Merging Process on level$_i$:
Compute all gradients.
gradient(k) = $\sum_{j=0}^{MergeCount-1} (T_j - \min\{T_k\})$
If (gradient(k) < T$_{threshold}$) then Merge.
level$_{i+1}$
level$_i$

Fig. 3.4: The merging processes of each level decide the analyzing geometry by computing the gradients of initial estimator. The figure is a simple example of grid-merging process after the estimating temperature distribution is computed.

In order to construct adaptive simulation geometry, we approach "level" as the simulating size of grids. Take Fig. 3.4 for example, the finer grids belong to the higher level, and the coarser grids belong to lower level. As shown in Fig. 3.4, we illustrate a simple example of merging process. First of all, the heat information provided by z-tile estimator has been set up on part A, B, C, and D, each one of which is composed of four $level_{i+1}$ grids on $level_i$ merging process. Secondly, we divide the section into finer grids with larger gradients and to merge the section to coarser grid with lower gradients, where "gradient" represents the summation of heat differences for four $level_{i+1}$ children in $level_i$ merging process. After choosing the lowest heat information of four members for being the comparing base, we calculate the gradients on every sections of all levels to make the merging decisions. In order to reveal the maximum difference between children, we define $gradient$ of section k including $MergeCount$ children grids as:

$$gradient(k) = \sum_{j=0}^{MergeCount-1} (T_j - \min\{T_k\}), \tag{3.2}$$

where $MergeCount$ is the children count of the merging section. Then, until each simulating units satisfy the gradient threshold, the entire simulating geometry is constructed. Therefore, the resolution units sliced on initial estimating are merged to coarser level grid size if the gradient between its neighbors is smaller than the given gradient threshold. Otherwise, the original size grids are flagged leaf as the current level in $level_i$ merging. Finally, our simulator, constructing each grid (shown in Fig. 3.6(a)(b)) into equivalent circuit (shown in Fig. 3.6(c)) by using thermal resistant model (shown in Fig. 2.2), considers both lateral and vertical thermal dissipation.

```
3D-AADI: Adaptive Simulation Grids Framework
Input:
  Material structures and power dissipation
Output:
  Data structures of node list on the whole chip
01  Calculate average temperature by total power and
    equivalent kappa of each layer.
02  for each material $Z$ layer do
03      Decide grid number according to gradient of average
        temperature between neighbor layers.
04  end
05  Estimate initial temperature based on Z-tile simulating
    on the defined maximum grid granularity.
06  if LocalRefine == true then
07      Modify MaxLevel
08  for level $i$ = MaxLevel - 1 : 0 do
09      Calculate gradient and average temperature of
        each grid on level $i$.
10  end
11  for level $i$ = MaxLevel - 1 : 0 do
12      for grid $k$ = LevelGridCount - 1 : 0 do
13          if LevelGridGradient >= Threshold then
14              Construct(/Make leaf) Node as level $i + 1$
15      end
16      for grid $k$ = LevelGridCount - 1 : 0 do
17          if one of grid neighbors has been Node then
18              Construct(/Merge) Node as level $i$
19      end
20  end
```

Fig. 3.5: *Adaptive Simulation Grids Framework.*

Pseudo code of adaptive simulating grids framework is derived in Fig. 3.5. First of all, we calculate average temperature by total power and equivalent kappa of each layer to decide the z grid number. Then, we establish the initial heat estimation by slicing into user-given or estimator-defined proper granularity, and check every grids until the whole geometry of simulating grids are adaptive for satisfying the gradient threshold.

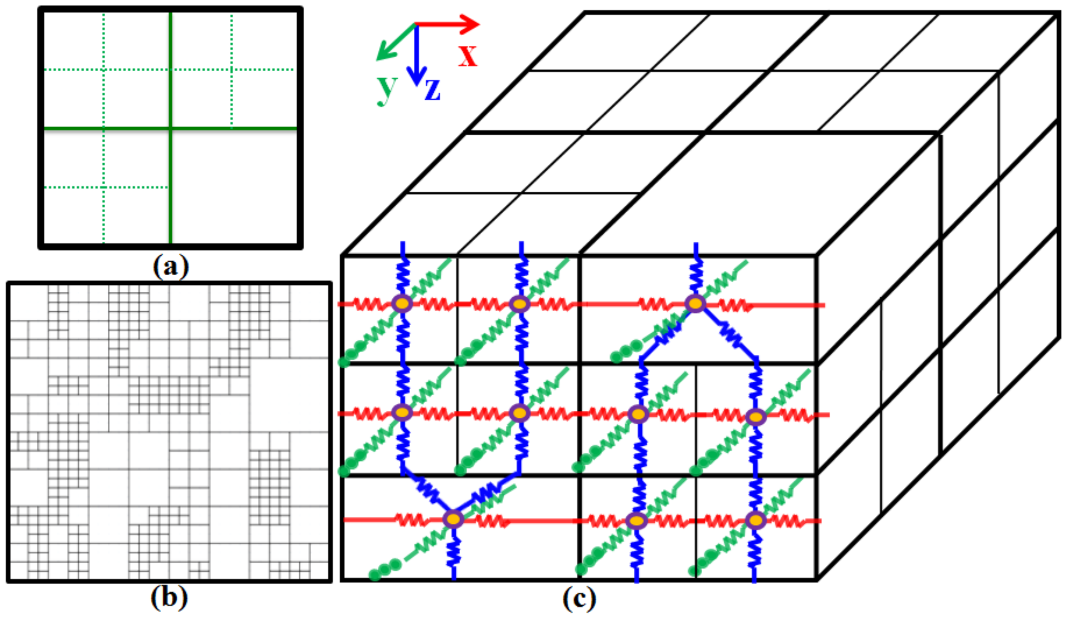

Fig. 3.6: The geometry distribution of adaptive simulation grids considering the gradient of estimating temperature. (a) The front view of chip sliced into simulation grids. (b) The front view of non-uniform simulating case of our test chip 2. This is also the initial adaptive geometry on layer 1 of Fig. 4.8. (c) The perspective view after constructing the equivalent thermal resistant mesh of 3D chip, and the front view is shown on (a).

3.3 Non-Uniform Meshing Construction on 3D-AADI

For a given chip, the temperature distribution in the steady state is governed by Eq.(2.3) and is subject to the boundary conditions in Eq.(2.2). To solve Eq.(2.4) with the finite-difference method, discretization is necessary in spacial domain. When u is differentiable at x, then we have the following limit by Taylor's formula:

$$u(x+h) = u(x) + h\frac{\mathrm{d}u}{\mathrm{d}x} + \frac{h^2}{2}\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \frac{h^3}{6}\frac{\mathrm{d}^3u}{\mathrm{d}x^3} + \frac{h^4}{24}\frac{\mathrm{d}^4u}{\mathrm{d}x^4}(\xi_+) \tag{3.3}$$

for some (ξ_+) in the interval $(x, x+h)$ and

$$u(x-h) = u(x) - h\frac{\mathrm{d}u}{\mathrm{d}x} + \frac{h^2}{2}\frac{\mathrm{d}^2u}{\mathrm{d}x^2} - \frac{h^3}{6}\frac{\mathrm{d}^3u}{\mathrm{d}x^3} + \frac{h^4}{24}\frac{\mathrm{d}^4u}{\mathrm{d}x^4}(\xi_-) \tag{3.4}$$

in which (ξ_-) belongs to the interval $(x-h, x)$. Therefore, adding Eq.(3.4) and Eq.(3.3), and dividing through by h^2, the approximation of the second order derivative of u at x is

$$\frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2} = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} - \frac{h^2}{12}\frac{\mathrm{d}^4 u}{\mathrm{d}x^4}(\xi), \tag{3.5}$$

where $\xi_- \le \xi \le \xi_+$. The formula is called a centered difference approximation of the second derivative.

Applying the centered difference approximation in Eq.(3.5) to a finite number of grid points in a second-order parabolic partial differential equation,

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}, \\ u(x,y,z,0) &= u_0(x,y,z), \end{aligned} \tag{3.6}$$

with u (independent of t) prescribed on the boundaries for the $\frac{\partial^2}{\partial x^2}$, $\frac{\partial^2}{\partial y^2}$, and $\frac{\partial^2}{\partial z^2}$ terms, we use a mesh size of Δx for the x variable, Δy for the y variable, and Δz for the z variable to discretize the continuous spacial domain into mesh grids. Then, according to central finite-difference discretization, the second order accurate approximation of the temperature $T(\overrightarrow{r})$ at grid point (i,j,k) can be replaced by $T(i\Delta x, j\Delta y, k\Delta z)$ which is denoted as $T^n_{i,j,k}$ for the rest of the paper with respect to x can be expressed as:

$$\frac{\partial^2 T}{\partial x^2}\Big|^n_{i,j,k} = \frac{T^n_{i-1,j,k} - 2T^n_{i,j,k} + T^n_{i+1,j,k}}{(\Delta x)^2} + O((\Delta x)^2) \tag{3.7a}$$

$$\approx \frac{T^n_{i-1,j,k} - 2T^n_{i,j,k} + T^n_{i+1,j,k}}{(\Delta x)^2} \equiv \frac{\delta_x^2 T^n}{(\Delta x)^2}, \tag{3.7b}$$

where the truncation error is $O((\Delta x)^2)$, and similar processes can be applied to the y and z directions.

Concerning ADI method, it introduced by Peaceman and Rachford [20], and Douglas and Gunn [21] in the mid-1950s was developed for solving equations arising from finite difference discretization of elliptic and parabolic PDEs.

