

UNIVERSITY OF CALIFORNIA

Los Angeles

**Placement and Design Planning  
for 3D Integrated Circuits**

A dissertation submitted in partial satisfaction  
of the requirements for the degree  
Doctor of Philosophy in Computer Science


by

**Guojie Luo**

2011

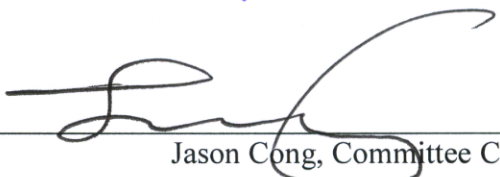
© Copyright by  
Guojie Luo  
2011

The dissertation of Guojie Luo is approved.

  
M.C. Frank Chang

  
Milos Ercegovac

  
Miodrag Potkonjak

  
Jason Cong, Committee Chair

University of California, Los Angeles

2011

To my family...

## TABLE OF CONTENTS

Chapter 1	Introduction.....	1
1.1	Background .....	1
1.2	Overview of the Dissertation.....	8
Chapter 2	Problem Formulation and Related Work .....	12
2.1	Introduction .....	12
2.2	Preliminaries.....	13
2.3	Previous Work.....	15
2.4	Problem Statement .....	18
2.4.1	Wirelength Objective Function .....	18
2.4.2	Non-Overlapping Constraints .....	20
2.4.3	Timing Constraints .....	22
2.4.4	Thermal Constraints .....	22
2.4.5	Formulation of the 3D Placement Problem .....	23
Chapter 3	3D-Craft: A 3D Physical Design Flow Based on OpenAccess .....	25
3.1	Introduction .....	25
3.2	Overview of 3D-Craft and OpenAccess Extension for 3D Design .....	26
3.2.1	Preliminaries on OpenAccess .....	27
3.2.2	Issues in Extending OpenAccess for 3D Physical Design .....	29
3.2.3	Solutions for Extending OpenAccess for 3D Designs .....	29
3.3	Components in 3D-Craft .....	31
3.3.1	Resistive Thermal Model .....	31
3.3.2	Thermal-Aware 3D Floorplanning .....	33
3.3.3	Multilevel Analytical 3D Placement (mPL-3D).....	34

3.3.4	TSV Insertion and Planning.....	35
3.3.5	3D Detailed Routing.....	35
3.4	Conclusions and Future Work.....	36
Chapter 4	Thermal-Aware 3D Placement via Transformation .....	37
4.1	Transformation-Based 3D Placement Framework.....	38
4.2	Transformations for 3D Placement .....	39
4.2.1	Local Stacking Transformation .....	39
4.2.2	Transformation through Folding .....	42
4.2.3	Window-Based Stacking/Folding.....	43
4.3	Refinement: RCN Graph-Based Die Reassignment .....	45
4.3.1	Conflict-Net Graph.....	45
4.3.2	Relaxed Non-Overlap Constraint .....	47
4.4	Experimental Results .....	48
4.5	Conclusions .....	51
Chapter 5	Efficient Gradient Computation for Density-Constrained Analytical Placement Approaches.....	52
5.1	Problem Formulation .....	55
5.2	General Smoothed Density .....	57
5.2.1	Helmholtz Smoothing .....	57
5.2.2	Poisson Smoothing .....	57
5.2.3	Gaussian Smoothing.....	58
5.2.4	General Global Smoothing .....	58
5.3	Density Penalty Function .....	59
5.3.1	Quadratic Penalty Method .....	59
5.3.2	Augmented Lagrangian Method.....	61
5.4	Efficient Gradient Computation.....	62

5.4.1	Naive Computation.....	63
5.4.2	Efficient Computation.....	64
5.5	Practical Implementation in a Global Placer.....	69
5.5.1	Density Grids and Smoothed Density.....	70
5.5.2	Step Size Selection.....	70
5.5.3	Penalty Factor Update Scheme.....	71
5.5.4	Lagrangian Multiplier Update Scheme.....	72
5.5.5	Stopping Criterion.....	72
5.6	Experimental Results.....	72
5.7	Conclusions.....	76
Chapter 6	Multilevel Analytical Placement for 3D ICs.....	77
6.1	Continuous Relaxation for Die Assignment.....	78
6.2	3D Global Placement with Bell-Shaped Area Projection.....	79
6.3	3D Global Placement with Huber-Based Smoothing.....	85
6.3.1	3D Area Density Constraints.....	86
6.3.2	Local Smoothing of the Overlapping Function.....	87
6.3.3	Density Penalty Function and Global Smoothing.....	91
6.4	Mixed-Size 3D Placement Flow.....	94
6.4.1	3D Floorplan-Based Initial Solution.....	95
6.4.2	Analytical Solver with Multiple-Stepsize Scheme.....	96
6.5	Multilevel Scheme.....	102
6.6	Experimental Results.....	103
6.6.1	Results on Standard-Cell 3D Placement with Bell-Shaped Area Projection.....	103
6.6.2	Results on Standard-Cell 3D Placement with Huber-Based Smoothing .....	107

6.6.3	Results on Mixed-Size 3D Placement.....	111
6.7	Conclusions .....	114
Chapter 7	Thermal-Aware Cell and TSV Co-Placement for 3D ICs .....	116
7.1	Introduction .....	116
7.2	Motivation .....	118
7.3	Properties of a Thermally-Optimal TSV Distribution.....	120
7.4	Thermal-Aware 3D Placement.....	128
7.4.1	Thermal-Aware Cell/TSV Co-Placement .....	128
7.4.2	Overall Thermal-Aware Placement Flow .....	130
7.5	Experimental Results .....	131
7.6	Conclusions .....	135
Chapter 8	Case Study: 3D Physical Design of LEON3 Microprocessor .....	136
8.1	Standard-Cell Implementation .....	136
8.2	Mixed-Size Implementation and Physical Hierarchy Exploration .....	139
8.3	Mixed-Size Implementation and Timing Characteristics.....	144
Chapter 9	Conclusions and Future Work .....	147
Appendix:	Manual of 3D-Craft .....	150
1)	Setting Up the Standard Cell Library and the TSV Library .....	150
2)	Setting Up the 3D Design Library .....	150
3)	Running the 3D Floorplanner or the Real 3D Placer.....	151
4)	TSV Insertion, Net Splitting and Chip Resizing .....	152
5)	TSV Planning and Legalization .....	153
6)	3D Design Export for Commercial EDA tools .....	153
References.....		155



## LIST OF FIGURES

Figure 1. Two examples of 3D ICs in a cross-section view .....	2
Figure 2. Minimum stacking yields at different area overhead ratios .....	6
Figure 3. A typical adhoc 3D physical design flow [42] .....	13
Figure 4. (a) Area constraint is satisfied; (b) Area constraint is not satisfied .....	21
Figure 5. Overview of 3D-Craft .....	27
Figure 6. An example of the hierarchical design database [101] .....	28
Figure 7. An example of the <i>oaAppDef</i> mechanism .....	29
Figure 8. Resistive thermal model .....	32
Figure 9. Placement transformation framework .....	38
Figure 10. Local stacking transformation .....	40
Figure 11. Two folding-based transformation schemes .....	43
Figure 12. 2×2 windows with different die assignments .....	45
Figure 13. Relaxed conflict-net graph .....	47
Figure 14. Relaxation of non-overlap constraint .....	47
Figure 15. 3D analytical placement flow .....	79
Figure 16. An example of the density function .....	80
Figure 17. Bell-shaped density projection function .....	82
Figure 18. The overlapping function, the bell-shaped approximation, and the Huber-based approximation .....	91
Figure 19. Our mixed-size 3D placement flow .....	94

Figure 20. Global placement results with difficulties in legalization .....	96
Figure 21. Global placement results guided by 3D floorplanning .....	96
Figure 22. A mixed-size linear placement example.....	98
Figure 23. The corresponding uniform-size linear placement.....	98
Figure 24. Tradeoff curves for ibm01 .....	106
Figure 25. Tradeoff curves for ibm09.....	106
Figure 26. Comparisons of various analytical 3D formulations .....	109
Figure 27. Uniform power with clustered TSVs vs. consistent TSV and power distribution .....	120
Figure 28. A two-die example of the thermal resistive network .....	122
Figure 29. A two-die example to illustrate Theorem 7.2.....	128
Figure 30. Comparison of thermal optimizations on wb_conmax .....	134
Figure 31. LEON3 processor core block diagram [94].....	136
Figure 32. The 3D placement by mPL-3D, the TSV distribution, and the routing congestion analysis by Cadence Encounter .....	139
Figure 33. Placement of logical units.....	141
Figure 34. Plot of the per-unit area consumption.....	142
Figure 35. Restricted placement for some logical modules.....	144

## LIST OF TABLES

Table 1. Industrial practices of TSV technologies .....	7
Table 2. Benchmark characteristics and results of mPL6 and LST transformations.....	49
Table 3. Results of folding and window-based transformations .....	49
Table 4. Thermal-aware T3Place results .....	50
Table 5. Comparisons with existing 3D placement in [43] .....	50
Table 6. Benchmark statistics.....	73
Table 7. Experimental results on IBM-HB+ benchmark (mPL6 & SCAMPI).....	74
Table 8. Experimental results on IBM-HB+ benchmarks (our methods).....	74
Table 9. Experimental results on the modified ISPD'05 and ISPD'06 benchmarks .....	75
Table 10. Experimental results for the multilevel analytical 3D placement method with $\alpha_{TSV}=10$ .....	104
Table 11. Experimental results with 3D global smoothing of 1-level placement (flat)...	108
Table 12. Experimental results with 3D global smoothing of 2-level placement.....	108
Table 13. Experimental results with 3D global smoothing of 5-level placement .....	108
Table 14. Experimental results with 2D global smoothing with $\alpha_{TSV} = 100$ .....	109
Table 15. Comparisons of the analytical 3D placement algorithms .....	110
Table 16. Statistics of the benchmarks .....	112
Table 17. Effect of the multiple-stepsize scheme .....	113
Table 18. 3D placement results .....	113
Table 19. Major notations.....	121

Table 20. Circuit statistics .....	132
Table 21. Results of our co-placement method and comparisons with other methods....	134
Table 22. Statistics of the 2D and 3D implementations .....	138
Table 23. Overall statistics of the synthesized netlist .....	140
Table 24. Statistics of the per-unit area consumption.....	142
Table 25. HPWL and TSV comparisons for different placements .....	144
Table 26. Four implementations of the PE unit.....	145
Table 27. 3D Implementations of a 4-core LEON3 .....	146
Table 28. Reported 3D-aware EDA tools.....	148

## ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Prof. Jason Cong. I would not be able to complete this thesis without his advice and support through these years of my graduate study. In addition to providing me the necessary trainings towards a qualified PhD, he also taught me lessons during personal communications and group events to demonstrate the practices towards an excellent researcher and educator.

I am grateful to my doctoral committee members, Prof. Frank Chang, Prof. Milos Ercegovac, and Prof. Miodrag Potkonjak for their valuable comments and suggestions of this thesis. I also appreciate the opportunity to assist Prof. Ercegovac in teaching the class on Logic Design of Digital Systems, where I gained trainings and experiences on teaching.

I would like to thank Prof. Lieven Vandenberghe, Prof. Tony Chan, and my colleagues Yan Zhang, Kenton Sze, Min Xie, John Lee, Eric Radke, Kelly Tsota, and Bingjun Xiao for participating in the discussions during the placement group meeting. Yan, Kenton, and Min gave me very helpful tutorials to the physical design automation problems. John and Eric were my closest colleagues through all these years in my graduate study, both of whom helped me a lot for the technical details of optimization methods. Kelly and Bingjun were excellent collaborators in our participation of the ISPD'11 routability-driven placement contest, where we won the second place among all the teams over the world.

I wish to thank Dr. Jeonghee Shin and Dr. John Darringer at IBM T.J. Watson Research Center, where I spent three years as a part-time research intern. It was a nice experience to work with the researchers with the vision and experience of industrial practices.

I want to thank my colleague Dr. Yiyu Shi at the Missouri University of Science and Technology for the collaboration on the thermal-aware TSV/cell co-placement problem discussed in this thesis (Chapter 7). I am also grateful for the inspiring discussions with him on the 3D integration-related problems.

I would also like to thank all the other colleagues that I have worked with in the past few years: Kirill Minkovich, Yi Zou, Bin Liu, Karthik Gururaj, Chunyue Liu, Bo Yuan, Hui Huang, YuHui Huang, Yu-Ting Chen, Muhuan Huang, Sen Li, and Peng Zhang. I enjoyed the days that I spent with them during research discussions and social events.

I would like to thank Janice Martin-Wheeler, Alexandra Luong, Terry Valai, Verra Morgan, and Steve Arbuckle for their help in the administrative support that makes my graduate study smooth. Particularly, I really appreciate Janice for her patience and helpful comments in proofreading all the drafts of my publications and this thesis.

Finally, I express my truly appreciation to my parents Zhikun Luo and Shaobing Liao, and my girlfriend Xiaoming Zhu. Without their love, encourage and moral support, I could not have accomplished what I have achieved so far.

The research activities involved in this thesis are partially supported by National Science Foundation (NSF) under CCF-0430077 and CCF-0528583, Semiconductor Research Corporation (SRC) under task 1460.001, Gigascale Silicon Research Center

(GSRC) under task 2049.002, and International Business Machine (IBM) under a Defense Advanced Research Projects Agency (DARPA) subcontract.

## VITA

January 1983	Born, Foshan, Guangdong, China
2001-2005	B.S. Computer Science Department, Peking University, Beijing, China
2005-2008	M.S. Computer Science Department, University of California, Los Angeles, USA
2008-Present	Ph.D. Computer Science Department, University of California, Los Angeles, USA

## PUBLICATIONS

Jason Cong, Guojie Luo, Jie Wei, and Yan Zhang, “Thermal-aware 3D IC Placement via Transformation,” *Proceedings of the 12<sup>th</sup> Asia and South Pacific Design Automation Conference (ASP-DAC)*, Yokohama, Japan, pp. 780-785, January 2007.

Jason Cong, and Guojie Luo, “Highly Efficient Gradient Computation for Density Constrained Analytical Placement Problems,” *Proceedings of the International Symposium on Physical Design (ISPD)*, Portland, OR, April 2008.

Jason Cong, Chunyue Liu, Guojie Luo, Quantitative Studies of Impact of 3D IC Design on Repeater Usage, *Proceedings of the 25<sup>th</sup> International VLSI/ULSI Multilevel Interconnection Conference (VMIC)*, Fremont, CA, October 2008.



Jason Cong, Guojie Luo, and Eric Radke, "Highly Efficient Gradient Computation for Density-Constrained Analytical Placement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, Volume 27, Number 12, pp. 2133-2144, December 2008.

Jason Cong, and Guojie Luo, "A Multilevel Analytical Placement for 3D ICs," *Proceedings of the 14<sup>th</sup> Asia and South Pacific Design Automation Conference (ASP-DAC)*, Yokohama, Japan.

Jason Cong and Guojie Luo, "A 3D Physical Design Flow Based on OpenAccess", *International Conference on Communications, Circuits and Systems (ICCCAS)*, San Jose California, pp. 1103-1107, July 2009. (Invited Paper)

Jason Cong and Guojie Luo, "Advances and Challenges in 3D Physical Design", *IPSJ Transactions on System LSI Design Methodology (TSLDM)*, Volume 3, pp. 2-18, February 2010. (Invited paper)

Jason Cong and Guojie Luo, "Analytical 3D Placement for Mixed-Size Circuits", *Proceedings of the International Symposium on Physical Design (ISPD)*, San Francisco, CA, pp. 61-66, March 2010.

Thorlindur Thorolfsson, Guojie Luo, Jason Cong and Paul D. Franzon, "Logic-on-Logic 3D Integration and Placement", *Proceedings of the 2nd IEEE International 3D System Integration Conference (3DIC)*, Munich, Germany, November 2010.

Jason Cong and Guojie Luo, "Chapter 5: Thermal-Aware 3D Placement," *Three Dimensional System Integration: IC Stacking Process and Design*, Springer Publishers 2010.