We define the deviation $e^n_{i,j,k}$ of the evolving solution $T^n_{i,j,k}$ from the desired finite-difference approximation to the steady-state convergence, $T^*_{x,y,z}$. Substituting Eq.(2.4) and Eq.(3.7b), the convergent solution T^* satisfies the equation

$$\mu\left[\frac{\delta_x^2}{(\Delta x)^2} + \frac{\delta_y^2}{(\Delta y)^2} + \frac{\delta_z^2}{(\Delta z)^2}\right] T^* = \mathbf{p} \tag{3.8}$$

where μ is $\frac{k}{\rho C_p}$. Therefore, this deviation is given at the nth iteration by

$$e_{i,j,k}^{n} = T_{i,j,k}^{n} - T_{i,j,k}^{*}. \tag{3.9}$$

Eq.(3.8) using central finite-difference discretization results in the system of equation,

$$\mathbf{Ht} + \mathbf{Vt} + \mathbf{Zt} = \mathbf{p}, \tag{3.10}$$

where $\mathbf{H}$, $\mathbf{V}$, and $\mathbf{Z}$ represent the central finite-difference discretization in Eq.(3.7b) to the operators $\frac{\delta_x^2}{(\Delta x)^2}$, $\frac{\delta_y^2}{(\Delta y)^2}$, and $\frac{\delta_z^2}{(\Delta z)^2}$ in Eq.(3.8), respectively. The ADI algorithm consists of iterating by solving Eq.(3.10) in the x, y, and z directions alternatively as follows.

$$\begin{cases} (\mathbf{H} + \rho_n \mathbf{I}) \cdot \mathbf{t}^{\left(n+\frac{1}{3}\right)} = & \mathbf{p} - (\mathbf{V} + \mathbf{Z} - \rho_\mathbf{n} \mathbf{I}) \cdot \mathbf{t}^{(\mathbf{n})} \\ (\mathbf{V} + \rho_n \mathbf{I}) \cdot \mathbf{t}^{\left(n+\frac{2}{3}\right)} = & \mathbf{p} - (\mathbf{H} + \mathbf{Z} - \rho_\mathbf{n} \mathbf{I}) \cdot \mathbf{t}^{\left(\mathbf{n}+\frac{1}{3}\right)} \\ (\mathbf{Z} + \rho_n \mathbf{I}) \cdot \mathbf{t}^{(n+1)} = & \mathbf{p} - (\mathbf{H} + \mathbf{V} - \rho_\mathbf{n} \mathbf{I}) \cdot \mathbf{t}^{\left(\mathbf{n}+\frac{2}{3}\right)} \end{cases} \tag{3.11}$$

where ρ_n is a sequence of positive acceleration parameters. Since this work applies ADI concept to non-uniform geometry, analyzing of ρ_n is still a difficulty to address.

Take Fig. 3.7(b) for example, when x sub-iteration executing, the one and only uncertainty is x implicit vector, the connection of other directions are explicit. For the executing direction d, the "Scan and do LU-decomposition for every LUVectorLists on d" function in Fig. 3.9 first checks this node being the head (e.g. Fig. 3.8 node 1, 9, 17, 18, and 27) of LU vector or not. For these head nodes, we find all neighbors to direction d and assign this node to be the head of these LU vectors for all branches (i.e. node 1, 9, and 17 have one, node 18 has two, and node 27 has four branch vectors to $x-$direction in Fig. 3.8) on d. Then, for each neighbor, we point to the next only one neighbor (e.g. Fig. 3.8 node 2, 10, 18, 19, 23, 28, 32, 37, and 40) on d and add it to be members of its LU vector until there is not only one neighbor (e.g. Fig. 3.8 node 18) or until the level of pointing node is smaller than that of head node (e.g. Fig. 3.8 node 39). Next, we regard this node as end (e.g. Fig. 3.8 node 8, 16, 18, 22, 26, 31, 35, and 39) of this LU vector and assign the end to be also the head of all next neighbors on d. Finally, after "Solve LUt = p by reusing LU-decomposition for every LUVectorLists on d" function in Fig. 3.13 provides new temperature for this LU vector, we update these new results into data structures. Furthermore, in order to overcome the convergence and to keep the accuracy, we

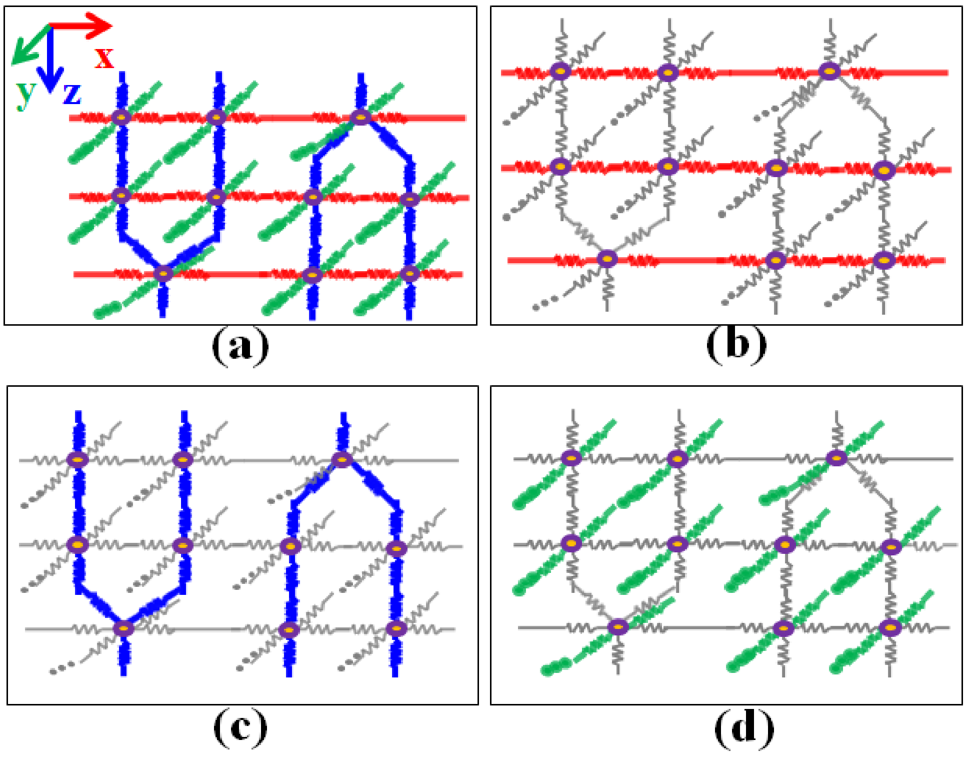

Fig. 3.7: (a) The constructing mesh including all directions. (b)(c)(d) One of different alternative directions utilizing 3D-AADI method when scanning on $x-$, $y-$, and $z-$direction, respectively. The step by step illustrate the framework of scan and solve LUVectorList of d.

treat every nodes with the same LU updating frequency on non-uniform meshing by making the end implicit and the head explicit (e.g. Fig. 3.8 node 18) of the node being both head and end. Besides, we deal with the updating temperature of gathering end (e.g. Fig. 3.8 node 39) to be the average of all these gathering results. Similarly, y-direction and z-direction, the other two sub-iteration utilize the same scanning rule.

Generally speaking, ADI method is one of numerical iterative methods. This method has the properties of linear complexity because of separating the equivalent circuit system (Eq.(2.5)) into different alternating subsystems. This characteristic guarantees that $\mathbf{G}$ matrix of every direction can be performed linearly during LU decomposition. Take G_x matrix, a subsystem on $x-$direction, for example, several sub-LU matrices is shown on Fig. 3.10, where G_{x_k}, one thermal conductance sub-matrix, is tridiagonal and linear time complexity during matrix solving

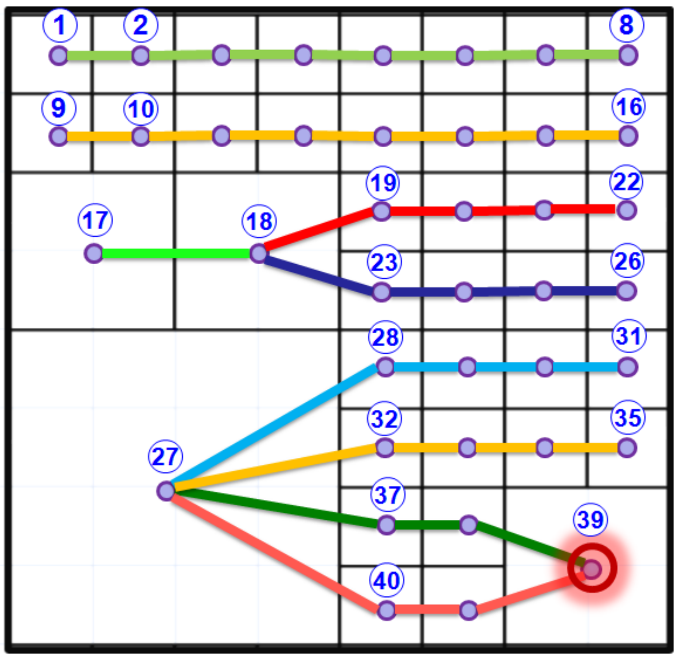

Fig. 3.8: Finding the stamping LU members on x-direction of one of every layers on our 3D-AADI algorithm. The red rings stand for gathering nodes.

. In other words, each direction of ADI method is only related to the front and the back nodes, so solving-matrix execution of G on each direction has linear runtime. Since the strict convergence threshold causes the more times of iteration, the convergence rate of iteration can be controlled by the user given accuracy which is in our guaranteeing range.