Jeonghee Shin, John A. Darringer, Guojie Luo, Alan J. Weger, and Charles L. Johnson, "Early Chip Planning Cockpit", *Design, Automation and Test in Europe (DATE)*, Grenoble, France, March, 2011.

Jason Cong, Guojie Luo and Yiyu Shi, "Thermal-Aware Cell and Through-Silicon-Via Co-Placement for 3D ICs," *Proceedings of the 48th Annual Design Automation Conference (DAC)*, San Diego, CA, June 2011.

Jason Cong and Guojie Luo, "Chapter 5: 3D Physical Design", *Three Dimensional System Integration: IC Stacking Process and Design*, Springer Publishers, 2011.

## ABSTRACT OF THE DISSERTATION

# **Placement and Design Planning for 3D Integrated Circuits**

by

**Guojie Luo**

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2011

Professor Jason Cong, Chair

3D integration enables an additional dimension of freedom for processor design. It enhances device packaging density and shortens the length of global interconnects; it thus benefits functionality, performance and power of 3D *integrated circuits* (ICs).

A major portion of these advantages comes from the reduction of interconnect complexity in 3D integration, when the interconnect delay plays an important part in 2D performance barriers. In order to exploit these new features that do not exist in 2D circuits, novel techniques must be developed in physical design, including the stages of floorplanning and placement. The die assignment of standard cells and IP macros, signal/thermal *through-silicon via* (TSV) planning, thermal-awareness and traditional wirelength metric are the challenges that need to be handled in the 3D physical design. In this dissertation, we focus on three aspects of the placement algorithms for 3D physical

design: (i) gradient computation of density penalty functions for analytical placement; (ii) analytical 3D placement algorithms and the support of mixed-size designs; (iii) thermal-aware cell and TSV co-placement for 3D designs.

As an early attempt, we presented transformation-based 3D placement approaches. These had already demonstrated that a 4-die 3D implementation can reduce the wirelength by as much as 50% compared to a 2D implementation by taking advantage of a significant amount of TSVs to shorten wirelength.

Later on, we presented a 3D placement framework, which consists of a 3D floorplanning step and a 3D placement step. The 3D placement step is the focus of this dissertation. It supports a pseudo-3D placement mode (die assignment is fixed) and a real-3D placement mode (die assignment is variable). The 3D floorplanning step is performed on a coarsened netlist; it is also used to determine the die assignment for the pseudo-3D placement mode and is used as an initial solution for the real-3D placement mode. Particularly, (i) we presented an efficient gradient computation of density penalty functions which are applicable to both 2D and 3D density-constrained analytical placement algorithms; (ii) we discussed the formulation of density constraints to capture the area distribution in a 3D placement region with mixed-size support, and presented the bell-shaped area projection approach and the Huber-based local smoothing approach; (iii) we presented a TSV/cell co-placement feature in the pseudo-3D placement mode to relieve the thermal issues at the placement stage.

The gradient computation of density penalty functions is a general technique applicable to both 2D and 3D density-constrained analytical placement algorithms. This

technique reduces the runtime for gradient computation by a factor of  $n$  compared to a naïve approach, where  $n$  is the problem size. It enables an accurate and efficient gradient computation, while state-of-the-art analytical algorithms rely on approximations. The experimental results show that it reduces 15% of the wirelength on the benchmarks with large area variations.

Based on the efficient gradient computation method, we were able to formulate the 3D area density constraints with a bell-shaped area projection approach and a Huber-based smoothing approach to measure overlaps. These approaches achieve 30% shorter wirelength with slightly fewer TSVs than the transformation approach with the best wirelength; these approaches also achieve 30% fewer TSVs with a 10% shorter wirelength than the transformation approach with fewest TSVs. The mixed-size designs are supported with a multi-stepsizes scheme in the analytical solver; this reduces the wirelength by 27% on average compared to 2D mixed-size implementations.

The thermal issues in 3D ICs are challenging due to the stacked structure. Fortunately, the TSVs provide heat flow paths with much greater thermal conductivity than the dielectric layers. We derived a metric to evaluate the TSV and power distribution efficiently as an indirect evaluation of temperature. This metric is integrated into the pseudo-3D placement mode, and is able to reduce the temperature by 30 °C with only 5% degradation in wirelength.

These algorithms are integrated in the 3D placement stage of a 3D physical design flow, named 3D-Craft and developed at UCLA. The application of 3D-Craft on an open-source microprocessor is demonstrated.

# Chapter 1

## Introduction

### 1.1 Background

3D integration promises to further increase integration density, beyond Moore's Law, and offers the potential to significantly reduce interconnect delays and improve system performance [10]. Furthermore, the shortened wirelength, especially that of the clock net, also lessens the power consumption of circuits. 3D integration also provides a flexible way to carry out the heterogeneous system-on-chip (SoC) design by integrating disparate technologies, such as memory and logic circuits, radio frequency (RF) and mixed signal components, optoelectronic devices, etc., onto different dies of a 3D *integrated circuit* (IC).

Figure 1 shows two examples of 3-die 3D ICs in a cross-section view. The bottom die, the middle die and the top die are labeled 1, 2, and 3, respectively. The physical layers are parallel to the  $(x, y)$  plane, and are bonded along the  $z$ -direction, where the darker shaded bands are dielectric layers, the lighter shaded bands are silicon layers, and the white bands are metal layers. The large rectangles vertically drilling through silicon layers represent *through-silicon vias* (TSVs), which connect logic gates on different silicon layers. The I/O ports open above the topmost layer. Figure 1(a) presents a 3D IC by bonding three dies in a back-to-face order, where the back side (the silicon layer) of the upper-level die is bonded to the front side (the topmost metal layer) of the lower-level

die. Figure 1(b) presents another 3D IC, where the middle die is bonded face-to-face to the bottom die, and the top die is bonded face-to-back to the middle die.

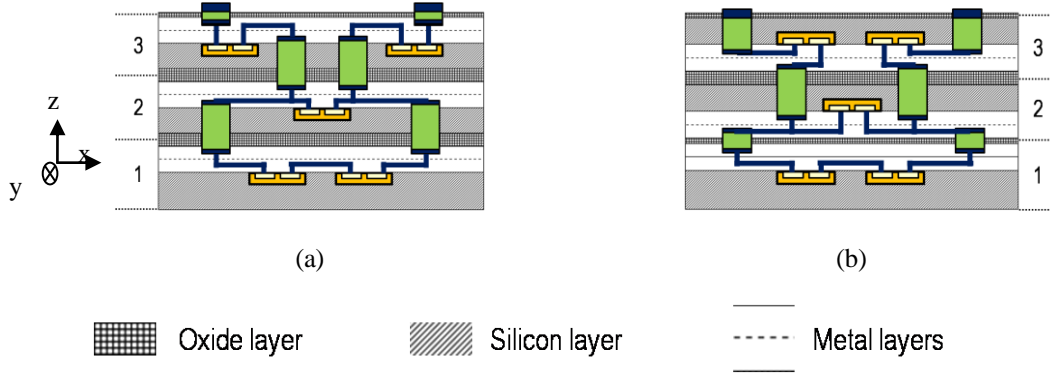


Figure 1. Two examples of 3D ICs in a cross-section view

In a general sense, 3D integration technologies can be classified into three categories [18]: (i) 3D packaging, (ii) 3D TSV-based integration (top-down, parallel integration), and (iii) 3D monolithic integration (bottom-up, sequential integration). The 3D packaging techniques, including the wire-bonded die-stack technique and the Ball-Grid-Array (BGA)-stack technique, have been widely used in many consumer products, especially the handheld devices. On the other end, 3D monolithic integration, which builds active semiconductor device layers sequentially on a single wafer, is still at an early R&D stage. The 3D TSV-based integration fabricates individual wafers in parallel, and stacks the wafers or dies with TSVs as die-to-die interconnections. Recently this technology has attracted significant research and development efforts from both industry and academia. The 3D TSV-based integration is the focus of this dissertation.

To form the interconnections across different dies, the TSVs are etched through the silicon layer with deep reactive-ion etching, insulated with thermal oxide, and then filled

with liner and conductor [58]. The TSVs are open after wafer thinning, and the bottom of the wafer is insulated; the TSV contacts are made by backside metallization. Depending on when the TSVs are fabricated, the TSV formation has various versions\* of (i) via-first, (ii) via-middle, (iii) via-last, and (iv) via-after [85]. The via-first process fabricates TSVs before building transistors and metal layers on a wafer. The via-middle process fabricates TSVs after building transistors, but before building metal layers. The via-last process fabricates TSVs after building transistors and metal layers. The via-after process fabricates TSVs after the bonding to another device wafer.

The dies in a 3D IC may be bonded in the order of (i) face-to-back or (ii) face-to-face [18]. In a single die, the bottom of the silicon substrate is called the “back” of this die, and the top of the metal layers is called the “face” of this die. In the face-to-back bonding, the face of a lower die is bonded to the back of the upper die. In the face-to-face bonding, the face of the lower die is bonded to the face of the upper die. In the via-first, via-middle and via-last processes, the TSVs of the upper dies (wafers) are fabricated before bonding; in the via-after process, the TSVs are fabricated after bonding.

These bonding orders are applicable to the process options including (i) wafer-to-wafer, (ii) die-to-wafer, and (iii) die-to-die stacking [58][18]. The wafer-to-wafer process has the lowest operating cost among the three, but it has limitations on the overall yield and chip size, except the cases where we stack extremely high-yield wafers, or stack one cheap, high-yield wafer onto an expensive, low-yield wafer [85]. The die-to-wafer and die-to-die processes are flexible for Known-Good-Die (KGD) process to improve yield.

---

\* Please note that the definitions of these terminologies are not consistent among different organizations at this point.



They are suitable for heterogeneous integration, but they are costly in handling and bonding.

Compared to 2D integration, the stacking processes (TSV formation, wafer thinning, die/wafer bonding, etc.) certainly introduce yield loss. However, as mentioned earlier, the die-to-wafer and die-to-die processes are able to improve yield with KGDs. In a simplified homogenous stacking model [63], the functional yield of 2D integration and 3D integration (with KGDs) can be expressed as

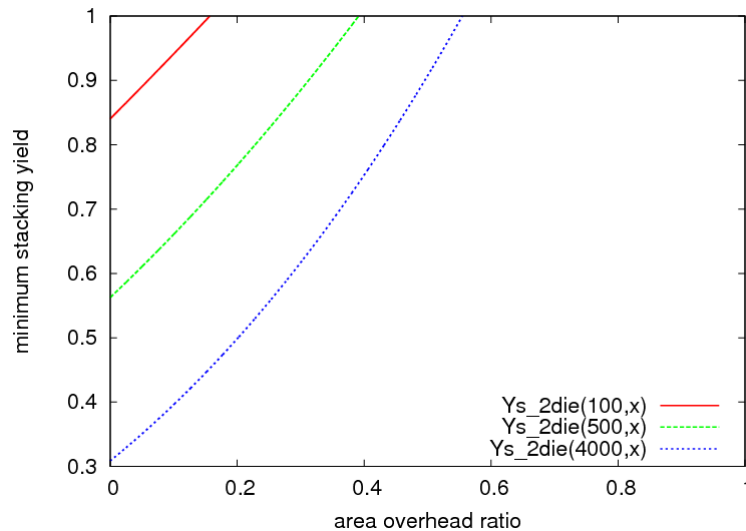
$$\begin{aligned}
 Y_{2D} &= \left(1 + \frac{D_0}{\alpha} \cdot A\right)^{-\alpha} \\
 Y_{3D} &= \left(1 + \frac{D_0}{\alpha} \cdot \frac{(1+x)A}{K}\right)^{-\alpha} Y_S^{K-1}
 \end{aligned} \tag{1}$$

where  $D_0$  is the density of point-defects per unit area (typically ranging from 0.001/mm<sup>2</sup> to 0.006/mm<sup>2</sup>),  $\alpha$  is a parameter to model the non-uniformly distributed defects (typically ranging from 1.0 to 5.0),  $A$  is the die area in the baseline 2D implementation,  $K$  is the number of dies in the 3D implementation,  $x$  is the ratio of the area overhead, and  $Y_S$  is the geometric mean of the yield per stacking step.

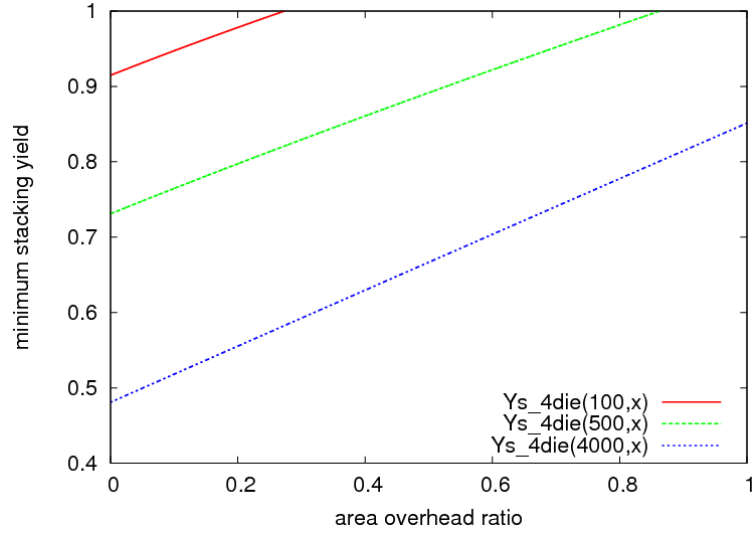
In order to produce the same amount of chips with the same volume of wafers using 3D integration, there is a minimum requirement that  $Y_{3D}/(1+x) \geq Y_{2D}$ , so the stacking yield has to satisfy that

$$Y_S \geq (1+x)^{\frac{1}{K-1}} \left( \frac{\alpha + D_0 \cdot A \cdot (1+x)/K}{\alpha + D_0 \cdot A} \right)^{\frac{\alpha}{K-1}} \tag{2}$$

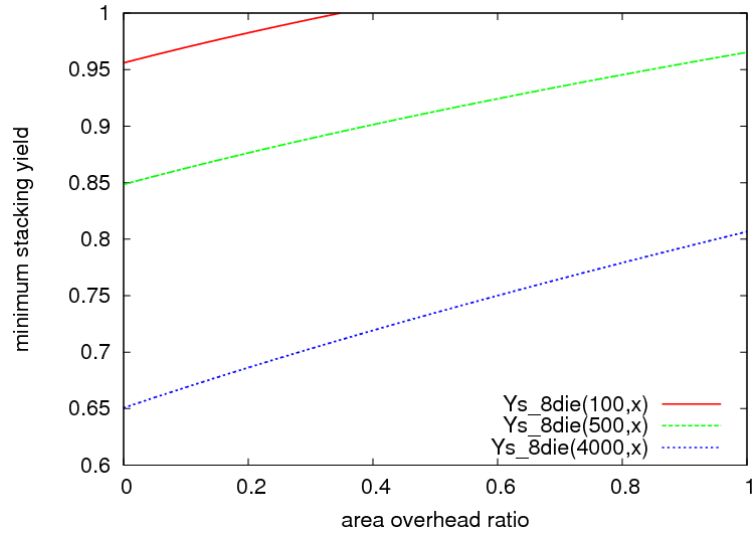
This expression gives us a relation between the area overhead  $x$  and the stacking yield  $Y_s$ . The area overhead comes from the area consumed by TSVs and the pre-bond TSV testing circuits [21]. In Figure 2, we set  $D_0$  to  $0.004/\text{mm}^2$  and set  $\alpha$  to 2.0 as in [21]. The curves show the minimum requirements on the stacking yield for 3D implementations with 2, 4, and 8 dies in Figure 2 (a), (b), and (c), respectively. The baseline die areas include  $100 \text{ mm}^2$ ,  $500 \text{ mm}^2$ , and  $4000 \text{ mm}^2$ . The area of  $100 \text{ mm}^2$  is selected to represent the processors for mobile computing (e.g., Apple A5 with a die area of  $122 \text{ mm}^2$  [11]); the area of  $500 \text{ mm}^2$  is selected to represent the processors for high-performance computing (e.g., IBM POWER7 with a die area of  $567 \text{ mm}^2$  [86]); and the area of  $4000 \text{ mm}^2$  is made up for highly-integrated system-on-chips.