3.4 Non-Uniform Meshing Calculation on 3D-AADI

According to the property of linear complexity, ADI algorithm is suitable to be applied to solve the larger scale matrix. Furthermore, in order to conquer the performance limitation due to finite-difference method on both critical and non-critical positions, we develop 3D-AADI algorithm and display the convergence of non-uniform meshing nodes. In addition, authors apply it to solving the non-uniform nodes. The illustration of scanning and deciding the members of every LU lists on x, y and z directions is shown in Fig. 3.8, Fig. 3.11 and Fig. 3.12, respectively. Taking this simulating geometry for example, there are three kinds of simulating size which are constructed by the previous geometry-deciding process (on section 3.2). After scanning and

3D-AADI: Scan and do LU-decomposition for every LUVectorLists on d Framework

Input:

Doing direction d

Output:

LUVector lists and LU-decomposition results of sub-iteration d

```
01 for Node $n$ = NodeCount - 1 : 0 do
02     if ($n$ belongs to boundary of head)
         or (there exists Head) then
03         for Neighbor $nei$ = NeiSize - 1 : 0 do
04             Add $n$ to be Head of this LUVector.
05             This = next neighbor (on $d$) of $n$.
06             while This has only one next neighbor do
07                 Add This into this LUVector.
08                 This = next neighbor (on $d$) of This.
09                 if This.level < Head.level then
10                     break
11             end
12             Add This to be End of this LUVector.
13             for Neighbor $Nnei$ = ThisNeiSize - 1 : 0 do
14                 Add This to be Head of $Nnei$'s LUVector.
15             end
16             if Head belongs to boundary of head then
17                 Stamp subLU members excepting Head.
18             else
19                 Stamp subLU members including Head.
20             subLU-DecompositionSolver(ThisLUVector).
21         end
22 end
```

Fig. 3.9: *Scan and do LU-decomposition for every LUVectorLists on* d *Framework.*

deciding the members of every LU lists, we construct thermal circuits and stamp thermal conductance based on our 3D-AADI algorithm into LU-decomposition solver. (The pseudo code in detail of scanning function is derived in Fig. 3.9.) As shown in Fig. 3.10, it plays a very important role that the execution time of each sub-G_x matrices, G_{x_k}, is linear with number of unknown nodes. It is also important that the dimensions of each sub-matrix, G_{x_1}, G_{x_2}, ..., and G_{x_N}, are independent. In this way, the linear complexity property of each sub-LU matrix implies to the linear complexity property of the entire algorithm. We elaborate the framework of non-uniform meshing calculation on 3D-AADI by Fig. 3.13. First of all, we scan and do LU-decomposition for every LUVectorLists on every d to obtain the LU-decomposition results for reusing during iterations. Secondly, while the maximum of absolute temperature difference at last iteration is bigger than the given convergent threshold, we execute the next iteration in-

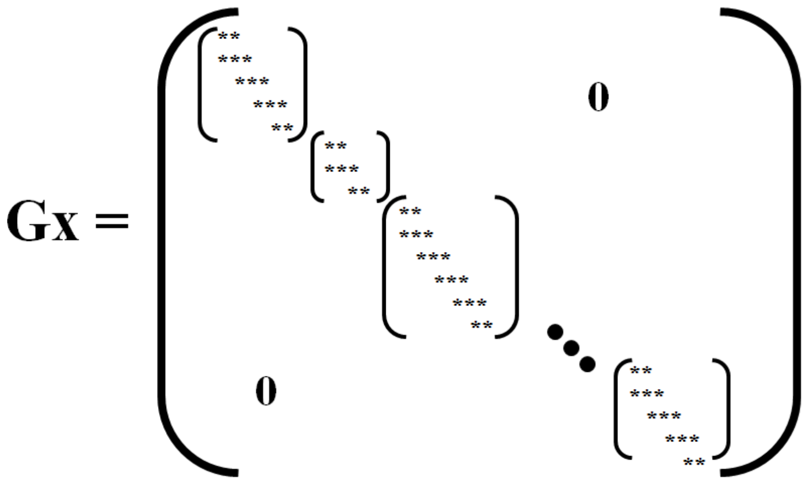

Fig. 3.10: The entire stamping LU matrix of x-direction on our 3D-AADI algorithm. The dimension of unknown-node vector for every sub-LU matrices, G_{x_1}, G_{x_2}, ..., and G_{x_N} can be independent. In this way, the linear complexity property of each sub-LU matrix implies to the linear complexity property of the entire algorithm.

cluding three sub-iteration, x, y and z directions. Concerning the gathering nodes generated by non-uniform construction, we average all gathering results of relating LUVectorLists to balance the convergence and the accuracy. The gathering nodes are defined by red rings shown in Fig. 3.8, Fig. 3.11, and Fig. 3.12. The function named "Scan and do LU-decomposition for every LUVectorLists on d" on line 2 of Fig. 3.13 scans and executes LU-decomposition for every LUVectorLists of direction, d. As long as one entire iteration, including three sub-iteration, is done, we calculate the maximum and average difference temperature to decide whether it is convergent or not.

According to the following derivation, because of the non-uniform meshing of thermal conductance and the independent dimension between sub-LU matrices, convergent iterations and executing time are wasted by waiting different convergence of several matrices with different numerical scale. We apply the matrix splitting [22]

$$\mathbf{G} = \mathbf{M} - \mathbf{N} \tag{3.12}$$

to the constructed original linear system, $\mathbf{G}$, of Eq.(2.5). Then, $(\mathbf{M} - \mathbf{N})\mathbf{t} = \mathbf{p}$ can be defined

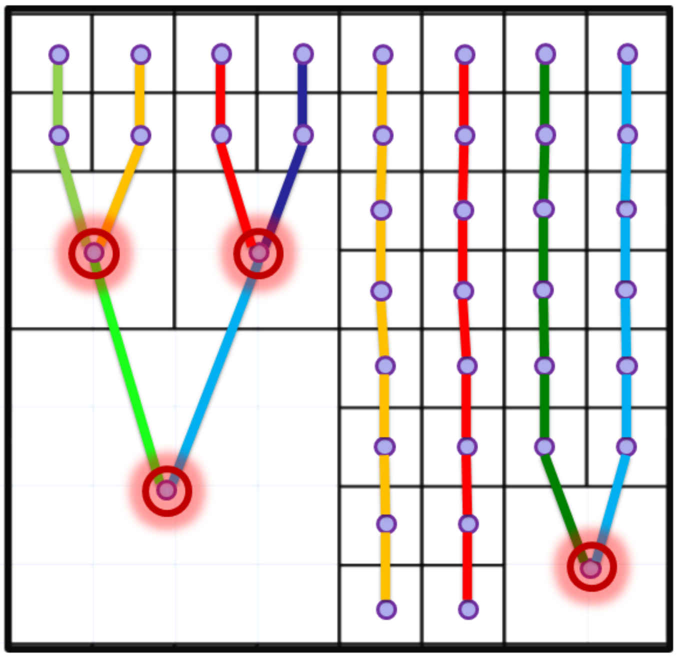
Fig. 3.11: Finding the stamping LU members on y-direction of one of every layers on our 3D-AADI algorithm. The red rings stand for gathering nodes.

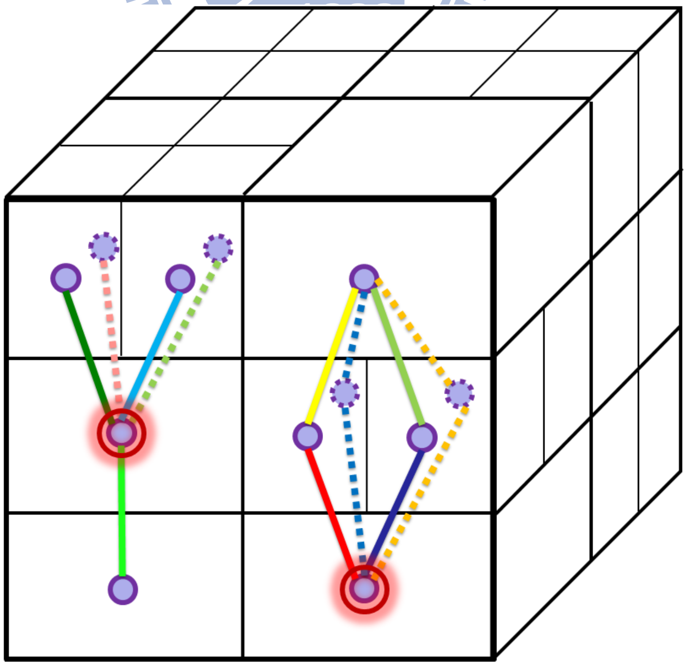
Fig. 3.12: Finding the stamping LU members on z-direction of one of every layers on our 3D-AADI algorithm. The red rings stand for gathering nodes. Besides, the rings with dotted lines stand for the horizontal neighbors on the same layer on z-direction.

```
3D-AADI: Non-Uniform Meshing Calculation Framework
Input:
   Data structures of node list on the whole chip
Output:
   Convergent 3D-AADI temperature distribution

01  for Direction $d$ = 0 : 2 ( $X-, Y-, Z-$ ) do
02      Scan and do LU-decomposition for every LUVectorLists on $d$.(Fig. 3.9)
03  end
04  while (AbsoluteMaxDifferenceT > ConvergentThresholdT)
           or (iteration == 0) do
05      AbsoluteMaxDifferenceT = 0
06      for Direction $d$ = 0 : 2 ( $X-, Y-, Z-$ ) do
07          Solve $LUt = p$ by reusing LU-decomposition for every LUVectorLists on $d$.
08          for Node $n$ = NodeCount - 1 : 0 do
09              if End node belongs to gathering tail then
10                  $n$.newT = average of all $n$.newT.
11              $n$.oldT = $n$.newT.
12          end
13      end
14      Update AbsoluteMaxDifferenceT for checking convergency.
15      iteration ++ .
16  end
```

Fig. 3.13: *Non-Uniform Meshing Calculation Framework.*

by recurrence,

$$\mathbf{t^{k+1}} = \mathbf{M^{-1}Nt^{k}} + \mathbf{M^{-1}p}, \tag{3.13}$$

where $\mathbf{t^{k+1}}$ is implicit and $\mathbf{t^{k}}$ is explicit during iterations. For solving the system, $\mathbf{t^{k}}$ is approaching $\mathbf{t^{k+1}}$ when the iteration converges. Eq.(3.13) has the relation

$$\left(\mathbf{I} - \mathbf{M}^{-1}\mathbf{N}\right)\mathbf{t} = \mathbf{M^{-1}p}.$$

Utilizing Eq.(3.12), we replace **N** with $\mathbf{M} - \mathbf{G}$ and derive as following.

$$\left(\mathbf{I} - \mathbf{M}^{-1}\left(\mathbf{M} - \mathbf{G}\right)\right)\mathbf{t} = \mathbf{M^{-1}p}.$$

$$\left(\mathbf{I} - \mathbf{M}^{-1}\mathbf{M} + \mathbf{M}^{-1}\mathbf{G}\right)\mathbf{t} = \mathbf{M^{-1}p}.$$

$$\mathbf{M}^{-1}\mathbf{G}\mathbf{t} = \mathbf{M^{-1}p}. \tag{3.14}$$

The Jacobi iteration determines the ith component of the next approximation so as to annihilate the ith component of residual vector. In the following, u_i^k denotes the ith component of the iterate $\mathbf{t^k}$ and p_i the ith component of the right-hand side $\mathbf{p}$ in Eq.(2.5). Thus, writing

$$\left(\mathbf{p} - \mathbf{G}\mathbf{t^{k+1}}\right)_i = 0,$$

in which $(\mathbf{vector})_i$ represents the ith component of **vector**, yields

$$g_{ii} u_i^{k+1} = - \sum_{j=1, j \neq i}^{n} g_{ij} u_j^k + p_i$$

or

$$u_i^{k+1} = \frac{1}{g_{ii}} \left(p_i - \sum_{j=1, j \neq i}^{N} g_{ij} u_j^k \right), i = 1, ..., N, \tag{3.15}$$

where g_{ij} is (i, j) component of $\mathbf{G}$. Then, block relaxation schemes [22] are generalizations of point relaxation schemes described in Eq.(3.15). They update typically a subvector of the solution vector, instead of only one component. The matrix $\mathbf{G}$ and the right-hand side and solution vectors are partitioned from Eq.(2.5) as follows:

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} & \cdots & \mathbf{G}_{1m} \\ \mathbf{G}_{21} & \mathbf{G}_{22} & \cdots & \mathbf{G}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_{m1} & \mathbf{G}_{m2} & \cdots & \mathbf{G}_{mm} \end{pmatrix}, \mathbf{t} = \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_m \end{pmatrix}, \mathbf{p} = \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_m \end{pmatrix}, \tag{3.16}$$

in which the partitionings of $\mathbf{p}$ and $\mathbf{t}$ into subvectors $\mathbf{p_i}$ and $\mathbf{t_i}$ are identical and compatible with the partitioning of $\mathbf{G}$. Thus, for any vector $\mathbf{t}$ partitioned as in Eq.(3.16), $[\mathbf{G}\mathbf{t}]_i = \sum_{j=1}^{m} \mathbf{G}_{ij}\mathbf{t_j}$, in which $[\mathbf{vector}]_i$ denotes the ith subvector of **vector** according to the above partitioning.

With the splitting definitions in Eq.(3.12) and the block relaxation schemes in Eq.(3.16), we generalize the previous iterative procedures in Eq.(3.13). Then, the block Jacobi iteration is now defined as a technique in which the new subvectors $\mathbf{t}_i^k$ are all replaced according to

$$\mathbf{t}_i^{k+1} = \mathbf{M}_i^{-1}\mathbf{N}_i\mathbf{t}_i^k + \mathbf{M}_i^{-1}\mathbf{p}_i, i = 1, ..., m, \tag{3.17}$$

where $\mathbf{M}_i$ is the corresponding preconditioning sub-matrix, and $\mathbf{N}_i$ are the corresponding sub-matrix of splitting matrices, $\mathbf{M}$ and $\mathbf{N}$, in Eq.(3.12), respectively. In order to improve the convergence of a preconditioned system in Eq.(3.13), we precondition each sub-matrix in Eq.(3.17) for every diagonal entry before LU-decomposition of each sub-system.

Since the implicit variables are only in one direction in each step, the matrix for solving the ADI method at each direction is tridiagonal. During the LU decomposition, no matrix solving is required, so runtime for solving the tri-diagonal matrix is linear. The complexity for ADI methods is $O(N \cdot 3 \cdot IJK) = O(node_{ADI})$, where N is the iteration number, $node_{ADI}$ is the node account for solving problem, I, J, and K are the discrete number on x, y, and z, respectively. Concerning the proposed 3D-AADI, we regard the nodes being both head and tail are implicit on only being tail, the complexity for 3D-AADI methods is $O(N \cdot [(I_1 + I_2 + ...) + (J_1 + J_2 + ...) + (K_1 + K_2 + ...)]) = O(node_{3DAADI})$, where $node_{3D-AADI}$ is the node account for non-uniform geometry, I_k, J_k, and K_k is the node number of each LU vector on x, y, and z, respectively. For the reason that $node_{3D-AADI}$ is much smaller than $node_{ADI}$, we improve the efficiency by non-uniform geometry.

Chapter 4

Experimental Results

To verify that 3D-AADI can not only provide accurate temperature but speed up on the same resolution, we develop the entire 3D-AADI algorithm and provide the simulating results. Since the complexity of 3D-AADI, which is linear, is much lower than the complexity of direct solving thermal conductance matrix, the higher simulating resolution leads to the more speeding up. Furthermore, 3D-AADI does save execution time and resource, that is to say, it can display the finer simulating resolution on finite hardware resource.

The developed 3D-AADI thermal simulator is implemented in C++ language and tested on a Linux system with Intel Xeon 3.0-GHz CPU and 32 GB memory. The size of given 3D test chip 1, a 3-layer circuit, is 379.06 μm by 379.06 μm for width and height respectively, and the size of given 3D test chip 2, a 3-layer circuit, is 6600 μm by 6600 μm for width and height respectively. The unit of the shown temperature distribution is $^\circ C$. The equivalent heat transfer coefficients of the primary and secondary heat flow paths, and thermal conductivity are 8700 W/($\mathrm{m}\cdot^\circ\mathrm{C}$), 2017 W/($\mathrm{m}\cdot^\circ\mathrm{C}$), and 148 W/($\mathrm{m}\cdot^\circ\mathrm{C}$),respectively. The boundary condition of each vertical surface is set to be isothermal [8] [9].

We demonstrated that our 3D-AADI simulator is much more efficient than traditional uniform ADI and FDM-MNA due to both avoiding the limited simulating performance of the most critical position and to the mathematical complexity. The approach can be not only adaptive for verification but incremental for application to 3D ICs especially on physical design due to the huge computation. Furthermore, the 3D-AADI tool can be regard as both a reliable thermal simulator and a thermal-driven kernel on 3DIC design flow.

4.1 Adaptive Distribution Results

In order to strike a balance between the runtime and accuracy, 3D-AADI simulating geometry can avoid being performance-restricted by the most critical position usually on hot spots and (thermal or signal) TSVs, because we develop the merging processes depending on heat gradients to construct adaptive geometry of simulation grids.

We compare adaptive simulating geometry with traditional uniform slicing to verify that 3D-AADI does avoid the performance limitation by merging the much less critical position. In this way, we display temperature by golden solution, uniform slicing of traditional ADI, and adaptive slicing of 3D-AADI respectively to demonstrate our adaptive contribution.

Fig. 4.1 shows the temperature by Finite Difference Method on Modified Nodal Analysis (FDM-MNA) temperature solver. Since the accuracy of FDM-MNA has been compared with SPICE by the same FDM thermal model, we treat FDM-MNA results as simulating golden solution. In order to compare both the convergence and accuracy, Fig. 4.2 shows temperature distribution of every layer. Fig. 4.3 shows the solution which is convergent to 0.01% providing by 3D-AADI algorithm with non-uniform grids. Fig. 4.4 shows the temperature which is convergent to 0.0001% by our 3D-AADI algorithm, where "convergent" means the maximum comparison rate of every corresponding-grid temperature between this and lase iteration.

Although the temperature range in Fig. 4.2 and Fig. 4.3 is too small to demonstrate the accuracy between 0.01% and 0.0001% convergence, our 3D-AADI can speed up 274 times than FDM-MNA (shown on Table 4.1) especially for higher resolution. According to Fig. 4.6 and Fig. 4.7, they point out that 3D-AADI solution is both very fast (61 times than FDM-MNA shown in Table 4.3) and very approaching to golden solution when it is 0.01% convergence. As a result of linear complexity, 3D-AADI has much higher efficiency, 117 times, than FDM-MNA. Besides, it is the same with golden solution when it is convergent to 0.0001% shown in Fig. 4.8.

We can clearly indicate that our 3D-AADI tool can be regarded as both a reliable thermal simulator and a thermal-driven kernel on 3D IC design flow. First, we display the results by traditional ADI algorithm and compare with our golden solution, FDM-MNA under the same simulating-grid geometry. Then, the maximum error percentage is under 10E-6. Furthermore,