(a) Required stacking yield for 2-die chips



(b) Required stacking yield for 4-die chips



(c) Required stacking yield for 8-die chips

Figure 2. Minimum stacking yields at different area overhead ratios

In Figure 2, the yield of 3D integration is greater than that of 2D integration if the stacking yield is greater than the minimum stacking yield. The curves show that when implementing a larger 3D design, the stacking yield required becomes lower, and the area overhead allowed becomes greater. The discussions above focus on the functional yield.

Please refer to [41] for the techniques to improve the parametric yield of 3D integration, and refer to [38][21] for the cost-oriented tradeoff analysis.

Due to the availability of TSV technologies, 3D integration is gaining momentum. The industrial practices of TSV technologies are listed in Table 1. The data show that the technologies with TSV pitches of tens of microns are mature and have been transferred to production. The technologies with TSV pitches of several microns are promising for production in the near future.

Table 1. Industrial practices of TSV technologies

Company / Organization	TSV diameter ( $\mu\text{m}$ )	TSV pitch ( $\mu\text{m}$ )	TSV process	Stacking technology	Bonding technology	Status
CEA-Leti [74]	70	unknown	via-last	unknown	D2W	Production (CMOS imager)
	2 -3	unknown	unknown	unknown	D2W	Electrically tested
IBM [58]	2-90	12-200	via-first via-middle via-last	SOI/F2F Bulk/F2F Bulk/F2B	D2D D2W W2W	Test-vehicle demonstration
IMEC [87]	5	unknown	via-middle	unknown	D2W	Functional circuit
Intel [12]	unknown	190	unknown	unknown	unknown	80-core prototype
Samsung [55]	7.5	unknown	unknown	unknown	unknown	Production (wide I/O DRAM)
Tezzaron [71]	1.2	<4	unknown	unknown	W2W	Production (Super-Contact)
TSMC [60]	unknown	30-40	unknown	F2B	D2W	Production (Xilinx 2.5D FPGA)

The emergence of 3D integration requires that the whole design flow be 3D-aware; the challenges for physical design tools to support 3D integration technologies stem from several issues [7][9][22]. Multiple dies in a 3D IC are connected using TSVs. However, TSVs are usually etched or drilled through device layers by special processes and are

costly to fabricate. A large number of TSVs will degrade the yield of the final chip. Also, under the current technologies, TSV pitches are very large compared to the sizes of regular metal wires, usually around 5-10  $\mu\text{m}$ . In 3D ICs TSVs are usually placed at the whitespace between the macro blocks or cells, so the number of TSVs will not only affect the routing resource but also the overall chip or package areas. Therefore, the number of TSVs in the circuit needs to be controlled and minimized.

Latency and power are still important criteria, where the floorplanner and placer have to consider the timing and power characteristics of TSVs. The thermal issues in 3D ICs become critical: (i) The vertically stacked multiple layers of active devices cause a rapid increase in power density; (ii) The thermal conductivity of the dielectric layers between the device layers is very low compared to silicon and metal. For instance, the thermal conductivity at room temperature (300 K) for the die-gluing material, epoxy, is 0.05 W/m/K [80], and the thermal conductivity of the metal layers embedded in a thickness of 12  $\mu\text{m}$  dielectric is about 2 W/m/K [81]. Both thermal conductivities are much smaller than the thermal conductivity of silicon (150 W/m/K) and copper (401 W/m/K). Therefore, the thermal issue needs to be considered during every step of the 3D physical design flow.

## **1.2 Overview of the Dissertation**

The remainder of this dissertation is organized as follows:

In Chapter 2 we first review prior work on the placement problem in physical design. We then discuss the new challenges emerging in 3D IC design, and finally we will formulate the problem that we are addressing in this research.

Chapter 3 describes a 3D physical design flow, named *3D-Craft*, currently being developed at UCLA. Specifically, we describe a preliminary 3D physical design flow and introduce the key modules, including a 3D floorplanner, a 3D placer, and a 3D router. The experimentation on the 3D physical design of a real-life microprocessor demonstrates the advantages of our system on effective wirelength reduction over 2D physical designs.

For the rapid adoption of 3D integration, Chapter 4 presents several transformation approaches that convert a given 2D placement to a 3D placement. The transformation approaches include (i) *local stacking*, (ii) *folding-2*, (iii) *folding-4*, and (iv) *window-based stacking/folding* transformations. These approaches are capable of obtaining different inter-die TSV densities for different 3D IC manufacturing abilities. Moreover, a novel relaxed conflict-net (RCN) graph-based die assignment method is applied to further refine the 3D placements in terms of TSV density and peak temperature.

Chapter 5, Chapter 6, and Chapter 7 will thoroughly discuss the analytical 3D placement framework, which is the focus of this dissertation.

Chapter 5 solves the gradient computation problem for density constraints with global smoothing in analytical placers. Analytical placement is the state-of-the-art placement technique, where the non-overlapping constraints are approximated by density constraints. In this chapter we unify a wide range of density smoothing techniques, called global smoothing, and present a highly efficient method to compute the gradient of such smoothed densities used in the best academic placers. Experiments show that replacing the approximated gradient computation in mPL6 with the exact gradient computation

improves the wirelength by 15% on the IBM-HB+ benchmark and by 3% on the modified ISPD'05 and ISPD'06 placement contest benchmarks with movable macros.

Chapter 6 describes analytical 3D placement approaches that minimize a weighted sum of the total wirelength and the inter-die TSV density subject to area density constraints. Specifically, two approaches are discussed for area overlap removal: (i) bell-shaped area projection with pseudo layers, and (ii) Huber function-based local smoothing. In addition, a multiple-stepsizes scheme is derived for mixed-size circuits to obtain better quality. Experimental results show that this analytical placer is effective to achieve tradeoffs between the wirelength and the inter-die TSV density. Compared to the transformation-based 3D placement approaches in Chapter 4, the analytical approaches are able to achieve 12% shorter wirelength and 29% fewer TSVs compared to the previous results with shortest wirelength; they are also able to achieve 20% shorter wirelength and 50% fewer TSVs compared to the previous results with fewest TSVs.

Chapter 7 tackles the thermal issues by taking advantage of the thermal conductivity of TSVs. Our study indicates that the solutions from a thermal-aware 3D placer without modeling the thermal conductivity of TSVs are suboptimal. However, it is runtime-impractical to use a thermal solver to evaluate the intermediate placement solutions (which happens thousands of times). In this work we are able to prove that in a thermally optimal condition of the TSV distributions in a 3D placement. Based on this criterion, we implement an efficient thermal-aware 3D placement tool. Compared to the methods that prefer a uniform power distribution that only results in an 8% peak temperature reduction,

our method reduces the peak temperature by 34% on average with even slightly less wirelength overhead.

A case study of 3D physical design is reported in Chapter 8, where we applied 3D-Craft to the LEON3 microprocessor and analyzed the impact of 3D integration. Chapter 9 concludes our research efforts in physical design automation for 3D integration and proposes future work.



## Chapter 2

### Problem Formulation and Related Work

#### 2.1 Introduction

The physical design process for 3D ICs is similar to that used for the traditional 2D physical design, in the sense that it transforms the circuit representation from a netlist into a geometric representation by the steps of floorplanning, placement and routing. While the multiple-layer metals have already had a 3D structure in traditional ICs for interconnects, the 3D integration allows multiple layers of logic devices to be integrated in the third dimension by bonding stacks of multiple dies to form 3D chips. Each die, which is similar to a traditional 2D IC, consists of one silicon layer and several metal layers, and different dies are connected by TSVs.

A typical adhoc 3D physical design flow, which is developed for a 3-die 3D chip tape-out, is shown in Figure 3. This kind of design flow partitions the whole design into multiple designs at the very beginning of the flow. It has two disadvantages: (i) The partitioning relies heavily on the designer's experience. It becomes less feasible for large-scale designs and the designs without a clear mapping from the logical partitions to the physical dies. (ii) Without 3D-aware design tools, the 2D design tools are not able to obtain consistent 3D design data. Instead, the data from one die has to be passed to another die in an iterative way, which makes the solution suboptimal [78].

Due to the lack of 3D design tools, we were motivated to develop a fully 3D-aware physical design flow, which mainly focuses on the 3D placement stage.

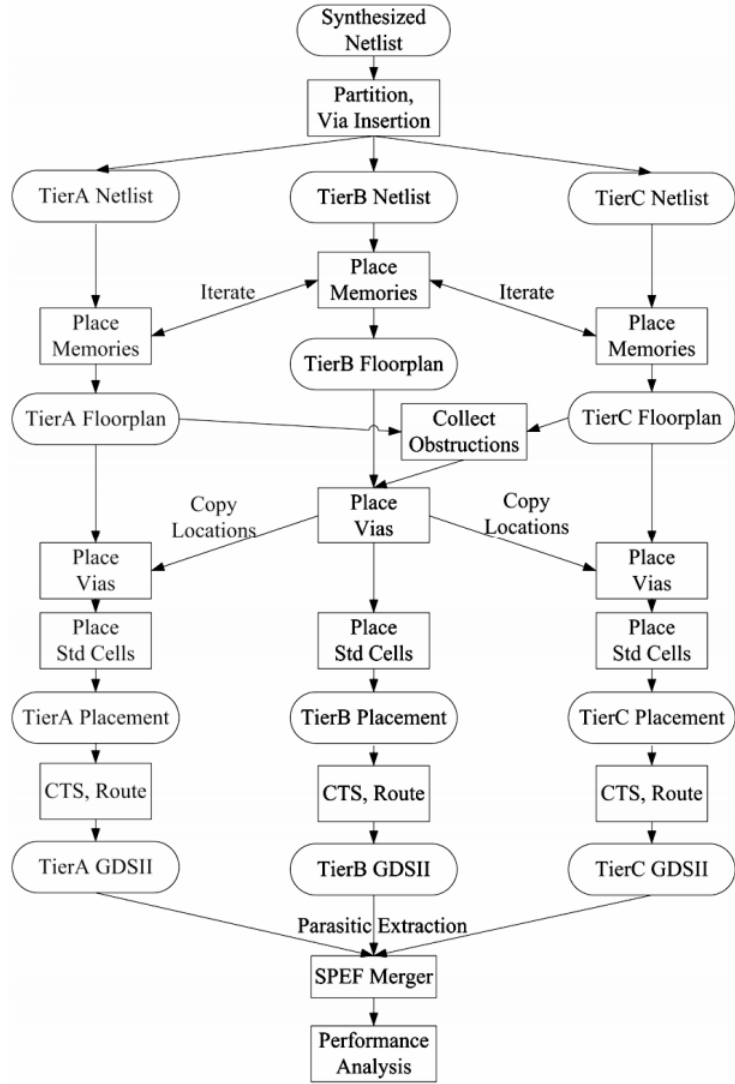


Figure 3. A typical adhoc 3D physical design flow [42]

## 2.2 Preliminaries

In this section we define the basic concepts that will be used throughout the rest of the discussions.

**Definition 2.2.1 Netlist** A netlist describes the connectivity of a circuit in the physical design domain. A *node* in a netlist represents either (i) an I/O pad, (ii) a standard cell (a basic logic or memory element), (iii) an intellectual property (IP) block, or (iv) a high-

level module. A *net* connects two or more nodes together, where the interface between a node and a net is called a *pin*. In this dissertation we assume all nodes are hard, where each node is associated with a fixed width and a fixed height.

**Definition 2.2.2 Placement region** Placement is a process of mapping from the nodes in the structural netlist to their physical locations in a chip. In traditional 2D physical design, the placement region is usually a rectangle with the same size as the target die. In 3D physical design, multiple dies are bonded to form a 3D chip. Thus, the placement regions are multiple aligned rectangles.

**Definition 2.2.3 Wirelength** The wirelength of a net is the total length of the routed wires of its implementation. The total wirelength of a circuit has roughly positive correlations to its performance, routability, and dynamic power, and thus can be used as a metric of the placement quality. However, the routed wirelength is difficult to exactly predict before actual routing, the *half-perimeter wirelength* (HPWL) is usually used as a placement metric instead. In the remainder of this dissertation, the term wirelength refers to HPWL by default.

**Definition 2.2.4 Non-overlapping constraints** A necessary condition of a legal 3D placement is that for every pair of nodes, they are either placed on different dies, or they are placed on the same die without any overlaps.

**Definition 2.2.5 Timing constraints** Every netlist is associated with a *timing graph*. A *timing point* is usually an input/output terminal or an input/output pin of a node. Given a required clock frequency (frequencies), the *arrival time* (AT) and *required arrival time* (RAT) can be computed at each timing point by static timing analysis (STA), where the

difference between RAT and AT is defined as the *slack* of a timing point. The slacks are related to the logic delays, the interconnect delays, and the required frequencies. The timing constraints require that all the slacks are nonnegative.

**Definition 2.2.6 Thermal constraints** Given a placement and the power dissipation of the nodes, a 3D *power map* can be computed. A compact thermal model is built based on the physical structure of the target 3D IC (the thickness and thermal conductivity of all the stacked materials) and the cooling capability of the heat sink. A 3D *thermal map* is then computed from the compact thermal model and the power map. To avoid overheated 3D ICs, a *thermal constraint* is set on the values in the thermal map, which must not exceed a user-defined thermal budget.

## 2.3 Previous Work

The state-of-the-art techniques for 2D placement can be classified into flat placement approaches, top-down partitioning-based approaches, and multilevel placement approaches [65]. These approaches exhibit scalability for the growing complexity of modern VLSI circuits. In order to handle the scalability issues, these approaches divide the placement problem into three stages: (i) global placement, (ii) legalization, and (iii) detailed placement. Given an initial solution, the global placement refines the solution until the area of placeable objects in every pre-defined region is not greater than the capacity of that region. These regions are handled in a top-down fashion from coarsest level to finest level by the partitioning-based approaches and the multilevel placement approaches, and are handled in a flat fashion at the finest level by the flat placement approaches. After the global placement, legalization proceeds to assign placeable objects

to placement sites, and the detailed placement performs local refinements to obtain the final solution.

As the modern 2D placement techniques evolve, a number of 3D placement techniques are also developed to cope with the opportunities and challenges from 3D integration. Most of the existing techniques, especially at the global placement stage, can be viewed as extensions of 2D placement techniques. We group the 3D global placement techniques into the following categories:

- *Partition-based approaches* [35][45][5]. This kind of approach applies a sequence of bi-partitions to perform the global placement in a divide-and-conquer paradigm, with inter-die  $z$  cuts to minimize the number of TSVs, or intra-die  $x/y$  cuts to minimize the wirelength. The cost of partitioning is measured by the cutsize of the nets across partitions, where the cuts can be further weighted to reflect temperature [45] and routability [5]. The order of the cutting directions determines the TSV density: the earlier that  $z$  cuts are performed, the lower the TSV density; the later that  $z$  cuts are performed, the higher the TSV density.
- *Force-directed approaches* [43][52][46]. Since the unconstrained quadratic wirelength minimization will result in a great amount of overlaps, repulsive forces are introduced for overlap removal. The repulsive forces are computed iteratively, which eventually reduce the overlaps to an acceptable amount. There are two methods for computing the repulsive forces: (i) the forces point to the negative gradient of the area density field [43][52]; or (ii) the forces point to the desired

placement estimated by cell shifting [46]. Additional thermal-aware repulsive forces [43] can be computed from the temperature field.

- *Analytical approaches* [82], a.k.a. generalized force-directed approaches. The analytical solver minimizes a sequence of penalized objectives, one of which is usually a wirelength/TSV objective plus a weighted overlap penalty. The weight of the overlap penalty increases from a small number until the overlaps are reduced to an acceptable amount. For example, [82] computes an overlap penalty from the unevenness of area distribution by computing DCT frequencies, and locally approximates this penalty function by a quadratic function. Minimizers of such overlap penalty are legal placements. The work in [48] extends the bell-shaped function to measure the area density for 3D cubes for the formulation of the 3D area density constraints.
- *Partition-first approaches* [1][54]. Unlike the approaches mentioned above, this kind of approach divides the 3D global placement into two steps: (i) the vertical partitioning step to determine the die assignment, and (ii) the intra-die placement step to determine the locations of placeable objects inside every die. The vertical partitioning step is performed either by mincut partitioning [1], or “controlled-size cut” partitioning [54]. The intra-die placement can be implemented by straightforwardly extending any 2D placement approaches, like the simulated annealing approach [1] or the force-directed approach [54].

There are few publications specific to the 3D legalization and the 3D detailed placement problems. Usually, legalization and detailed placement can be completed by running 2D legalizers and 2D detailed placers die-by-die.

## 2.4 Problem Statement

This section formulates the 3D placement problem of interest throughout this dissertation. The wirelength objective and the non-overlapping constraints are formulated as the basic concerns. In addition, the thermal constraints and the timing constraints that relate to the reliability and performance are also discussed.

Given a netlist  $H = (V, E)$ , the number of dies  $K$ , and the per-die placement region  $R = [0, W_{\text{die}}] \times [0, H_{\text{die}}]$ , where  $V$  is the set of nodes, including standard cells, intellectual property (IP) blocks, and other high-level hard macros, and  $E$  is the set of nets, a placement  $(x_i, y_i, z_i)$  of node  $v_i \in V$  satisfies that its center is  $(x_i, y_i) \in R$  and its die assignment is  $z_i \in \{1, 2, \dots, K\}$ . The 3D placement problem is to find an optimal placement  $(x_i, y_i, z_i)$  for every  $v_i \in V$ , so that an objective function of the weighted total wirelength is minimized, subject to constraints such as non-overlapping constraints, performance constraints, and temperature constraints.

### 2.4.1 Wirelength Objective Function

The quality of a placement solution can be measured by its performance, power and routability, but the measurement is nontrivial. In order to model these aspects during the

optimization stage, the weighted total wirelength is a widely accepted metric of placement qualities [65]. Formally, the objective function is defined as

$$\text{OBJ}(\bar{x}, \bar{y}, \bar{z}) = \sum_{e \in E} (1 + \gamma_e) (\text{WL}(e) + \alpha_{\text{TSV}} \cdot \text{TSV}(e)) \quad (3)$$

The objective function depends on the placement  $(\bar{x}, \bar{y}, \bar{z})$ , and it is a weighted sum of the wirelength  $\text{WL}(e)$  and the number of TSVs  $\text{TSV}(e)$  over all the nets. The weight  $(1 + \gamma_e)$  reflects the criticality of the net  $e$ , which is usually related to the performance optimization (will be discussed in Section 2.4.3). The unweighted wirelength is represented by setting  $\gamma_e$  to zero. This weight is able to model thermal effects by relating the weight to the thermal resistance, electronic capacitance, and switching activity [45].

The wirelength  $\text{WL}(e)$  is usually estimated by the half-perimeter wirelength (HPWL):

$$\text{WL}(e) = \left( \max_{v_i \in e} \{x_i\} - \min_{v_i \in e} \{x_i\} \right) + \left( \max_{v_i \in e} \{y_i\} - \min_{v_i \in e} \{y_i\} \right) \quad (4)$$

Similarly,  $\text{TSV}(e)$  is modeled by the range of  $\{z_i : v_i \in e\}$  [43][45][34]:

$$\text{TSV}(e) = \max_{v_i \in e} \{z_i\} - \min_{v_i \in e} \{z_i\} \quad (5)$$

The coefficient  $\alpha_{\text{TSV}}$  is the weight for the TSVs; it models a TSV as a length of wire. For example, the work [36] evaluates that under the 0.18 $\mu\text{m}$  Silicon-On-Insulator (SOI) technology a TSV with a length of 3 $\mu\text{m}$  is roughly equivalent to 8 $\mu\text{m}$ -20 $\mu\text{m}$  of metal-2 wire in terms of capacitance, and it is equivalent to about 0.2 $\mu\text{m}$  of metal-2 wire in terms of resistance. Thus a coefficient  $\alpha_{\text{TSV}}$  between 8 $\mu\text{m}$  and 20 $\mu\text{m}$  can be used for optimizing power or delay in this case.



## 2.4.2 Non-Overlapping Constraints

The ultimate goal of non-overlap constraints can be expressed as the following:

$$\begin{aligned} |x_i - x_j| &\geq (w_i + w_j)/2 \\ \text{or} & \quad \text{for every node pair } (v_i, v_j) \text{ with } z_i = z_j \\ |y_i - y_j| &\geq (h_i + h_j)/2 \end{aligned} \quad (6)$$

where  $(x_i, y_i, z_i)$  is the placement of node  $v_i$  with its width and heights as  $w_i$  and  $h_i$ , respectively. The same applies to node  $v_j$ . Such constraints were used directly in some analytical placers early on, such as [14].

However, this formulation leads to a huge number of either-or constraints, which grows quadratically with the number of nodes. This amount of constraints is not practical for modern large-scale circuits.

To formulate and handle these pair-wise non-overlapping constraints, modern placers use a more scalable procedure to divide the placement into global placement and detailed placement. Detailed placement assigns every node to a legal site to satisfy the constraints in equation (6), and allows global placement to relax the pair-wise non-overlapping constraints by regional area density constraints:

$$\sum_{v_i \in V} \text{Overlap}(\text{bin}_{m,n,k}, v_i) \leq w_{\text{bin}} \cdot h_{\text{bin}} \quad \begin{array}{l} 1 \leq m \leq M \\ \text{for } 1 \leq n \leq N \\ 1 \leq k \leq K \end{array} \quad (7)$$

For a 3D IC with  $K$  dies, each die is divided into  $M \times N$  bins for the measurement of overlaps, where the width and height of each bin is  $w_{\text{bin}} = W_{\text{die}}/M$  and  $h_{\text{bin}} = H_{\text{die}}/N$ , respectively. If every  $\text{bin}_{m,n,k}$  satisfies the inequality (7), the global placement satisfies

the non-overlapping constraints. Examples of the regional area density constraints on one die are given in Figure 4.

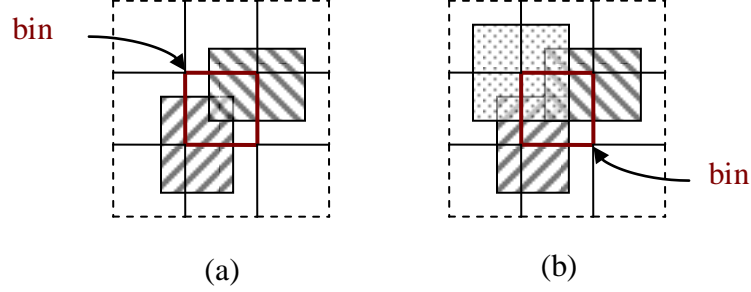


Figure 4. (a) Area constraint is satisfied; (b) Area constraint is not satisfied

The overlapped area between  $bin_{m,n,k}$  and node  $v_i$  is defined as:

$$\text{Overlap}(bin_{m,n,k}, v_i) = \delta(z_i, k) \cdot \text{Overlap}_x(bin_{m,n,k}, v_i) \cdot \text{Overlap}_y(bin_{m,n,k}, v_i) \quad (8)$$

where  $\delta(z_i, k)$  indicates whether node  $v_i$  is on the same die as  $bin_{m,n,k}$ , and the functions  $\text{Overlap}_x$  and  $\text{Overlap}_y$  are the overlaps between the projections of node  $v_i$  (a rectangle) and  $bin_{m,n,k}$  (another rectangle) on the x-axis and the y-axis, respectively. The center of  $bin_{m,n,k}$  is located at  $((m+1/2) \cdot w_{bin}, (n+1/2) \cdot h_{bin})$  on die  $k$ .

Formally, these functions are defined as the following,

$$\delta(z_i, k) = \begin{cases} 1 & (z_i = k) \\ 0 & (z_i \neq k) \end{cases} \quad (9)$$

$$\begin{aligned} & \text{Overlap}_x(bin_{m,n,k}, v_i) \\ = & \text{common\_length}([(m-1) \cdot w_{bin}, m \cdot w_{bin}], [x_i - w_i/2, x_i + w_i/2]) \end{aligned} \quad (10)$$

$$\begin{aligned} & \text{Overlap}_y(bin_{m,n,k}, v_i) \\ = & \text{common\_length}([(n-1) \cdot h_{bin}, n \cdot h_{bin}], [y_i - h_i/2, y_i + h_i/2]) \end{aligned} \quad (11)$$

where  $\text{common\_length}([a,b],[c,d])$  is the common length between the segment  $[a,b]$  and the segment  $[c,d]$ .

Therefore, there are only  $M \times N \times K$  regional area density constraints, the number of which is usually much smaller than the number of node pairs.

### 2.4.3 Timing Constraints

Let  $\text{PI}$  be the set of primary inputs,  $\text{PO}$  be the set of primary outputs,  $\text{OUT}_{\text{clk}}$  be the set of output pins of clock nodes, and  $\text{IN}_{\text{clk}}$  be the set of input pins of clock nodes. As defined in Section 2.2, the timing constraints can be set as the following,

$$\text{slack}_i(\bar{x}, \bar{y}, \bar{z}) \geq 0 \quad \text{for } i \in \text{PO} \cup \text{IN}_{\text{clk}} \quad (12)$$

According to the work on net weighting schemes in timing-aware placement [17], these non-negative constraints on slacks can be satisfied by iteratively minimizing the weighted wirelength objectives. Provided a net weighting function (as  $\gamma_e$  in Section 2.4.1) that satisfies the asymptotic slack control properties, the net weighting schemes are guaranteed to converge to the original timing-constrained placement problem.

### 2.4.4 Thermal Constraints

Given the power dissipation  $p_i$  of every node  $v_i \in V$ , the 3D power map is computed as the following assuming the power is evenly distributed inside a node,

$$P_{m,n,k}(\bar{x}, \bar{y}, \bar{z}) = \sum_{v_i \in V} \frac{P_i}{w_i h_i} \cdot \text{Overlap}(\text{bin}_{m,n,k}, v_i) \quad (13)$$

where the placement region is divided into  $M \times N \times K$  bins.<sup>†</sup>

In a compact thermal model, the vectorized 3D power map (with  $M \times N \times K$  elements) and the vectorized 3D thermal map (also with  $M \times N \times K$  elements) is in a linear relationship, which is expressed as

$$A_{(M \times N \times K)^2} \left( T_{m,n,k}(\bar{x}, \bar{y}, \bar{z}) \right)_{M \times N \times K} = \left( P_{m,n,k}(\bar{x}, \bar{y}, \bar{z}) \right)_{M \times N \times K} \quad (14)$$

Thus, the 3D thermal map is computed by solving the linear system above. The maximum element of  $T_{m,n,k}(x, y, z)$  is the maximum temperature, so that we can formulate the thermal constraints as

$$T_{m,n,k}(\bar{x}, \bar{y}, \bar{z}) \leq T_{\max} \quad \begin{array}{l} 1 \leq m \leq M \\ \text{for } 1 \leq n \leq N \\ 1 \leq k \leq K \end{array} \quad (15)$$

#### 2.4.5 Formulation of the 3D Placement Problem

Based on the formulations in the previous subsections, the 3D placement problem of interest is formulated as,

$$\begin{array}{ll} \text{minimize} & \sum_{e \in E} (1 + \gamma_e) (\text{WL}(e) + \alpha_{\text{TSV}} \cdot \text{TSV}(e)) \\ \text{subject to} & \sum_{v_i \in V} \text{Overlap}(\text{bin}_{m,n,k}, v_i) \leq w_{\text{bin}} \cdot h_{\text{bin}} \quad \begin{array}{l} 1 \leq m \leq M \\ \text{for } 1 \leq n \leq N \\ 1 \leq k \leq K \end{array} \\ & \left( T_{m,n,k}(\bar{x}, \bar{y}, \bar{z}) \leq T_{\max} \right) \\ & \left( \text{Delay}_i(\bar{x}, \bar{y}, \bar{z}) \leq 1/\text{freq} \right) \quad \text{for } i \in \text{PO} \cup \text{IN}_{\text{clk}} \end{array} \quad (16)$$

The thermal constraints will be set as soft constraints. We shall demonstrate several approaches to reduce the peak temperature, and obtain the tradeoffs between the

---

<sup>†</sup> To reduce the number of symbols, we use the same number of bins to as in the non-overlapping constraints. Please note that these two resolutions are independent and could be with different number of bins.

wirelength quality and the temperature reduction. The timing constraints are listed for completeness, which could be integrated in the wirelength objective with the net weighting schemes [17] to meet the performance target. In the following sections of this dissertation, we shall focus on the techniques used to solve the 3D placement problem with wirelength objective and the non-overlapping constraints; the temperature and the performance will be evaluated, but not be set as hard constraints.

## Chapter 3

### 3D-Craft: A 3D Physical Design Flow Based on OpenAccess

3D integration technologies have recently attracted great attention due to the potential performance improvement, power consumption reduction and heterogeneous integration. In this chapter we present an OpenAccess-based 3D physical design flow, named 3D-Craft, to facilitate the rapid adoption of 3D integration. The OpenAccess extension for 3D-Craft is discussed, and the key components that include a 3D floorplanner and the 3D placer mPL-3D are presented. In Chapter 8, we will also demonstrate the application of 3D-Craft for the 3D physical design of an open-source processor, and show that the 3D implementation can reduce both the half-perimeter wirelength and the routed wirelength by about 30% compared to the 2D implementation.

#### 3.1 Introduction

In recent years 3D physical design has attracted an increasing amount of attention. There is a significant amount of work on the floorplanning [30], placement [31] and routing [25] for 3D integration. However, all these tools developed by different research groups, using different formats to represent the design data, create barriers for researchers who need to make use of the existing design automation tools to conduct further studies on 3D IC. This problem motivated us to develop an infrastructure for the 3D design data representation and assist in the interoperation of physical design tools.

In this chapter we present an OpenAccess [101] extension for 3D physical design automation. The main difficulty in developing the OpenAccess extension is to make it

applicable, not only to a specific 3D integration technology, but also applicable to other possible 3D integration technologies (as introduced in Section 1.1). The issues include how to represent the multiple-die structure for a 3D integration technology and how to represent TSVs in a 3D design. To solve these issues, we have made the following contributions in this chapter:

- We define the database architecture for 3D physical design based on OpenAccess. This architecture is capable of representing the multi-die structure in a general way, and it is also capable of representing the structure of TSVs and their occurrence in a 3D design.
- We implement 3D-Craft based on the OpenAccess extension. This is a 3D physical design flow including a 3D floorplanner, a 3D placer with TSV planning, and the interface with commercial detailed routers.

The remainder of this chapter is organized as follows. Section 3.2 illustrates an overview of 3D-Craft, introduces OpenAccess and discusses the issues and solutions in the extension for 3D ICs. Section 3.3 presents the key components in 3D-Craft. Finally, Section 3.4 present conclusions and discusses future work.

## **3.2 Overview of 3D-Craft and OpenAccess Extension for 3D Design**

The 3D-Craft's physical design flow is illustrated in Figure 5. Several physical design tools are integrated based on the 3D OpenAccess, including a 3D floorplanner, a 3D placer with TSV planning, and a commercial 2D detailed router. A thermal resistive network model is also integrated for thermal evaluations. The detailed information for this collection of physical design tools will be presented in Section 3.3. Before we present

the components in 3D-Craft, we shall first introduce OpenAccess and discuss the issues and solutions in the extension for 3D physical design.