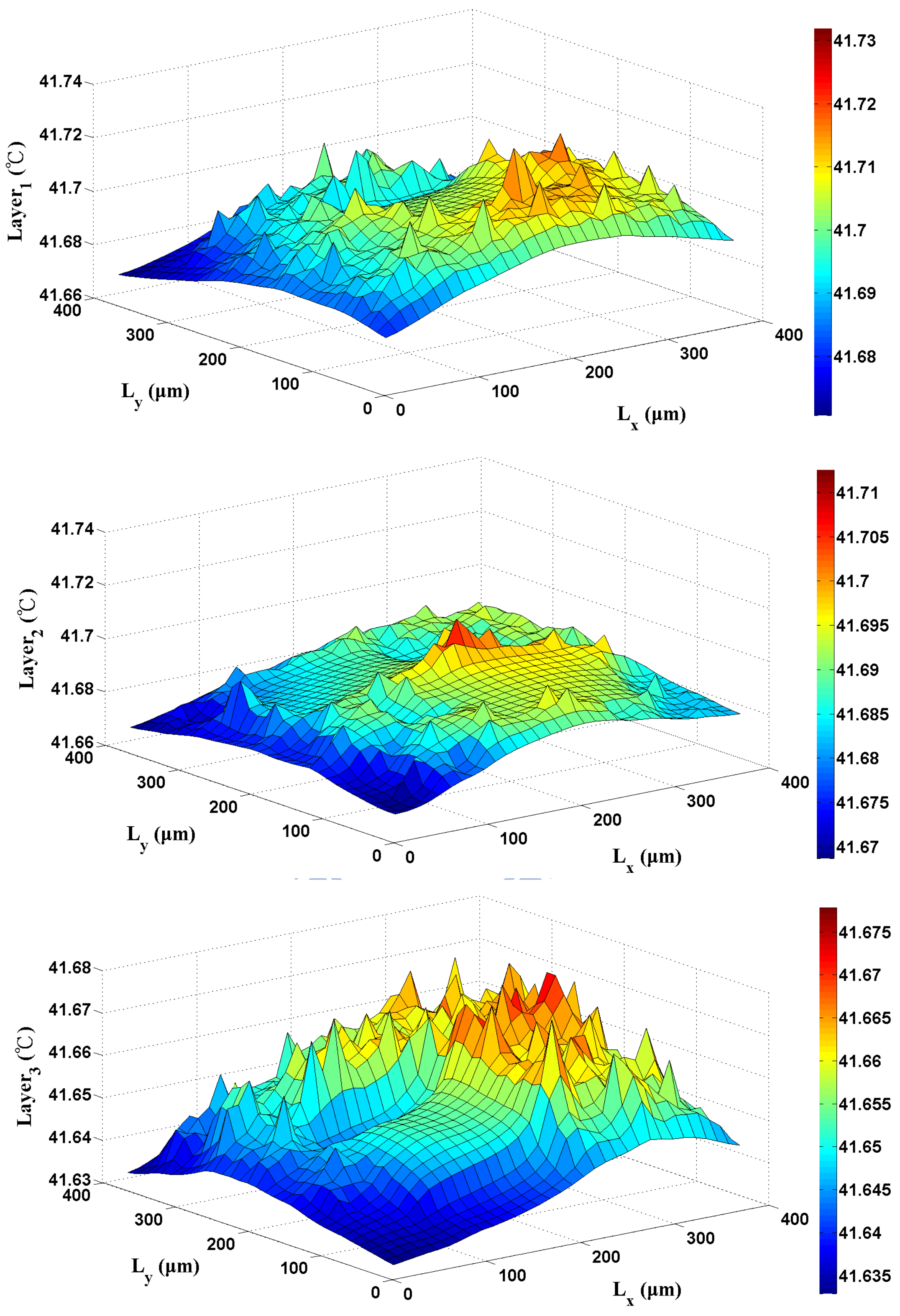

Fig. 4.1: Finite Difference Method on Modified Nodal Analysis (FDM-MNA) Results on test chip 1. We regard the FDM-MNA temperature distribution as our golden solution and compare the error rate with its average degree. The temperature profiles are shown by tiers of every active layers. Actually, our simulating tiers on z can be much larger than active numbers depending on user-setting accuracy.

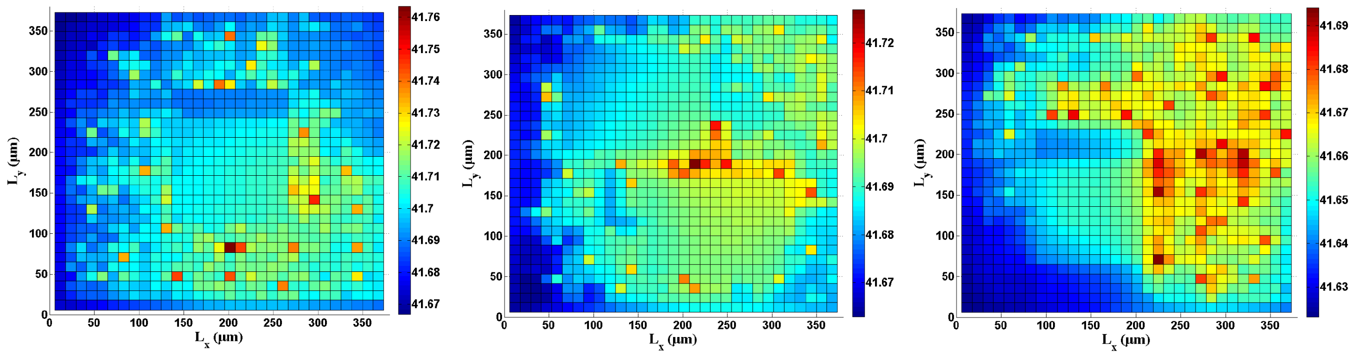

Fig. 4.2: FDM-MNA golden solution temperature with grids-number 32 by 32 of test case 1.

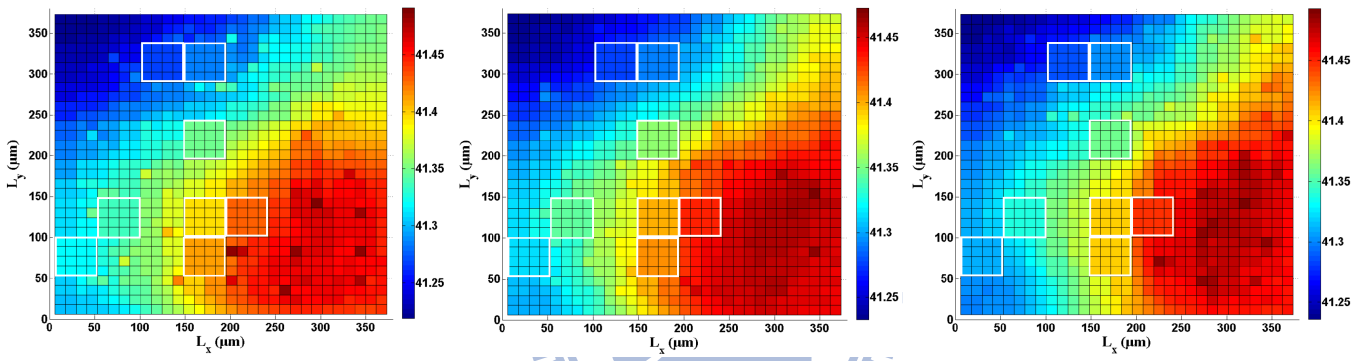

Fig. 4.3: The solution which is convergent to 0.01% providing by 3D-AADI algorithm with non-uniform grids of test case 1.

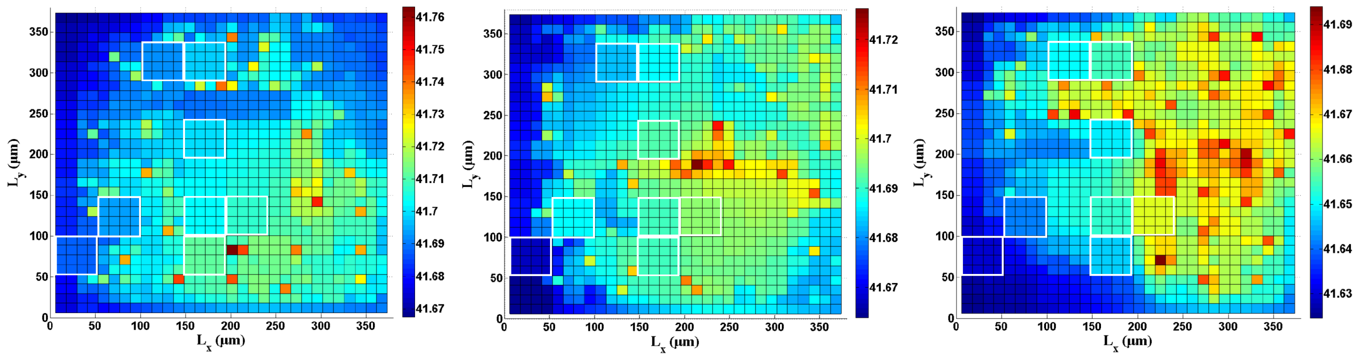

Fig. 4.4: Temperature solution which is convergent to 0.0001% providing by our 3D-AADI algorithm with non-uniform grids of test case 1.