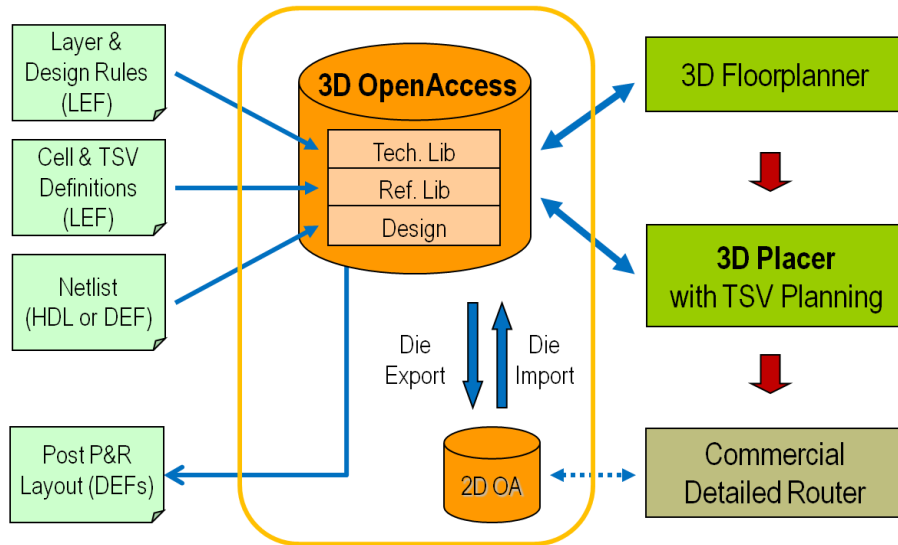


Figure 5. Overview of 3D-Craft

### 3.2.1 Preliminaries on OpenAccess

OpenAccess is an infrastructure designed and maintained by Si2 [101] to achieve the interoperability of EDA applications and design data. The design data management is through a C++ API that defines classes and member functions to create, access and manipulate the databases. The OpenAccess API consists of a set of packages, where the technology database package and the design database package are the most important packages related to place and route (P&R) tools.

The technology library holds data that is generally applied across all the designs developed from a specific technology. For example, the minimum wire width of a metal layer is specified in the technology library as a layer constraint.



The design database refers to design-specific data, which includes cell/macro libraries and the top design that consists of instances of cells and macros. The data in the design library is represented in a hierarchical way; an example is illustrated in Figure 6. In this example, the design library contains the cells XOR/AND/OR, the “macro” HalfAdder, and the “top design” FullAdder. The top design consists of instances of designs in the cell library and the macro library.

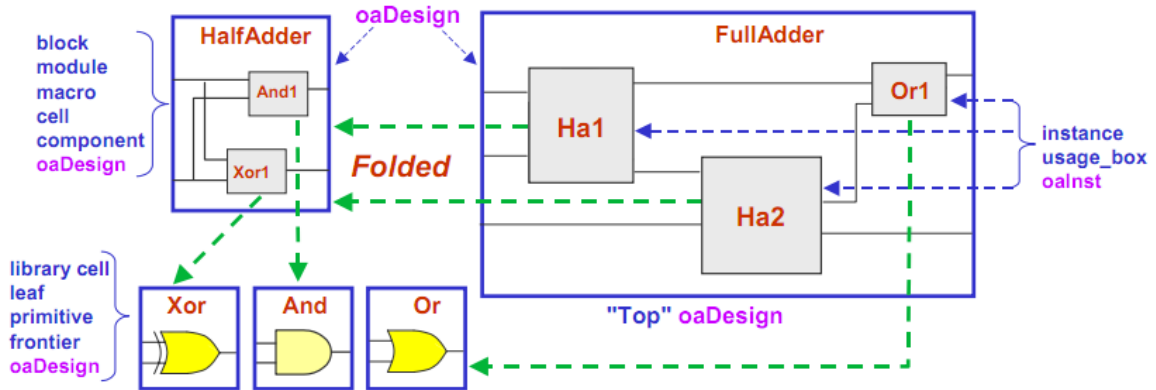


Figure 6. An example of the hierarchical design database [101]

The *oaAppDef* mechanism is a way to add extension values to existing database objects, through which we attach additional information to OpenAccess objects to realize the 3D design representation. An example is shown in Figure 7, where the attributes “name,” “cell name,” and “origin” are native attributes of the *oaInst* objects. And we use *oaIntAppDef* to add an integer attribute to these objects for the extended information in 3D designs. So every instantiation of *oaInst* includes the native attributes of “name,” “cell name,” and “origin,” as well as the extended attribute of “die.” The detailed use of *oaAppDef* for OpenAccess extension will be discussed in Section 3.2.3.

<b>oaInst</b>	Native Attribute Types			<b>oaIntAppDef</b>
	↓	↓	↓	↓
Instantiations	name	cell name	origin	die
	“Ha1”	“HalfAdder”	(2,6)	1
	“Ha2”	“HalfAdder”	(10,1)	2
	“Or1”	“Or”	(18,9)	1

Figure 7. An example of the *oaAppDef* mechanism

### 3.2.2 Issues in Extending OpenAccess for 3D Physical Design

Although OpenAccess is general enough for data representation of traditional 2D designs, it lacks some necessary features for 3D IC technologies; therefore, there are two very important issues to be considered in the extension.

The first issue is the multiple-die structure in a 3D design. Physically, a 3D design can be viewed as a stack of multiple 2D dies. There are face-to-back and face-to-face bonding of dies to form a 3D IC (see Section 1.1). Therefore, the OpenAccess extension should be able to represent the multiple-die structure and also the bonding method.

The other issue is representation of TSVs. In current 3D integration technologies, the size of a TSV is comparable to the size of an inverter, and it also has electrical and mechanical characteristics that are different from traditional vias; thus, TSVs should be represented explicitly and acknowledged by physical design tools.

### 3.2.3 Solutions for Extending OpenAccess for 3D Designs

The database architecture for 3D designs consists of a technology library, a reference library and a design library.

The technology library stores the technology information of each die, where each die has a structure similar to a single 2D design. The design rules for the metal layers and the traditional via layers are described in the technology library, as well as the RC characteristics of these layers. To represent all this information, the metal layers and normal via layers are listed in order from bottom to top. A string called *bonding* is added to the *oaDesign* object by the *oaAppDef* mechanism to describe the bonding order of dies and the order of the metal layers and normal via layers. For example, the string “FBB” tells us that there are three dies, “F,” “B,” and “B,” in a 3D chip. The first die in a face-on-top (“F”) direction starts with a silicon layer, followed by several interconnect layers. The second and third dies in a back-on-top (“B”) direction start with several interconnect layers first, followed by the silicon layer. In such a way, the multi-die structure is represented in the OpenAccess database, which is the solution of the first issue referred to Section 3.2.2.

The reference library contains a TSV library and a cell/macro library. Based on the fact that a TSV consumes silicon area in a way similar to a cell or a macro, we encapsulate the TSV as a pseudo cell in the TSV library. The metal layer consumptions, as well as the silicon layer (which a TSV drills through) consumption, are captured by the pseudo cell. Different TSVs connecting different pairs of dies may have different structures; thus there may be multiple pseudo cell prototypes in the TSV library. This solves the second issue referred to in Section 3.2.2. The cell/macro library contains the abstract (bounding box and I/O pins) of standard cells and IP blocks that are available for a design instance.

The design library stores the netlist and its P&R information. The original netlist representation before placement and routing is the same as the netlist representation for 2D designs. After placement, an integer called *die* is added to every *oaInst*, as shown in Figure 7, to represent which die a cell instance is placed on. After TSV planning, the TSV locations are determined, and instances of the pseudo cells in the TSV library are created to represent these TSVs. When the TSV locations are determined, it is straightforward to partition the 3D design to multiple 2D dies physically and run commercial detailed routers.

### **3.3 Components in 3D-Craft**

We have illustrated our thermal-driven 3D physical design flow in Figure 5. The design flow is composed of three major steps: 3D floorplanning, 3D placement and detailed routing. The macro blocks or cells are first placed into the stacked dies, and the interconnections are then routed. The individual tool components in the design flow can interoperate with each other through OpenAccess.

#### **3.3.1 Resistive Thermal Model**

Since the clock cycle of a modern chip is orders of magnitude smaller than the timing constant for thermal conduction, considering the steady-state temperature is enough. Also, only the heat generated by transistor switches is considered, so the cells are treated as the only heat sources with constant given power densities. Since a heat sink is usually attached to the substrate, the bottom side of the tile stack is isothermal of constant

temperature. Because the chip is usually packaged in thermally insulated materials, the four side walls and top of the chip are treated as adiabatic.

A thermal resistive model for 3D IC is developed in [80]. Compared to the accurate simulation tool, the error of the resistive network model is smaller than 2%. A tile structure is imposed on the circuit stack (as shown on the left of Figure 8). Each tile stack contains an array of tiles, one from each device layer (as shown in the middle of Figure 8). A tile stack is modeled as a resistive chain (as shown on the right of Figure 8). The tile stacks are connected by lateral resistances. A voltage source is used for the isothermal base of heat sink temperature, and current sources are present in every tile to represent heat sources.

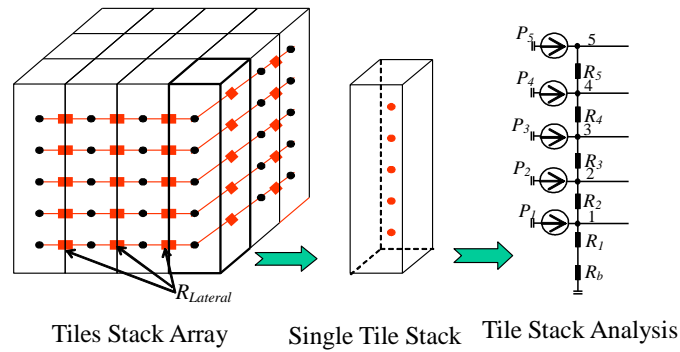


Figure 8. Resistive thermal model

The tile stacks are connected by lateral resistances. A voltage source is used for the isothermal base of heat sink temperature, and current sources are present in every tile to represent heat sources.

### 3.3.2 Thermal-Aware 3D Floorplanning

The 3D floorplanning problem is similar to the 3D placement problem, except that the nodes in the netlist are high-level modules, and the number of nodes, in the scale of tens to hundreds, is much smaller than the placement problem. Due to the problem size, stochastic combinatorial optimization techniques, such as simulated annealing algorithms [56], are preferred.

We integrated the 3D floorplanner in [32], which is a 3D floorplanning algorithm for 2D blocks. The 3D floorplan representation is composed of two parts: the transitive closure graph which is used to represent each die, and a bucket structure to store the relationship between nodes on different dies. Two versions of cost functions can be used during 3D floorplanning, including the temperature-penalized cost ( $\text{cost}_{TP}$ ) and the temperature-constrained cost ( $\text{cost}_{TC}$ ):

$$\begin{aligned}\text{cost}_{TP} &= \alpha \cdot \text{HPWL} + \beta \cdot \text{Area} + \gamma \cdot \text{TSV} + \eta \cdot T \\ \text{cost}_{TC} &= \alpha \cdot \text{HPWL} + \beta \cdot \text{Area}^* + \gamma \cdot \text{TSV}\end{aligned}\tag{17}$$

where the terms HPWL, Area, TSV are the normalized wirelength, chip area, and TSV number, respectively. To speed up the simulated annealing process, the normalized temperature  $T$  in  $\text{cost}_{TP}$  is estimated by the vertical heat flow analysis [32]. The term  $\text{Area}^*$  in  $\text{cost}_{TC}$  is the normalized chip area consumed by both nodes and TSVs, where the number of additional thermal TSVs to meet the thermal constraint is estimated by the vertical thermal TSV distribution in [26].

In the design flow, the 3D floorplanner is able to plan either a hierarchical netlist or a flat netlist. In the latter case, the flat netlist will be partitioned into about 80 to 100

partitions, as the virtual high-level nodes for planning, in order to reduce the problem size and the runtime. These partitions will be flattened before 3D placement. Some large hard macros, in a mixed-size netlist, will be legalized and fixed after 3D floorplanning, so that the 3D placer can focus on placing the standard cells and small macros.

### 3.3.3 Multilevel Analytical 3D Placement (mPL-3D)

The 3D placement is the focus of this dissertation. As formulated in Section 2.4, the 3D placer solves the following problem:

$$\begin{aligned}
& \text{minimize} && \sum_{e \in E} (1 + \gamma_e) (\text{WL}(e) + \alpha_{\text{TSV}} \cdot \text{TSV}(e)) \\
& \text{subject to} && \sum_{v_i \in V} \text{Overlap}(\text{bin}_{m,n,k}, v_i) \leq w_{\text{bin}} \cdot h_{\text{bin}} \quad \text{for } 1 \leq n \leq N \\
& && 1 \leq m \leq M \\
& && 1 \leq k \leq K
\end{aligned} \tag{18}$$

The multilevel analytical placement engine will be thoroughly discussed in Chapter 6. The solution obtained from global placement satisfies the regional area density constraints, thus the die assignment can be done by simply snapping the nodes to the nearest die. After die assignment, the remaining 2D legalization and detailed placement are done die-by-die as traditional 2D detailed placement. A two-step approach is used for the mixed-size legalization, where the macros are legalized first, followed by standard-cell legalization.

The experimental results indicate that a 3D standard cell placement for a 4-die 3D chip can reduce the wirelength by about 50% compared to a 2D placement, if the area of TSVs is ignored. The results also show that the analytical placement engine is able to

achieve good-quality tradeoffs between wirelength and TSV number, so it is adaptive for different 3D integration technologies.

### **3.3.4 TSV Insertion and Planning**

The TSVs can be inserted only after the die assignment of the nodes is determined, which is possible after either the 3D floorplanning or the 3D placement. In 3D-Craft, the netlist is modified using net splitting [54] during TSV insertion, which assumes a single TSV is inserted for every two neighboring dies connected by a net. The TSV insertion and net splitting can also be implemented using the minimum spanning tree approach [48], or by running an actual 3D router [25].

After TSV insertion and net splitting, another round of 3D placement with a fixed die assignment is performed to co-optimize the placement of nodes and TSVs, which further reduces the split wirelength and the peak temperature. Chapter 7 will discuss this co-optimization method in detailed.

### **3.3.5 3D Detailed Routing**

After the signal and thermal TSV planning, the 3D design can be decomposed into several 2D dies, and there is no difference between 3D detailed routing and 2D detailed routing. Therefore, it is best to make use of a well-developed commercial detailed router.

The decomposed 2D designs can be routed by any router, including the commercial routers in the Cadence Encounter [93] or the Magma Talus [98]. The routers read in the decomposed design and complete the detailed routing.



Magma Talus supports reading in existing global routing using Magma-TCL script. So, we implemented a tool to exchange the routing data with Magma Talus. The global routing paths are represented as boxes in the Magma data model. Thus the result of 3D global routing is imported to Talus by creating boxes using Magma-TCL script. The tool we implemented in the flow converts the global routing into a Magma-TCL script to create those boxes. After loading the script and running the detailed router on these 2D designs, the tool again parses the detailed routing information and writes back to the 3D design library. The routed 3D design is obtained after this step.

### **3.4 Conclusions and Future Work**

In this chapter we presented our extension of OpenAccess for 3D designs and presented 3D-Craft as a referenced 3D physical design flow. The database architecture for 3D designs is described, and the implementation details based on an OpenAccess extension through the *oaAppDef* mechanism is introduced. Several physical design tools with 3D awareness are integrated in the 3D-Craft, including a 3D floorplanner and the 3D placer with TSV planning. The application of 3D-Craft on the open-source processor LEON3 will be presented in Chapter 8. This shows that the 3D implementation can reduce both half-perimeter wirelength and routed wirelength effectively compared to the 2D implementation.

Future work includes the integration of the following physical design tools: 3D cube packing tools which support planning real 3D modules, the 3D power/ground network optimization tools, and the 3D clock tree synthesis tools.

## Chapter 4

### Thermal-Aware 3D Placement via Transformation

In this chapter we present our early attempt at 3D placement by generating thermal-aware 3D placement from existing 2D placement results through a two-step procedure: 3D transformation and refinement through die assignment. For 3D transformation, we develop local stacking transformation, folding-based transformation and the window-based stacking/folding transformation methods. The die assignment refinement procedure is based on a relaxed conflict-net (RCN) graph representation. The advantages of our 3D placement transformation include:

- The existing 2D placement core engine can be easily reused. Significant progress has been made on 2D placement in recent years. An efficient transformation from a 2D placement to a 3D placement enables us to leverage the existing high-quality 2D placers.
- A discrete die assignment algorithm based on graph representation is developed for die assignment of cells. No rounding for die assignment is necessary for analytical placement in 3D (as in some previous approaches).
- A simple yet effective thermal cost is derived for temperature optimization during die assignment. No time-consuming thermal profiling is needed during the optimization process.

- A flexible TSV number and wirelength tradeoff is offered by different transformation schemes and the parameter settings in the RCN graph-based layer assignment. This allows our algorithm to be used for different 3D technologies.

#### 4.1 Transformation-Based 3D Placement Framework

Figure 9 shows the framework of the transformation-based 3D placement algorithm. The components with a dashed boundary are the existing tools that we use. A 2D wirelength- and/or thermal- driven placer is first used to generate a 2D placement for the target design. The quality of the final 3D placement highly depends on the initial placement. The 2D placement is then transformed into a legalized 3D placement according to the given 3D technology. During the transformation, wirelength, TSV number and temperature are considered. A refinement process through die reassignment will be carried out after 3D transformation to further reduce the TSV number and bring down the maximum on-chip temperature. Finally, a 2D detailed placer will further refine the placement result for each die.

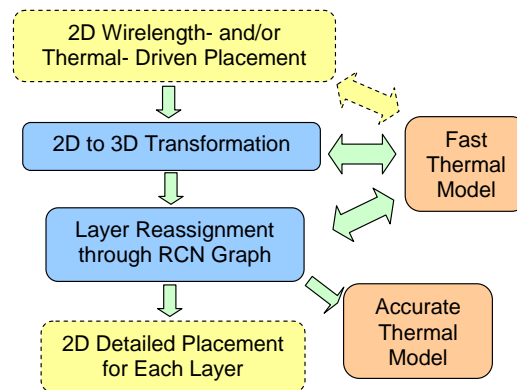


Figure 9. Placement transformation framework

## 4.2 Transformations for 3D Placement

In this section we discuss the transformation schemes from a 2D placement to a 3D placement. Initial optimized 2D placement is done on a chip  $K$  times larger than a single die of the 3D chip, where  $K$  is the number of dies, so that the total area of the 2D placement is equal to that of the target 3D chip. Given this 2D placement with minimized wirelength, *local stacking transformation* can achieve even shorter wirelength for the same netlist with 3D integration. We also present two folding-based transformation schemes, *folding-2* and *folding-4*, which can generate 3D placement with a very low TSV number. Moreover, TSV number and wirelength tradeoffs can be achieved by the *window-based stacking/folding*. All these transformation methods can guarantee wirelength reduction over the initial 2D placement.

### 4.2.1 Local Stacking Transformation

Local stacking transformation (LST) consists of two steps, stacking and legalization, as shown in Figure 10. The stacking step shrinks the chip uniformly but does not shrink cell areas so that cells are stacked in a region  $K$  times smaller and remain as the original relative locations. The legalization step minimizes maximum on-chip temperature and TSV number through the position assignment of cells. The result of LST is a legalized 3D placement.

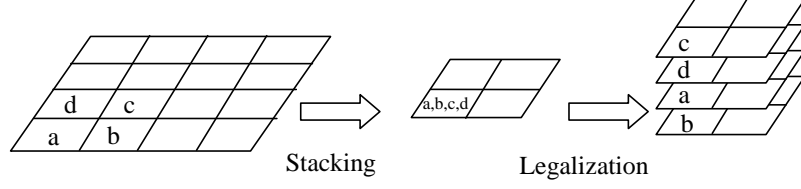


Figure 10. Local stacking transformation

#### 4.2.1.1 Local Stacking

For a  $K$ -die 3D design, if the original 2D placement area is  $S$ , then the area of a single die in the 3D design is  $S/K$ . During the local stacking step, we first shrink the width and height of the original 2D placement by a ratio of  $\sqrt{K}$ , so that the aspect ratio of the placement region is unchanged. Cell locations  $(x_i, y_i)$  of node  $v_i$  are also transformed to new locations  $(x'_i, y'_i)$ , where  $x'_i = x_i/\sqrt{K}$ , and  $y'_i = y_i/\sqrt{K}$ .

After such a transformation, the initial 2D placement is turned into a 2D placement of area  $S/K$  and has an average cell density of  $K$ , which later will be distributed to the  $K$  dies in the legalization step (next subsection). As shown in Figure 10, a group of neighboring nodes have overlaps after the stacking process.

#### 4.2.1.2 Tetris-Style Legalization

After stacking, we sort all the nodes by their  $x$ -coordinates in an increasing order. Starting from the leftmost node, we determine the location of nodes one by one in a way similar to the Tetris-style 2D legalization [47]. Each time we consider the leftmost legal position of all rows at all dies. We pick one position by minimizing the relocation cost  $R$ :

$$R = \alpha \cdot d + \beta \cdot v + \gamma \cdot t \quad (19)$$

where  $d$  is the node displacement from the stacking result,  $v$  is the TSV number and  $t$  is the thermal cost. Coefficients  $\alpha, \beta, \gamma$  are predetermined weights. The displacement  $d$  is related to the  $(x, y)$  locations of the nodes, and  $v$  and  $t$  are related to the die assignment of the nodes.

#### 4.2.1.3 Thermal Optimization

In this chapter temperature optimization is considered through the die assignment of the nodes. As mentioned in Section 1.1, under the current 3D integration technologies, the heat sink(s) are usually attached at the bottom (and/or top) side(s) of the 3D IC stack, with other boundaries being adiabatic. So the main dominant heat flow within the 3D IC stack is vertical towards the heat sink. The study in [26] shows that the  $z$  location of a node has a larger influence on the final temperature than the  $(x, y)$  location of the node. The lateral heat flow can be considered if the initial 2D placement is thermal-aware, so that hot nodes will be evenly distributed to avoid hot spots.

The full resistive thermal model mentioned in Section 3.3.1 is used for the final temperature evaluation. During the inner loops of the optimization process, a much simpler and faster thermal model [80] is used for the temperature optimization to speed up the placement process. Each tile stack is viewed as an independent thermal resistive chain, as shown in Figure 8. The maximum temperature of such a tile stack then can be expressed as the following:

$$T = \sum_{i=1}^K R_i \cdot \left( \sum_{j=i}^K P_j \right) + R_b \cdot \left( \sum_{i=1}^K P_i \right) = \sum_{i=1}^K P_i \cdot \left( \sum_{j=1}^i R_j + R_b \right) \quad (20)$$

Besides the reduced runtime, such a simple close-form equation can also provide a direct guide to the thermal-aware die assignment. Equation (20) tells us that the maximum temperature of a tile stack is the weighted sum of the power number at each die, while the weight of each die is the sum of the resistances below that die. The dies that are closer to the heat sink will have smaller weights.

The thermal cost  $t_{i,j}$  to assign node  $v_j$  to die  $i$  in equation (19) can be written as

$$t_{i,j} = p_j \cdot \left( \sum_{k=1}^i R_k + R_b \right) \quad (21)$$

This thermal cost of die assignment is used in equation (19) for legalization, and also used in Section 4.3 for die assignment refinement.

#### 4.2.2 Transformation through Folding

LST achieves short wirelength by stacking the neighboring nodes together. However, a great number of TSVs will be generated when the nodes of local nets are put on top of one another. If the target 3D integration technology only allows a limited TSV density, we need to use the transformations that generate fewer TSVs.

Folding-based transformation is to fold the original 2D placement like a piece of paper without cutting off any parts of the placement. The distance between any two nodes will not increase and the total wirelength is *guaranteed* to decrease. TSVs are only introduced to the nets crossing the folding lines (shown as the dashed lines in Figure 11). With an initial 2D placement of minimized wirelength, the number of such long nets should be fairly small, which implies that the connections between the folded regions should be limited, resulting in much fewer TSVs (compared to that of the LST

transformation, where many dense local connections cross different dies). Figure 11(a) shows one way of folding, named folding-2, by folding once at both  $x$  and  $y$  directions. Figure 11(b) shows another way of folding, named folding-4, by folding twice at both  $x$  and  $y$  directions. The folding results are legalized 3D placements, so no legalization step is necessary.

After folding-based transformations, only the lengths of the global nets that go across the folding lines (dotted lines in Figure 11) get reduced. Therefore, folding-based transformations cannot achieve as much wirelength reduction as LST. Furthermore, if we want to maintain the original aspect ratio, folding-based transformations are only applicable to the 3D design with an even number of dies.

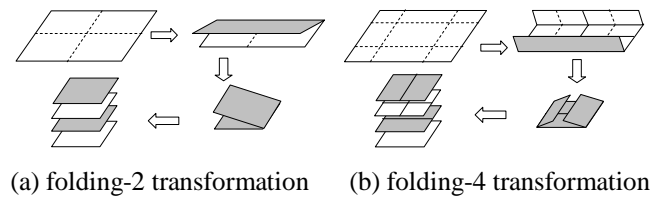


Figure 11. Two folding-based transformation schemes

### 4.2.3 Window-Based Stacking/Folding

As stated above (and as will be shown in Section 4.4), LST achieves the greatest wirelength reduction at the expense of large numbers of TSVs, while folding results in a much smaller TSV number but longer wirelength and possibly high TSV density along the folding lines.

An ideal 3D placement should have short wirelength with TSV density satisfying what the 3D integration technology can support. Moreover, we prefer an even TSV



density for routability issues. Therefore, we develop a window-based stacking/folding method for better TSV density control.

In this method we first divide the 2D placement into  $N \times N$  windows. Then we apply stacking or folding in every window. Each window can use different stacking/folding orders. Figure 12 shows the cases for  $N = 2$ . The 2D placement region is divided into  $2 \times 2$  windows (shown in solid lines). Each window is again divided into four squares (shown in dotted lines). The number in each square indicates the layer number of that square after stacking/folding. The four-die placements of each window are packed to form the final 3D placement.

Wirelength reduction occurs for the following reasons: the wirelength of the nets inside the same square is preserved; the wirelength of nets inside the same window is most likely reduced due to the effect of stacking/folding; and wirelength of nets that cross the different windows is reduced. Therefore the overall wirelength quality is improved.

Meanwhile, the TSVs are distributed evenly among different windows and can be reduced by choosing proper die assignments. TSVs are introduced by the nets that cross the boundaries between neighboring squares with different die numbers, and we call such a boundary between two neighboring squares a *transition*. Fewer transitions result in fewer TSVs. Intra-window transitions cannot be reduced because we need to distribute intra-window squares to different dies, so we focus on reducing inter-window transitions. Since the sequential die assignment in Figure 12(a) creates lots of transitions, we use another die assignment, as in Figure 12(b), called symmetric assignment to reduce the

amount of inter-window transitions to zero. So this layer assignment generates the smallest TSV number, while the wirelength is similar.

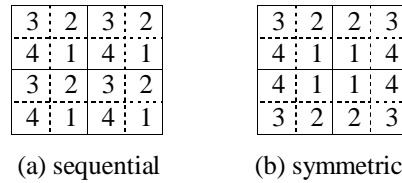


Figure 12. 2×2 windows with different die assignments

The wirelength vs. TSV number tradeoffs can be controlled by the number of windows, where the size of the windows is the granularity of node clusters that we would like to distribute into different dies.

### 4.3 Refinement: RCN Graph-Based Die Reassignment

During the 3D transformations discussed in the previous section, die assignment of nodes is based on simple heuristics. To further reduce the TSV number and the temperature, we design a novel die assignment algorithm to redistribute the nodes into different dies.

#### 4.3.1 Conflict-Net Graph

We extended a metal wire layer assignment algorithm designed in [19] for die assignment in 3D placement. For a given legalized 3D placement, a conflict-net (CN) graph is created, as shown in Figure 13, where both the cells (netlist nodes) and the vias (TSVs) are the nodes of the CN graph. One via node is assigned for each net. There are two types of edges, net edges and conflicting edges. Within each net, all cells are

connected to the via node by net edges in a star model. A conflict edge is created between cells that overlap with each other if they are placed on the same die.

We want to find a die assignment for each cell node in the graph so that the total cost, including edge costs and node costs, are minimized. Cost  $0$  is assigned to each net edge. If two cells connected by a conflicting edge are assigned to the same die, the cost of the conflicting edge is set to *infinity*; otherwise, the cost is set to  $0$ . The cost of a via node is the height of that via, which represents the total TSV number in that net. The heights of the vias are determined by the dies of the cells that connect with them. The cost of a cell node  $v_j$  is the thermal cost  $t_{i,j}$  of assigning  $v_j$  to die  $i$ , as in equation (21). The cost of a path is the sum of the edge costs and the node costs along that path.

The resulting graph is a directional acyclic graph. A dynamic programming optimization method can be used to find the optimal solution for each induced sub-tree of the graph in linear time. An algorithm that constructs a sequence of a maximal induced sub-tree from the CN graph is then used to cover a large portion of the original graph. It turns out that the average node number of the induced sub-trees can be as much as 40% to 50% of the total nodes in the graph. After the iterative optimization of the sub-trees, we can achieve a globally optimized solution. Please refer to [19] for the detailed algorithm to solve the die assignment problem with a CN graph.

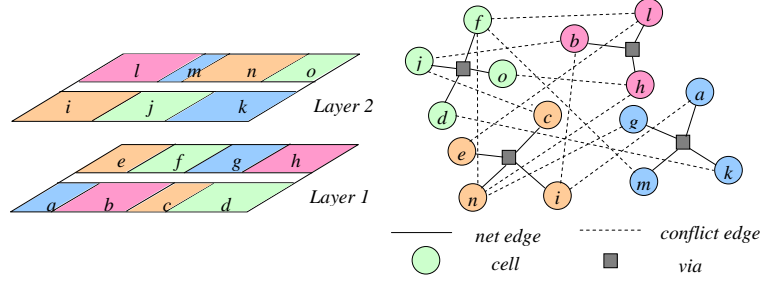


Figure 13. Relaxed conflict-net graph

### 4.3.2 Relaxed Non-Overlap Constraint

To further reduce the TSV number and the maximum on-chip temperature, the non-overlap constraints can be relaxed so that a small amount of overlap  $r$  is allowed in exchange for more freedom in layer reassignment of the cells.

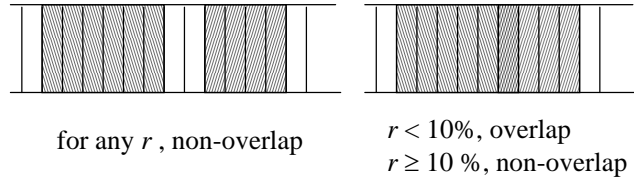


Figure 14. Relaxation of non-overlap constraint

The relaxed non-overlap is defined as follows.

$$\text{Overlap}(i, j) = \begin{cases} \text{false}, & \text{if } \frac{o(i, j)}{s(i) + s(j)} \leq r \\ \text{true}, & \text{if } \frac{o(i, j)}{s(i) + s(j)} > r \end{cases} \quad (22)$$

where  $o(i, j)$  is the overlapped area between cell  $v_i$  and  $v_j$ ,  $s(i) = w_i h_i$  is the area of cell  $v_i$ . The tolerant overlap ratio  $r$  is a positive real number between 0 and 0.5. This is illustrated in Figure 14.

However, with the relaxed non-overlap constraint, the die assignment result is no longer a legalized 3D placement, and another round of legalization is needed to eliminate these overlaps.

## 4.4 Experimental Results

We implemented the transformation-based 3D placer, named T3Place, featuring local stacking 3D transformation and RCN graph-based die reassignment. We also implemented two folding-based 3D transformation methods and window-based stacking/folding methods. In our experiments, we use mPL5 [15] and its detailed placer [27] to generate the initial legalized wirelength-driven 2D global placement results. After 3D transformation and refinement, we will apply the 2D detailed placer on each die to further reduce wirelength. The wirelength and TSV numbers are computed based on the models discussed in 2.4.1.

Our experiments are performed on 2.4GHz Pentium IV machines under Red Hat enterprise 3.0. We test T3Place using the popular 2D standard cell placement benchmark set, IBM-PLACE from [89]. A four-die 3D IC structure is assumed for all test cases.

In Table 2, we first compared the results of the wirelength-driven T3Place with the 2D placement results, where the wirelength (WL) is normalized with mPL5 as the baseline, and the number of TSVs (#TSV) is normalized with LST ( $r=10\%$ ) as the baseline. Compared to mPL5, LST ( $r=10\%, 20\%$ ) can reduce the wirelength by about  $2\times$  with four dies. Moreover, we compared LST transformations with folding-2 and folding-4, and the results are shown in Table 3. For folding-based transformations, no placement refinement is done because the TSV number is already small enough. Compared to LST,

folding-2 can reduce the TSV number by over 12.5× but with only 11% wirelength reduction over the 2D placement. With more folding lines, folding-4 can achieve a 15% wirelength reduction over the 2D placement with 7× TSV reduction compared to LST. At last, results of 8x8 window-based stacking demonstrate that our method is adaptive to different manufacturing abilities for TSV density.