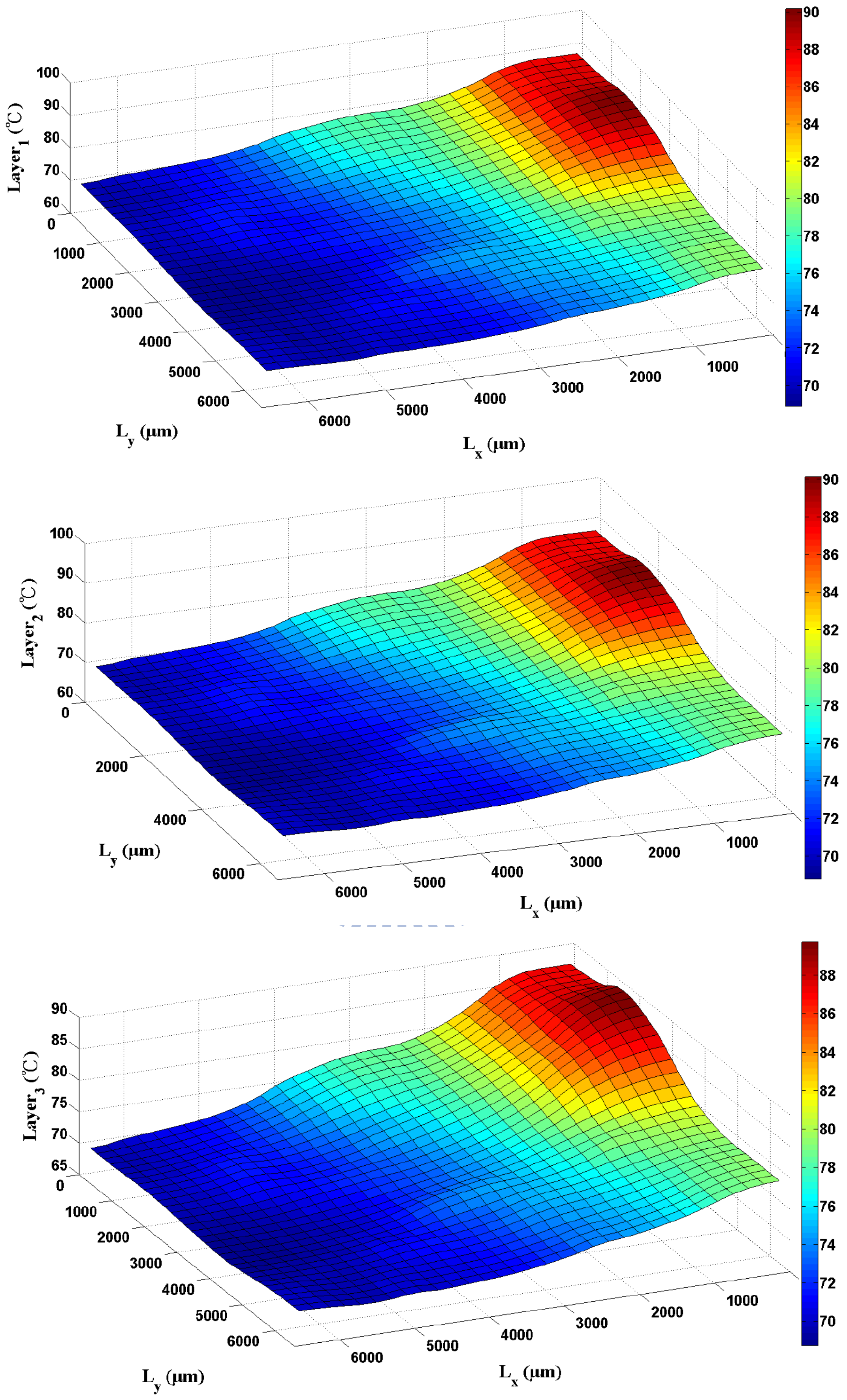

Fig. 4.5: Finite Difference Method on Modified Nodal Analysis (FDM-MNA) Results on test chip 2. We regard the FDM-MNA temperature distribution as our golden solution and compare the error rate with its average degree. The temperature profiles are shown by tiers of every active layers. Actually, our simulating tiers on z can be much larger than active numbers depending on user-setting accuracy.

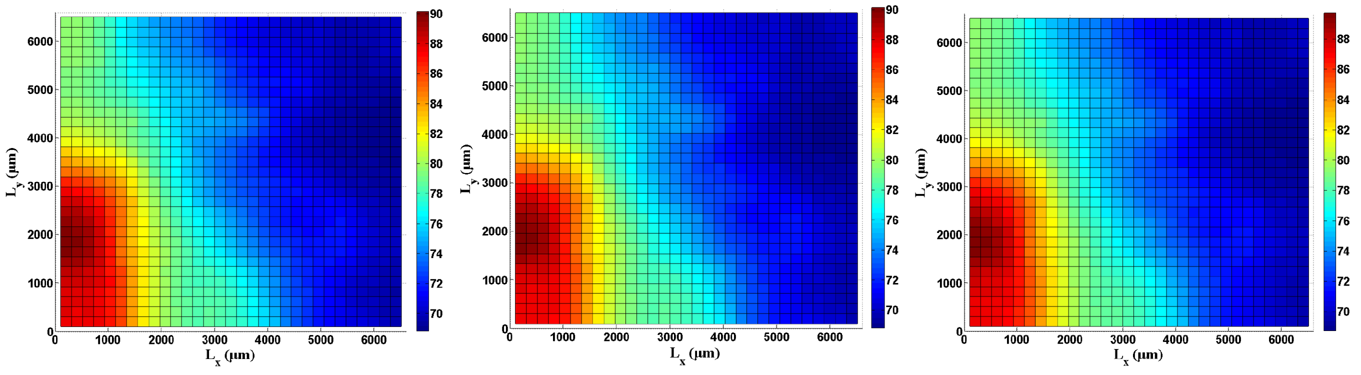
Fig. 4.6: FDM-MNA golden solution temperature with grids-number 32 by 32 of test case 2.

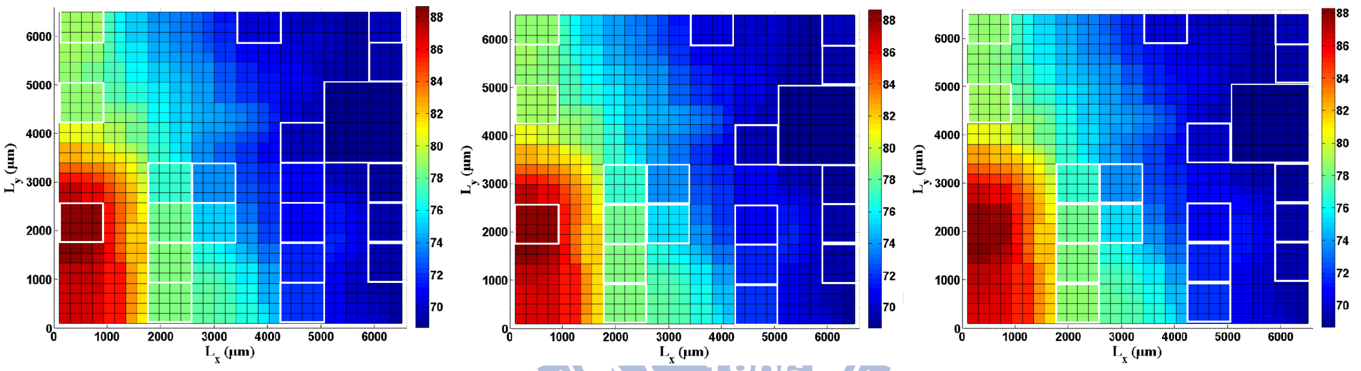
Fig. 4.7: The solution which is convergent to 0.01% providing by 3D-AADI algorithm with non-uniform grids of test case 2.

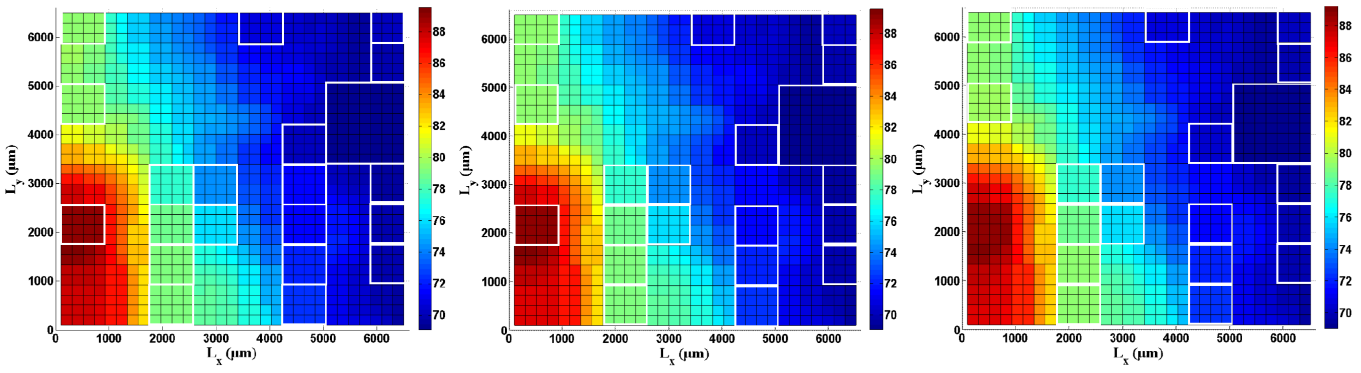
Fig. 4.8: Temperature solution which is convergent to 0.0001% providing by our 3D-AADI algorithm with non-uniform grids of test case 2.

we implement our 3D-AADI algorithm and compare with both golden solution and traditional ADI, the results (For example, 100-200 Lx and 100-200 Ly on layer 3 in Fig. 4.4 are merged. For example, 200-380 Lx and 0-380 Ly on layer 3 in Fig. 4.8 are merged.) are convergent to golden solution and our simulating-grid geometry is much more adaptive.

Furthermore, the comparison also indicates that an accurate temperature simulation should take lateral heat-transformation into consideration, because z-tile method, which applies 1-D compact thermal model to temperature estimators, displays the influence on only power source and material without lateral heat spreading. As the increasing number of simulating grids shown on Fig 3.3 , both the range of temperature and the error percentage increase.

4.2 Comparison of Convergence Results

We can construct adaptive simulating geometry and develop non-uniform scanning algorithm to handle non-uniform meshing which traditional ADI method utilized on uniform spatial step size of each direction can not solve. Besides, since we concern both a huge amount of computation and incremental updating during temperature-aware design on 3D chips, the LU and LUx complexity of 3D-AADI is linear.

Since we have demonstrated the adaptive geometry on section 4.1, we extract the execution time and the maximum error between non-uniform slicing, uniform slicing and FDM-MNA simulation methods in this section shown on Table 4.1 and Table 4.3. We can find out not only that all of them are speed up but also that the finer simulating granularity results in the more tremendous speeding up. In addition to time enhancement, since non-critical position is preserved from both unwanted runtime and resource consumption by the proposed adaptive simulating-geometry process, 3D-AADI can also reduce the maximum error rate shown in Table 4.3.

Concerning each process, Table 4.2 and Table 4.4 demonstrate that the partial executing time is linear-like growth for both LU and LUx. It is the reason that 3D-AADI is low-complexity algorithm for achieving both efficient runtime and accuracy.