Table 2. Benchmark characteristics and results of mPL6 and LST transformations

circuit	cell #	net #	mPL5	LST ( $r=10\%$ )		LST ( $r=20\%$ )	
				WL	#TSV	WL	#TSV
ibm01	12282	11507	5.19E+06	2.52E+06	18519	2.68E+06	14102
ibm03	22207	21621	1.37E+07	6.62E+06	30434	7.29E+06	21406
ibm04	26633	26163	1.67E+07	8.45E+06	37414	9.20E+06	26871
ibm06	32185	33354	2.20E+07	1.10E+07	50139	1.52E+07	32939
ibm07	45135	44394	3.73E+07	1.83E+07	65093	2.07E+07	44715
ibm08	50977	47944	3.94E+07	1.98E+07	70317	2.13E+07	49844
ibm09	51746	50393	3.46E+07	1.72E+07	72787	1.95E+07	50755
ibm13	81508	83806	6.58E+07	3.24E+07	121135	3.60E+07	85103
ibm15	158244	161196	1.65E+08	8.26E+07	246509	9.11E+07	176018
ibm18	210323	200565	2.43E+08	1.26E+08	297771	1.34E+08	208564
Avg.			<b>1.00</b>	<b>0.50</b>	<b>1.00</b>	<b>0.56</b>	<b>0.71</b>

Table 3. Results of folding and window-based transformations

circuit	Folding-2		Folding-4		LST (8x8 win)	
	WL	#TSV	WL	#TSV	WL	#TSV
ibm01	4.61E+06	1671	4.55E+06	2476	3.53E+06	6688
ibm03	1.14E+07	4125	1.11E+07	5909	8.36E+06	12318
ibm04	1.55E+07	2940	1.43E+07	6388	1.10E+07	15315
ibm06	2.02E+07	4116	1.83E+07	9077	1.44E+07	19315
ibm07	3.18E+07	5932	3.10E+07	8755	2.37E+07	25021
ibm08	3.48E+07	5801	3.28E+07	10181	2.56E+07	25205
ibm09	3.19E+07	4540	2.93E+07	8257	2.34E+07	23836
ibm13	6.03E+07	7696	5.85E+07	13071	4.50E+07	42568
ibm15	1.45E+08	15128	1.38E+08	23662	1.14E+08	72956
ibm18	2.24E+08	12077	2.08E+08	28287	1.74E+08	83380
Avg.	<b>0.89</b>	<b>0.08</b>	<b>0.85</b>	<b>0.14</b>	<b>0.67</b>	<b>0.36</b>

We also compared the thermal-aware T3Place with the wirelength-driven T3Place. The results are shown in Table 4. In both schemes, the LST overlap tolerance  $r$  is set to

10%. We report the temperature (Temp.) as the difference between the maximum on-chip temperature and the heat sink temperature. Compared to the wirelength-driven LST, the thermal-aware LST can reduce the maximum on-chip temperature by 37% on average with 6% more TSVs and 8% longer wirelength.

Table 4. Thermal-aware T3Place results

circuit	Thermal-aware LST (r = 10%)	Wirelength-driven LST (r = 10%)		
	Temp. ( °C)	WL	#TSV	Temp. ( °C)
ibm01	276.5	2.81E+06	19020	159.8
ibm03	196.7	7.13E+06	31780	121.6
ibm04	159.6	9.11E+06	40219	96.0
ibm06	160.4	1.23E+07	50576	103.5
ibm07	107.5	2.01E+07	69111	66.4
ibm08	97.7	2.05E+07	75397	63.2
ibm09	96.1	1.94E+07	78102	60.6
ibm13	249.3	3.47E+07	127520	156.2
ibm15	136.5	8.58E+07	260681	90.1
ibm18	89.4	1.31E+08	332012	58.7
<b>Avg.</b>	<b>1.00</b>	<b>0.54</b>	<b>1.06</b>	<b>0.63</b>

Furthermore, we compared the wirelength-driven T3Place with the force-directed 3D placement algorithm [43], as shown in Table 5.. To compare results, we also scaled the chip to 2cm×2cm and set the height of the TSVs to 20 μm as suggested by the authors of [43]. Compared to [43], T3Place shows significant wirelength reduction (over 5×), and we have been in contact with the authors of [43] to share the results and experimental setting. Because we use different thermal models for 3D IC, the temperature values are not comparable.

Table 5. Comparisons with existing 3D placement in [43]

circuits	Scaled WL	#TSV	Total WL under metric of [19]	Total WL in [19]
ibm01	13.3	18519	13.7	63.8

<b>ibm03</b>	29.6	30434	30.2	115.9
<b>ibm04</b>	32.5	37414	33.2	144.5
<b>ibm06</b>	43.3	50139	44.3	183.2
<b>ibm07</b>	53.1	65093	54.4	277.7
<b>ibm08</b>	54.2	70317	55.6	278.9
<b>ibm09</b>	46.3	72787	47.7	252.5

## 4.5 Conclusions

In this chapter we presented the transformation-based approaches as our early attempts to solve the 3D placement problem. We compared different transformation approaches, including LST, folding-2, folding-4 and window-based stacking/folding. Compared to a four-die 3D design with a 2D implementation, LST achieves a 2× wirelength reduction by taking advantage of a significant number of TSVs. The folding transformations are able to reduce the number of TSVs by 7× to 12.5× and still have an 11% to 15% advantage in wirelength over the 2D implementation. The window-based transformations provide trade-offs between wirelength and the number of TSVs. In addition, we developed a novel die assignment refinement through the RCN graph-based optimization; this can reduce TSV number and temperature significantly.



## Chapter 5

### Efficient Gradient Computation for Density-Constrained

#### Analytical Placement Approaches

After our early attempt to solve the 3D placement problem by the transformation-based approaches, as discussed in the previous chapter, we became interested in designing novel 3D placement algorithms ground-up in order to achieve better placement quality and enable more complex constraints (e.g., thermal constraints). 3D placement algorithms have to handle more complex non-overlapping constraints in the presence of multiple dies than do 2D placement algorithms. Therefore, we re-examine the non-overlapping constraints in 2D placement with insights, which will be applied as basic elements for the analytical 3D placement algorithm in Chapter 6 and the TSV distribution constraints in Chapter 7.

Recent analytical global placers use regional area density density constraints to approximate non-overlapping constraints (see Section 2.4.2), and show very successful results in both quality and scalability [39][16][20]. Differentiability of both the objective functions and constraint functions is usually required by analytical solvers. But the density function is normally not smooth; thus several smoothing techniques have been developed and implemented to overcome this problem, including: (i) the bell-shaped function [50][20] to replace the rectangle-shaped nodes with differential bell-shaped nodes; (ii) the smoothing operator defined by the Helmholtz equation [15]; (iii) the

Gaussian smoothing [20] for the density of fixed nodes; (iv) the Poisson equation [39] to transform area distribution to some smoothed potential.

In this chapter we are interested in the smoothing techniques (ii), (iii) and (iv) listed above, which we call *global smoothing* techniques because the smoothed density of a single bin is correlated globally with the original density of every bin. Global smoothing techniques were used by the top placers in the ISPD06 placement contest [64], and the contest results indicate that these techniques are effective in achieving high-quality solutions. However, until recently, these techniques did not completely conform to the standard nonlinear programming framework. The method in NTUplace3 [20] did not use Gaussian smoothing for movable nodes, but only for fixed nodes; The method in Kraftwerk2 [75] used the smoothed potential as the basis for a force-directed method, but it does not follow a standard nonlinear programming framework; The method in mPL5 [15] generalized the force-directed method and used a nonlinear programming formulation and solution technique based on the Uzawa algorithm [4], but it could only use a simple approximation of the gradient computation for the smoothed density function.

To adopt these global smoothing techniques into a nonlinear programming framework, a fundamental difficulty arises because of the high complexity of gradient computation of the density penalty function. Unlike the bell-shaped function smoothing technique where the gradient of the density penalty function can be written down explicitly, the global smoothing techniques do not seem to have any simple analytical

form and may require a large amount of numerical computation. This difficulty was the motivation for our work, which has resulted in the following contributions:

- We observed the common property of the global smoothing techniques (ii), (iii) and (iv), which makes our work extensible to handling a large class of smoothing techniques.
- We derived an equivalent expression for the gradient of the density penalty function, which leads to highly efficient numerical computation and reduces the time complexity by a factor of  $n$ , compared to a naïve computation.
- We used our efficient density gradient method in a nonlinear programming framework. We consider this to be a contribution because it is the first time that the density-constrained placement problem with global smoothing technique can be solved exactly in the general nonlinear programming framework. Moreover, we found that the resulting placement method supersedes the force-directed placement methods [39][15].
- In particular, we applied our gradient computation to the augmented Lagrangian method in a multilevel placement framework, and tested this on the IBM-HB+ benchmark [68]. The application leads to a 15% shorter wirelength than mPL6 [16]. It also leads to a 3% wirelength improvement, on average, in a modified ISPD'05 [66] and ISPD'06 [64] benchmark with movable macros.

This chapter is organized as follows. Section 5.1 introduces the problem formulation; Section 5.2 describes the class of smoothing techniques we are concerned with; Section 5.3 defines the density penalty function under two kinds of nonlinear programming

methods; Section 5.4 derives an equivalent expression for the gradient of density penalty function that leads to highly efficient computation; Section 5.5 discusses practical implementation in a global placer; Section 5.6 presents our experimental results; and finally, conclusions and future work are presented in Section 5.7.

## 5.1 Problem Formulation

We begin with a 2D placement problem for wirelength minimization under given regional area density constraints. The global placement problem is formulated as,

$$\begin{aligned} & \text{minimize} && \text{WL}(\bar{x}, \bar{y}) \\ & \text{subject to} && D_{m,n}(\bar{x}, \bar{y}) \leq C_{m,n} \quad \text{for} \quad \begin{array}{l} 1 \leq m \leq M \\ 1 \leq n \leq N \end{array} \end{aligned} \quad (23)$$

where  $D_{m,n}(\bar{x}, \bar{y}) = \sum_{v_i \in V} \text{Overlap}(\text{bin}_{m,n}, v_i)$  and  $C_{m,n} = w_{\text{bin}} \cdot h_{\text{bin}}$ . The regional area density

constraints are special cases of Section 2.4.2 with a single die ( $K = 1$ ).

Analytical placement algorithms require that all the functions are differentiable in the formulation. The wirelength objective can be approximated by one of the following differentiable versions, including quadratic wirelength [75],  $L_p$ -norm wirelength [53][15], log-sum-exp wirelength [67], CHKS wirelength [59], and weighted-average wirelength [48]. The remainder of this section will use the log-sum-exp wirelength:

$$\begin{aligned} \text{WL}_{\text{LSE}}(x, y) = \eta \cdot \sum_{e \in E} & \left( \log \sum_{v_i \in e} \exp\left(+\frac{x_i}{\eta}\right) + \log \sum_{v_i \in e} \exp\left(-\frac{x_i}{\eta}\right) \right. \\ & \left. + \log \sum_{v_i \in e} \exp\left(+\frac{y_i}{\eta}\right) + \log \sum_{v_i \in e} \exp\left(-\frac{y_i}{\eta}\right) \right) \end{aligned} \quad (24)$$

This approximated wirelength function is more accurate with a smaller  $\eta$ . For numerical stability, the parameter  $\eta$  is set to  $0.01 \times \max\{W_{\text{die}}, H_{\text{die}}\}$ .

According to [16], the inequality constraints can be transformed to equality constraints by introducing *filler nodes*. Thus the formulation becomes

$$\begin{aligned} & \text{minimize} && \text{WL}_{\text{LSE}}(\bar{x}, \bar{y}) \\ & \text{subject to} && D_{m,n}(\bar{x}, \bar{y}) = C_{m,n} \quad \text{for} \quad \begin{array}{l} 1 \leq m \leq M \\ 1 \leq n \leq N \end{array} \end{aligned} \quad (25)$$

These equality constraints can be formulated in continuous form by pushing the resolution  $M \times N$  to infinity:

$$\begin{aligned} & \text{minimize} && \text{WL}_{\text{LSE}}(\bar{x}, \bar{y}) \\ & \text{subject to} && (D(\bar{x}, \bar{y}))(u, v) = 1 \quad \text{for} \quad \begin{array}{l} u \in [0, W_{\text{die}}] \\ v \in [0, H_{\text{die}}] \end{array} \end{aligned} \quad (26)$$

where the area density  $(D(\bar{x}, \bar{y}))(u, v) = \sum_{v_i \in V} (D_i(x_i, y_i))(u, v)$  with

$$(D_i(x_i, y_i))(u, v) = \begin{cases} 1 & \left( \text{for} \quad \begin{array}{l} u \in [x_i - w_i/2, x_i + w_i/2] \\ v \in [y_i - h_i/2, y_i + h_i/2] \end{array} \right) \\ 0 & \text{(otherwise)} \end{cases} \quad (27)$$

as the area density contribution of node  $v_i$ .

This area density function  $(D(\bar{x}, \bar{y}))$  is still not differentiable. After replacing it with a smoothed density function  $(\hat{D}(\bar{x}, \bar{y}))$ , the final relaxed problem being solved is

$$\begin{aligned} & \text{minimize} && \text{WL}_{\text{LSE}}(\bar{x}, \bar{y}) \\ & \text{subject to} && (\hat{D}(\bar{x}, \bar{y}))(u, v) = 1 \quad \text{for} \quad \begin{array}{l} u \in [0, W_{\text{die}}] \\ v \in [0, H_{\text{die}}] \end{array} \end{aligned} \quad (28)$$

## 5.2 General Smoothed Density

Before introducing the general smoothed density, several density smoothing techniques will be reviewed first.

### 5.2.1 Helmholtz Smoothing

The smoothed density  $(\widehat{D}_H(\bar{x}, \bar{y}))(u, v)$  is defined as the solution of the Helmholtz equation,

$$\begin{cases} (\partial^2/\partial u^2 + \partial^2/\partial v^2 - \varepsilon)\widehat{D}_H(u, v) = -D(u, v) \\ \partial\widehat{D}_H(u, 0)/\partial u = \partial\widehat{D}_H(u, H_{\text{die}})/\partial u = 0 \\ \partial\widehat{D}_H(0, v)/\partial v = \partial\widehat{D}_H(W_{\text{die}}, v)/\partial v = 0 \end{cases} \quad (29)$$

The solution can be written down explicitly with Green's function [72].

$$(\widehat{D}_H(\bar{x}, \bar{y}))(u, v) = -\int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (D(\bar{x}, \bar{y}))(u', v') G_H(u, v, u', v') du' dv' \quad (30)$$

where  $G_H(u, v, u', v') = \frac{1}{W_{\text{die}} H_{\text{die}}} \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \frac{c_i c_j \cos(p_i u) \cos(q_j v) \cos(p_i u') \cos(q_j v')}{p_i^2 + q_j^2 + \varepsilon}$  with

constants  $c_i, c_j, p_i, q_j$  that only depend on  $i, j$ .

### 5.2.2 Poisson Smoothing

The smoothed density  $(\widehat{D}_H(\bar{x}, \bar{y}))(u, v)$  is a special case of Helmholtz smoothing with  $\varepsilon = 0$ . But the equation (29) with  $\varepsilon = 0$  is singular most of the time, except when

$\int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (D(\bar{x}, \bar{y}))(u, v) dv du = 0$ . This can be achieved by smoothing

$((D(\bar{x}, \bar{y}))(u, v) - 1)$  instead of smoothing  $(D(\bar{x}, \bar{y}))(u, v)$ , because the integral of the former over  $[0, W_{\text{die}}] \times [0, H_{\text{die}}]$  is always zero if the filler nodes are added properly.