We compare the maximum temperature difference and runtime between non-uniform slicing, uniform slicing and MNA simulation methods shown on Table 4.5 and Table 4.6. The comparison indicates that our 3D-AADI simulator is more accurate than traditional ADI simulator and much more efficient than MAN.

chip 1		FDM-MNA	Uniform Geometry				Non-Uniform Geometry			
grid	Cnvg		itr#	time	MxErRt	SpUp	itr#	time	MxErRt	SpUp
4x4	0.1%	30	2	10	0.0078	3	2	10	0.0078	3
	0.01%		12	10	0.0056	3	12	10	0.0056	3
8x8	0.1%	60	1	10	0.0087	6	2	10	0.0084	6
	0.01%		16	30	0.0064	2	19	30	0.0058	2
16x16	0.1%	620	1	40	0.0090	15.5	5	80	0.0085	7.75
	0.01%		8	50	0.0082	12.4	15	100	0.0074	6.2
32x32	0.1%	6040	2	180	0.0090	33.56	7	300	0.0129	20.13
	0.01%		3	180	0.0090	33.56	22	460	0.0109	13.13
64x64	0.1%	98060	2	680	0.0091	144.21	8	1020	0.0246	96.14
	0.01%		2	680	0.0091	144.21	49	2630	0.0154	37.29
128x128	0.1%	805880	2	2770	0.0091	290.93	14	2940	0.0119	274.11
	0.01%		2	2770	0.0091	290.93	49	5430	0.0065	148.41

Table 4.1: The executing time results of FDM-MNA, uniform and non-uniform simulating algorithm on test chip 1.

chip 1		Uniform Geometry				Non-Uniform Geometry			
grid	Cnvg	Est.	Cnstr	LU	LUx	Est.	Cnstr	LU	LUx
4x4	0.1%	0	0	10	0	0	0	10	0
	0.01%	0	0	10	0	0	0	10	0
8x8	0.1%	0	0	0	10	0	0	10	0
	0.01%	10	0	10	10	0	10	10	10
16x16	0.1%	10	0	30	0	10	20	40	10
	0.01%	10	0	20	20	10	10	40	40
32x32	0.1%	40	10	100	30	50	40	110	100
	0.01%	30	10	90	50	50	40	100	270
64x64	0.1%	160	30	340	150	160	160	300	400
	0.01%	160	30	340	150	160	150	300	2020
128x128	0.1%	640	150	1340	640	570	700	530	1140
	0.01%	640	150	1340	640	570	700	530	3630

Table 4.2: The partial executing time results of FDM-MNA, uniform and non-uniform simulating algorithm on test chip 1.

chip 2		FDM-MNA	Uniform Geometry				Non-Uniform Geometry			
grid	Cnvg		itr#	time	MxErRt	SpUp	itr#	time	MxErRt	SpUp
4x4	0.1%	40	12	10	0.0033	4	14	10	0.0321	4
	0.01%		23	10	0.0008	4	24	10	0.0321	4
8x8	0.1%	80	27	10	0.0117	8	26	10	0.0149	8
	0.01%		57	20	0.0020	4	65	10	0.0131	8
16x16	0.1%	600	47	60	0.0299	10	42	40	0.0316	15
	0.01%		122	120	0.0045	5	119	70	0.0222	8.57
32x32	0.1%	5860	59	620	0.0847	9.45	51	290	0.0875	20.21
	0.01%		271	2740	0.0117	2.14	238	1060	0.0216	5.53
64x64	0.1%	75880	32	2000	0.1578	37.94	42	1240	0.0937	61.19
	0.01%		515	30370	0.0347	2.50	266	6010	0.0208	12.63
128x128	0.1%	570630	11	4050	0.1947	140.90	40	4870	0.0657	117.17
	0.01%		506	136160	0.1005	4.19	192	18500	0.0328	30.84

Table 4.3: The executing time results of FDM-MNA, uniform and non-uniform simulating algorithm on test chip 2.

chip 2		Uniform Geometry				Non-Uniform Geometry			
grid	Cnvg	Est.	Cnstr	LU	LUx	Est.	Cnstr	LU	LUx
4x4	0.1%	0	0	0	10	10	0	0	0
	0.01%	10	0	0	0	10	0	0	0
8x8	0.1%	0	0	0	10	0	10	0	0
	0.01%	0	0	10	10	0	0	0	10
16x16	0.1%	0	10	0	50	0	10	0	30
	0.01%	0	10	0	110	0	10	0	60
32x32	0.1%	10	10	30	570	10	30	30	220
	0.01%	0	10	30	2700	10	20	30	1000
64x64	0.1%	50	30	160	1760	40	130	110	960
	0.01%	30	30	150	30140	40	120	110	5740
128x128	0.1%	170	150	750	2980	170	570	430	3700
	0.01%	160	150	750	135100	170	570	420	17340

Table 4.4: The partial executing time results of FDM-MNA, uniform and non-uniform simulating algorithm on test chip 2.

	Max.	4x4 grids		8x8 grids		16x16 grids		32x32 grids	
Geometry	Error	Node #	iter #	Node #	iter #	Node #	iter #	Node #	iter #
Uniform	1%		1		1		1		1
Simulating	0.5%	128	54	512	41	2048	86	8192	251
Grids	0.1%		1698		5683		17588		60804
Non-Uniform	1%		1		1		3		36
Simulating	0.5%	128	54	464	30	1736	54	5192	230
Grids	0.1%		1698		4939		11574		46656

Table 4.5: The iteration results of uniform and non-uniform simulating geometry on test chip 1.

	Max.	4x4 grids		8x8 grids		16x16 grids		32x32 grids	
Geometry	Error	Node #	iter #	Node #	iter #	Node #	iter #	Node #	iter #
Uniform	1%		8		30		91		298
Simulating	0.5%	128	11	512	41	2048	124	8192	403
Grids	0.1%		21		78		226		719
Non-Uniform	1%		10		33		88		263
Simulating	0.5%	56	13	344	48	944	124	3128	367
Grids	0.1%		20		83		202		613

Table 4.6: The iteration results of uniform and non-uniform simulating geometry on test chip 2.

Chapter 5

Future Works and Applications

5.1 Enhancement of Suggesting Resolution

After finding the integer i such that average-heat-difference changing rate of $level_i$ is very close to that of $level_{i+1}$ on Eq.(3.1), we can also take chip-size, runtime, accuracy, and hardware resource into consideration to decide the simulating resolution on Eq.(5.1).

$$resolution = i + \alpha \cdot Size + \beta \cdot Time + \gamma \cdot Accuracy^{-1} + \delta \cdot Resource \tag{5.1}$$

where $Size$ representing the chip size, $Time$ representing the tolerance runtime for users, $Resource$ representing the resource of test machine, and $Accuracy^{-1}$ representing the tolerance accuracy are user-setting constants, and α, β, γ, δ are author's experimental parameters.

5.2 Local Refinement and Incremental Design

For temperature verification, Fig. 3.1 gives an overview. On the other hand, concerning local refinement and incremental-kernel application, 3D-AADI owns the feature of executing sectional "3D-AADI stamping" (a matrix-establishing process shown in Fig. 5.1) for the updating regions during thermal-aware design iterations. The enhancement, local refinement, and application, incremental kernel, are discussed as future works on section 5.2.

Since the 3D-AADI method we proposed owns the feature of updating only the affected

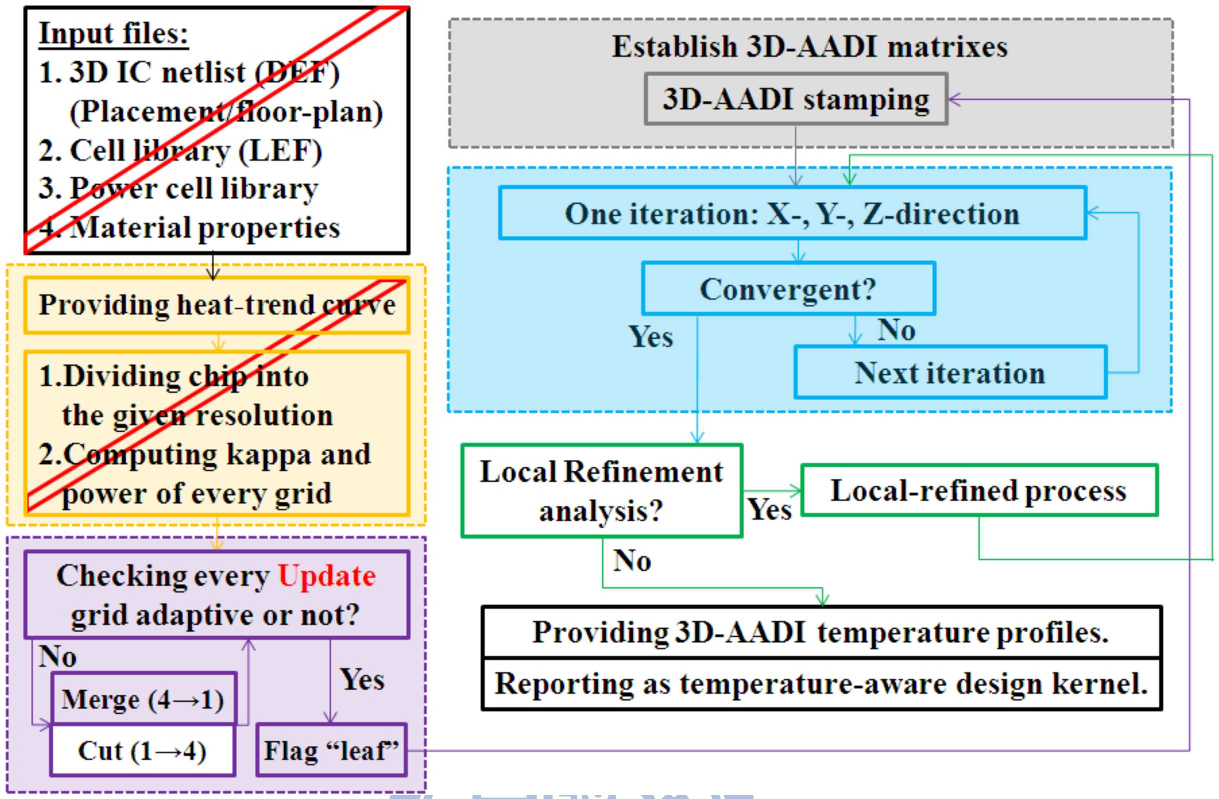
Input files:
1. 3D IC netlist (DEF) (Placement/floor-plan)
2. Cell library (LEF)
3. Power cell library
4. Material properties
Providing heat-trend curve
1.Dividing chip into the given resolution
2.Computing kappa and power of every grid
Checking every Update grid adaptive or not?
No
Merge (4→1)
Cut (1→4)
Yes
Flag "leaf"
Establish 3D-AADI matrixes
3D-AADI stamping
One iteration: X-, Y-, Z-direction
Convergent?
Yes
No
Next iteration
Local Refinement analysis?
Yes
Local-refined process
No
Providing 3D-AADI temperature profiles.
Reporting as temperature-aware design kernel.

Fig. 5.1: An overview of this work (which is in the background-color blocks) and 3D-AADI enhancement. Authors explain the left-middle orange block on section 3.1, the left-down purple block on section 3.2, the right-up gray block on section 3.3, and the right-middle blue block on section 3.4. Besides, the right-middle green sub-flow, which illustrates our 3D-AADI algorithm can be both enhanced in local-refined simulation and applied to thermal-driven kernel into the entire physical design flow, is discussed as future works and applications on section 5.2.

region of the changed grids by sectional "3D-AADI stamping" to the matrix $\mathbf{G}$ of Eq.(2.5), this simulator after being enhanced by local refinement and being applied by thermal kernel is both adaptive and incremental. Indeed, it is one of good solution to integrated into physical design flow as thermal-driven kernel due to not only that the node meshing can be non-uniform but also that the data structures of the updated sub-circuits are incremental. For these reasons, the thermal analysis solver can be both a simulator and thermal-aware design kernel. Fig. 5.2 illustrates the local adjustment of grids for satisfying the temperature gradient requirement on the affected region around the updated grids. Pseudo code of local refinement framework is derived in Fig. 5.3.

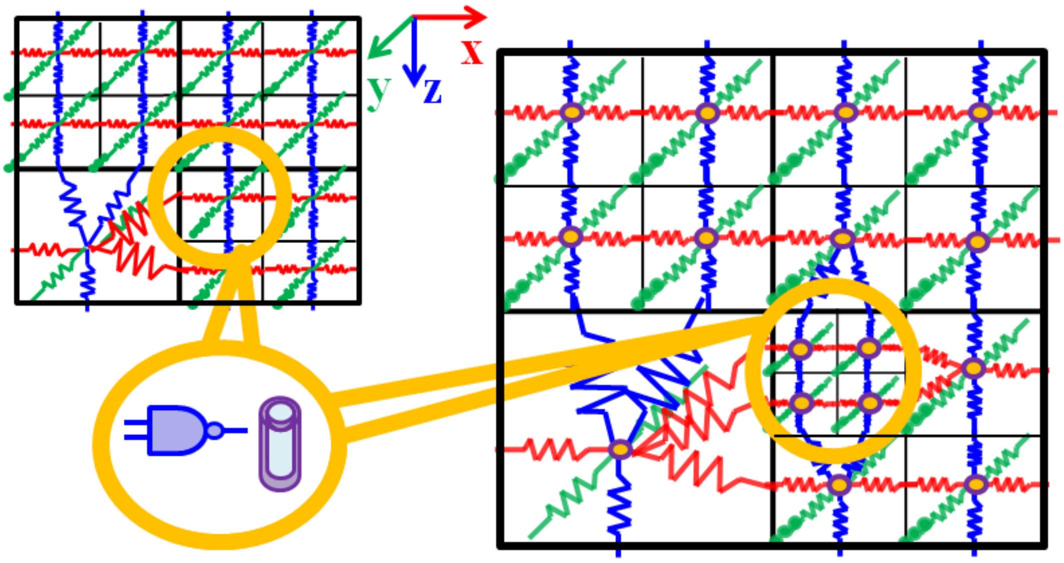

Fig. 5.2: Local refinement on the affected region around the updated grids for satisfying the requirement.

```
3D-AADI:Local Refinement Framework
Input:
   Affected field
Output:
   New refresh temperature of affected field

01  for Node $n$ = AffectedNodeCount - 1 : 0 do
02          Update and re-slice node of affected field.
03          Construct new node of affected field.
04          Connect neighbors of new node.
05          Update LUVector of affected field.
06          subADILUVectorSolver(UpdateLUVector).
08  end
07  Update temperature of affected field.
```

Fig. 5.3: *Local Refinement Framework.*

Chapter 6

Conclusions

3D ICs, which deal with cost-effective achievement by increasing the densities of interconnection between dies, are regarded as an attractive alternative solution for overcoming the bottlenecks on 2D planar ICs. In fact, 3D ICs offer the increased system a large number of advantages. However, one of critical challenges is heat dissipation due to higher accumulated power density and lower thermal conductivity of inter-layer dielectrics for vertical stacking layers of active tier. In this way, the management of thermal issues should be considered during physical design stages rather than only post-packaging verification on the future highly integrated systems. For these reasons, we develop an adaptive thermal simulator applying our 3D-AADI algorithm according to ADI method to provide temperature distribution during 3D IC physical design flow from floor-plan to verification.

The simulator constructs adaptive size of simulation grids to avoid the limited simulating performance of the most critical position. Furthermore, we apply the concept of ADI iteration method to non-uniform nodes. Eventually, the 3D-AADI tool can be regard as both a reliable thermal simulator and a thermal-driven kernel on 3D IC design flow. The simulator we developed is both adaptive and incremental.

We proposed a thermal simulator on 3D IC applying our 3D-AADI algorithm according to ADI concept to deal with temperature distribution during 3D IC physical design flow from floor-plan to verification. In order to avoid the performance limitation of the most critical position such as hot spots and (thermal or signal) TSVs, this simulator utilizes adaptive analysis size of grids. Furthermore, we develop 3D-AADI algorithm according to the concept of ADI iteration method and analysis the non-uniform meshing calculation and convergence. Finally, the simulator we developed is both adaptive and incremental, because the 3D-AADI tool can be regard as both a reliable thermal simulator and a thermal-driven kernel on 3D IC design flow from floor-plan to verification.

Bibliography

[1] P.D. Franzon, W.R. Davis, M.B. Steer, S. Lipa, E.C. Oh, T. Thorolfsson, S. Melamed, S. Luniya, T. Doxsee, S. Berkeley, et al. Design and CAD for 3D integrated circuits. In *Proceedings of the 45th annual Design Automation Conference*, pages 668–673. ACM, 2008.

[2] B. Black, DW Nelson, C. Webb, and N. Samra. 3D processing technology and its impact on iA32 microprocessors. In *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings*, pages 316–318, 2004.

[3] PD Franzon, WR Davis, MB Steer, H. Hao, S. Lipa, S. Luniya, C. Mineo, J. Oh, A. Sule, and T. Thorolfsson. Design for 3D Integration and Applications. In *Signals, Systems and Electronics, 2007. ISSSE'07. International Symposium on*, pages 263–266, 2007.

[4] S. Im and K. Banerjee. Full chip thermal analysis of planar (2-D) and vertically integrated (3-D) high performance ICs. In *International Electron Devices Meeting*, pages 727–730. IEEE; 1998, 2000.

[5] L. Jiang, S. Kolluri, B.J. Rubin, H. Smith, E.G. Colgan, M.R. Scheuermann, J.A. Wakil, A. Deutsch, and J. Gill. Thermal modeling of on-chip interconnects and 3D packaging using EM tools. *2008 IEEE-EPEP Electrical Performance of Electronic Packaging*, pages 279–282, 2008.

[6] Chang-Tzu Lin, Ding-Ming Kwai, Yung-Fa Chou, Ting-Sheng Chen, and Wen-Ching Wu. Cad reference flow for 3d via-last integrated circuits. In *Proceedings of the 2010 conference on Asia South Pacific design automation*, pages 187–192. Citeseer, 2010.

[7] J.L. Tsai, C.C.P. Chen, G. Chen, B. Goplen, H. Qian, Y. Zhan, S.M. Kang, MDF Wong, and SS Sapatnekar. Temperature-aware placement for SOCs. *Proceedings of the IEEE*, 94(8):1502–1518, 2006.

[8] Y. Zhan and SS Sapatnekar. High-efficiency green function-based thermal simulation algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(9):1661–1675, 2007.

[9] Pei-Yu Huang and Yu-Min Lee. Full-chip thermal analysis for the early design stage via generalized integral transforms. *IEEE Transactions on Very Large Scale Integration Systems*, 17(5):613–626, May 2009.

[10] T.Y. Wang and C.C.P. Chen. SPICE-compatible thermal simulation with lumped circuit modeling for thermal reliability analysis based on modeling order reduction. In *Proceedings of the 5th International Symposium on Quality Electronic Design*, pages 357–362. Citeseer, 2004.

[11] P. Li, LT Pileggi, M. Asheghi, and R. Chandra. IC thermal simulation and modeling via efficient multigrid-based approaches. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(9):1763–1776, 2006.

[12] T.Y. Wang, Y.M. Lee, and C.C.P. Chen. 3D thermal-ADI: an efficient chip-level transient thermal simulator. In *Proceedings of the 2003 international symposium on Physical design*, pages 10–17. ACM, 2003.

[13] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and MR 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.

[14] W. Huang, K. Skadron, S. Gurumurthi, R.J. Ribando, and M.R. Stan. Differentiating the roles of ir measurement and simulation for power and temperature-aware design. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pages 1–10. IEEE, 2009.

[15] P. Wilkerson, A. Raman, and M. Turowski. Fast, automated thermal simulation of three-dimensional integrated circuits. In *Thermal and Thermomechanical Phenomena in Electronic Systems*, pages 706–713, 2004.

[16] Y. Yang, Z. Gu, C. Zhu, RP Dick, and L. Shang. ISAC: Integrated space-and-time-adaptive chip-package thermal analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(1):86–99, 2007.

[17] J. Cong, G. Luo, J. Wei, and Y. Zhang. Thermal-aware 3d ic placement via transformation. In *Proceedings of the 2007 conference on Asia South Pacific design automation*, pages 780–785. Citeseer, 2007.

[18] W. Huang, M.R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam. Compact thermal modeling for temperature-aware design. In *Proceedings of the 41st annual Design Automation Conference*, pages 878–883. ACM, 2004.

[19] M.N. Ozisik. *Boundary Value Problems of Heat Conduction*. Dover, New York, 1968.

[20] D.W. Peaceman and H.H. Rachford Jr. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, pages 28–41, 1955.

[21] J. Douglas and J.E. Gunn. A general formulation of alternating direction methods. *Numerische Mathematik*, 6(1):428–453, 1964.

[22] Y. Saad and Y. Saad. *Iterative methods for sparse linear systems*. PWS Pub. Co., 1996.