### 5.2.3 Gaussian Smoothing

The smoothed density  $(\hat{D}_G(\bar{x}, \bar{y}))(u, v)$  is defined as the convolution between the Gaussian function and the original density  $(D(\bar{x}, \bar{y}))(u, v)$ , which is

$$(\hat{D}_G(\bar{x}, \bar{y}))(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (D(\bar{x}, \bar{y}))(u', v') G_G(u, v, u', v') du' dv' \quad (31)$$

where  $G_G(u, v, u', v') = \frac{1}{2\pi\sigma^2} \exp(-\frac{(u-u')^2 + (v-v')^2}{2\sigma^2})$  with  $\sigma$  as a smoothing parameter.

### 5.2.4 General Global Smoothing

We observed that the three smoothing techniques described above can be generalized into the following density-smoothing operation:

$$(\hat{D}(\bar{x}, \bar{y}))(u, v) = - \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (D(\bar{x}, \bar{y}))(u', v') G(u, v, u', v') du' dv' \quad (32)$$

where the symmetry property  $G(u, v, u', v') = G(u', v', u, v)$  holds. Intuitively,  $G(u, v, u', v')$  represents the amount of smoothed density at point  $(u, v)$  created by a unit of the original density at point  $(u', v')$ . We call this class of smoothing operations *global smoothing*, because  $G(u, v, u', v')$  is usually non-zero for every pair of  $(u, v)$  and  $(u', v')$ , which indicates that the influence of the original density to the smoothed density is global.

The discussions and results in the remainder of this chapter will be based on this class of general smoothing operations. Please note that the lower limit and upper limit of

the double integral are bounded in equation (32). This is only for the convenience of expression, and the following discussion is simple to extend for the unbounded cases.

Using this smoothing operation, we can transform the density constraints to the smoothed density constraints. The arc  $\hat{\cdot}$  is added on top of a density function to represent that it is a smoothed density.

### 5.3 Density Penalty Function

The problem specified in (28) is a constrained nonlinear programming problem. Its solution typically involves solving a sequence of unconstrained problems. Two commonly used methods are the quadratic penalty method and the augmented Lagrangian method, where the unconstrained problems optimize a combination of wirelength and penalty functions on the density constraints. The details will be described in the following subsections.

#### 5.3.1 Quadratic Penalty Method

The quadratic penalty function for problem (28) is

$$Q(\bar{x}, \bar{y}; \mu) = \text{WL}_{\text{LSE}}(\bar{x}, \bar{y}) + \frac{\mu}{2} \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} ((\hat{D}(\bar{x}, \bar{y}))(u, v) - 1)^2 dudv \quad (33)$$

We consider  $\hat{D}(\bar{x}, \bar{y})(*, *)$  to be a vector in the vector space  $C([0, W_{\text{die}}] \times [0, H_{\text{die}}])$ , which consists of all the continuous two-variable function defined in  $[0, W_{\text{die}}] \times [0, H_{\text{die}}]$ . In this vector space, for any two functions  $g_1(*, *)$ ,  $g_2(*, *)$ , we can define their inner product as follows:

$$g_1(*, *) \odot g_2(*, *) = \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} g_1(u, v) g_2(u, v) dudv \quad (34)$$



The norm of  $g(*,*)$  is defined as  $\|g(*,*)\| = g(*,*) \odot g(*,*)$ . With these definitions, the basic concepts, like the limit and the convergence, can also be defined, and the convergence of the quadratic penalty method in vector space can be proved seamlessly [62].

Under the notion of vector space, the quadratic penalty function is written as:

$$Q(\bar{x}, \bar{y}; \mu) = \mathbf{WL}_{\text{LSE}}(\bar{x}, \bar{y}) + P_Q(\bar{x}, \bar{y}; \mu) \quad (35)$$

where 
$$P_Q(\bar{x}, \bar{y}; \mu) = \frac{\mu}{2} \|\hat{D}(\bar{x}, \bar{y}) - \hat{I}\|^2 \quad (36)$$

And the gradient is expressed as:

$$\nabla Q(\bar{x}, \bar{y}; \mu) = \nabla \mathbf{WL}_{\text{LSE}}(\bar{x}, \bar{y}) + \nabla P_Q(\bar{x}, \bar{y}; \mu) \quad (37)$$

where 
$$\nabla P_Q(\bar{x}, \bar{y}; \mu) = \mu(\hat{D}(\bar{x}, \bar{y}) - 1) \odot \nabla \hat{D}(\bar{x}, \bar{y}) \quad (38)$$

with the scalar  $\mu$  as the quadratic penalty parameter.

We call  $P_Q(\bar{x}, \bar{y}; \mu)$  the *density penalty function* for the quadratic penalty method, and call  $\nabla P_Q(\bar{x}, \bar{y}; \mu)$  the *gradient of the density penalty function*.

The algorithmic framework of the quadratic penalty method [69] is given in Algorithm 1.

<p><b>Given</b>  Penalty factor <math>\mu^{(0)} \geq 0</math>,  tolerance <math>\{\tau^{(k)}\}</math> such that <math>\tau^{(k)} \rightarrow 0</math>,  and a starting point <math>(\bar{x}^{(0)}, \bar{y}^{(0)})</math>;  <b>for</b> <math>k = 0, 1, 2, \dots</math>  Starting at <math>(\bar{x}^{(k)}, \bar{y}^{(k)})</math>,  find an approximate minimizer <math>(\bar{x}^{(k+1)}, \bar{y}^{(k+1)})</math>,  such that <math>\ \nabla Q(\bar{x}^{(k+1)}, \bar{y}^{(k+1)}; \mu^{(k)})\  \leq \tau^{(k)}</math>;  <b>if</b> final convergence test is satisfied  <b>stop</b> with approximate solution <math>(\bar{x}^{(k+1)}, \bar{y}^{(k+1)})</math>;  <b>end if</b>  Choose new penalty parameter <math>\mu^{(k+1)} &gt; \mu^{(k)}</math>;  <b>end for</b></p>
---

Algorithm 1. Quadratic penalty method

### 5.3.2 Augmented Lagrangian Method

Similar to the quadratic penalty method in vector space, the augmented Lagrangian function for problem (28) can be written as:

$$L_A(\bar{x}, \bar{y}, \lambda; \mu) = \text{WL}_{\text{LSE}}(\bar{x}, \bar{y}) + P_A(\bar{x}, \bar{y}, \lambda; \mu) \quad (39)$$

where

$$P_A(\bar{x}, \bar{y}, \lambda; \mu) = \lambda \odot (\widehat{D}(\bar{x}, \bar{y}) - 1) + P_Q(\bar{x}, \bar{y}; \mu) \quad (40)$$

And its gradient is

$$\nabla L_A(\bar{x}, \bar{y}, \lambda; \mu) = \nabla \text{WL}_{\text{LSE}}(\bar{x}, \bar{y}) + \nabla P_A(\bar{x}, \bar{y}, \lambda; \mu) \quad (41)$$

where

$$\begin{aligned} \nabla P_A(\bar{x}, \bar{y}, \lambda; \mu) &= \lambda \odot \nabla \widehat{D}(\bar{x}, \bar{y}) + \nabla P_Q(\bar{x}, \bar{y}; \mu) \\ &= (\lambda + \mu(\widehat{D}(\bar{x}, \bar{y}) - \widehat{I})) \odot \nabla \widehat{D}(\bar{x}, \bar{y}) \end{aligned} \quad (42)$$

with a scalar  $\mu$  as the quadratic penalty parameter and  $\lambda(*, *) \in L^2([0, W_{\text{die}}] \times [0, H_{\text{die}}])$ , a twice differentiable function, as the Lagrangian multiplier.

Similarly, we call  $P_A(\bar{x}, \bar{y}, \lambda; \mu)$  the density penalty function for the augmented Lagrangian method, and call  $\nabla P_A(\bar{x}, \bar{y}, \lambda; \mu)$  the gradient of the density penalty function. The algorithmic framework of the augmented Lagrangian method [69] is given in Algorithm 2.

```

Given
   $\mu^{(0)} > 0$ , tolerance  $\tau^{(0)} > 0$ ,
  a starting point  $(\bar{x}^{(0)}, \bar{y}^{(0)})$  and  $\lambda^{(0)}$ ;
for  $k = 0, 1, 2, \dots$ 
  Starting at  $(\bar{x}^{(k)}, \bar{y}^{(k)})$ ,
  Find an approximate minimizer  $(\bar{x}^{(k+1)}, \bar{y}^{(k+1)})$ 
    such that  $\|\nabla L_A(\bar{x}^{(k+1)}, \bar{y}^{(k+1)}, \lambda^{(k)}, \mu^{(k)})\| \leq \tau^{(k)}$ ;
  if a convergence test is satisfied
    Stop with approximate solution  $(\bar{x}^{(k+1)}, \bar{y}^{(k+1)})$ ;
  end if
  Update  $\lambda^{(k+1)} \leftarrow \lambda^{(k)} + \mu^{(k)} (\widehat{D}(\bar{x}^{(k+1)}, \bar{y}^{(k+1)}) - \widehat{I})$ ;
  Choose new penalty parameter  $\mu^{(k+1)} \geq \mu^{(k)}$ ;
  Select tolerance  $\tau^{(k+1)}$ ;
end for

```

Algorithm 2. Augmented Lagrangian method

## 5.4 Efficient Gradient Computation

The gradient of the density penalty function of either the quadratic penalty method in equation (38), or the augmented Lagrangian method in equation (42) has the common form  $g \odot \nabla \widehat{D}(\bar{x}, \bar{y})$ , where the function  $g$  equals  $\mu(\widehat{D}(\bar{x}, \bar{y}) - \widehat{I})$  in the quadratic penalty method and equals  $\lambda + \mu(\widehat{D}(\bar{x}, \bar{y}) - \widehat{I})$  in the augmented Lagrangian method. It is a vector with  $2|V|$  elements, where the elements include  $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$  and  $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k$  for  $k = 1, 2, \dots, |V|$ .

The computation of this gradient is required hundreds to thousands of times when solving the constrained nonlinear programming problems. However, the computation is not trivial. We will explain why a naïve method is not practical and how we efficiently solve this problem. In the following sections, we assume the run time for the smoothing operation is  $T(|V|)$ , and it is superlinear (of higher order than  $O(|V|)$ ), as we need to consider every node for the smoothing operation.

#### 5.4.1 Naïve Computation

To compute  $g \odot \nabla \widehat{D}(\bar{x}, \bar{y})$ , a naïve method has to compute the components  $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$  and  $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k$  one by one, and then expand the inner product for each component:

$$g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} = \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} g(u, v) \frac{\partial (\widehat{D}(\bar{x}, \bar{y}))(u, v)}{\partial x_k} dudv \quad (43)$$

$$g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial y_k} = \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} g(u, v) \frac{\partial (\widehat{D}(\bar{x}, \bar{y}))(u, v)}{\partial y_k} dudv \quad (44)$$

By discretizing the placement region into  $M \times N$  bins, the double integrals are computed through double summation, and the partial derivatives are computed through a certain finite difference scheme.

The detailed description is given in Algorithm 3. In the loop between lines 04 to 11, there are two time-consuming parts. Line 07 consumes  $T(|V|)$  time in each loop, and the computation of  $\{\widehat{D}'_{ij}\}$  cannot be reused. Lines 06 and 08 consume  $O(MN) = O(|V|)$  time,

assuming the bin number is of the same magnitude as the number of nodes. Therefore, the time complexity is  $O(|V|)(T(|V|) + O(|V|)) = O(|V|)T(|V|)$  for this naïve computation.

**Input:**  $(\bar{x}, \bar{y}), \{g_{ij}\}$   
**Output:**  $g \odot \nabla \widehat{D}(\bar{x}, \bar{y})$   
**Algorithm:**  
01:  $\{D_{ij}\} \leftarrow \text{compute\_density}(\bar{x}, \bar{y});$   
02:  $\{\widehat{D}_{ij}\} \leftarrow \text{smooth}(\{D_{ij}\});$   
03: Select small  $\Delta x, \Delta y;$   
04: **for**  $k=1, 2, 3, \dots, n$   
05:  $x_k \leftarrow x_k + \Delta x;$   
06:  $\{D'_{ij}\} \leftarrow \text{compute\_density}(\bar{x}, \bar{y});$   
07:  $\{\widehat{D}'_{ij}\} \leftarrow \text{smooth}(\{D'_{ij}\});$   
08:  $g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} \leftarrow \sum_{i=1}^M \sum_{j=1}^N g_{ij} \frac{\widehat{D}'_{ij} - \widehat{D}_{ij}}{\Delta x} w_{\text{bin}} h_{\text{bin}};$   
09:  $x_k \leftarrow x_k - \Delta x;$   
10: Compute  $g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial y_k}$  as in lines 05-09;  
11: **end for**

Algorithm 3. Naïve gradient computation for the density penalty function

### 5.4.2 Efficient Computation

To avoid the time-consuming part of the naïve computation, the equivalent expressions of the inner products  $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$  and  $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k$  are derived for efficient computation, and are given in the following theorem.

**Theorem 5.1.** Let  $D(\bar{x}, \bar{y})$  and  $\widehat{D}(\bar{x}, \bar{y})$  be the density and smoothed density defined in Section 5.1, respectively, then for any function  $g \in L^2([0, a] \times [0, b])$ , we have

$$g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} = \int_{y_k - h_k/2}^{y_k + h_k/2} \left( \widehat{g}(x_k + \frac{w_k}{2}, v) - \widehat{g}(x_k - \frac{w_k}{2}, v) \right) dv \quad (45)$$

$$g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial y_k} = \int_{x_k - w_k/2}^{x_k + w_k/2} \left( \widehat{g}(u, y_k + \frac{h_k}{2}) - \widehat{g}(u, y_k - \frac{h_k}{2}) \right) du \quad (46)$$

**Proof.** Assume the function  $G(u, v, u', v')$  for the smoothing operation is continuous and differentiable over the region  $[0, W_{\text{die}}] \times [0, H_{\text{die}}]$ . Since the smoothing operation is linear,  $\widehat{D}(\bar{x}, \bar{y})$  can be decomposed to  $\sum_{k=1}^n \widehat{D}_k(x_k, y_k)$ , where  $\widehat{D}_k$  is the smoothed version of  $D_k$  defined in equation (27). Thus,

$$\frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} = \frac{\partial \widehat{D}_k(x_k, y_k)}{\partial x_k} \quad (47)$$

$$= \frac{\partial}{\partial x_k} \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (D_k(x_k, y_k))(u', v') G(u, v, u', v') du' dv' \quad (48)$$

$$= \frac{\partial}{\partial x_k} \int_{x_k - w_k/2}^{x_k + w_k/2} \int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, u', v') du' dv' \quad (49)$$

$$= \int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, x_k + w_k/2, v') dv' - \int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, x_k - w_k/2, v') dv' \quad (50)$$

Step (47) only keeps the smoothed density of node  $v_k$ , since the partial derivative of other smoothed densities with respect to  $x_k$  is zero. Step (48) expands  $\widehat{D}_k(x_k, y_k)$  that is defined in Section 5.2. Step (49) drops the integral region where  $D_k(x_k, y_k)$  is zero. Step (50) is derived according to the Leibniz Integral Rule. Therefore,

$$g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k = \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} g(u, v) \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} dudv \quad (51)$$

$$= \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} g(u, v) \left( \int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, x_k + w_k/2, v') dv' - \int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, x_k - w_k/2, v') dv' \right) dudv \quad (52)$$

$$= \int_{y_k - h_k/2}^{y_k + h_k/2} \left( \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} g(u, v) G(u, v, x_k + w_k/2, v') dudv - \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} g(u, v) G(u, v, x_k - w_k/2, v') dudv \right) dv' \quad (53)$$

$$= \int_{y_k - h_k/2}^{y_k + h_k/2} \left( \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} g(u, v) G(x_k + w_k/2, v', u, v) dudv - \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} g(u, v) G(x_k - w_k/2, v', u, v) dudv \right) dv' \quad (54)$$

$$= \int_{y_k - h_k/2}^{y_k + h_k/2} \left( \widehat{g}(x_k + w_k/2, v') - \widehat{g}(x_k - w_k/2, v') \right) dv' \quad (55)$$

Step (51) expands the inner product defined in Section 5.3.1; Step (52) substitutes  $\partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$  by the previous computation; Step (53) changes the order of integrals; Step (54) exchanges the variables of function  $G$  because of its symmetric property defined in Section 5.2.4; Step (55) applies the definition of smoothed density, as in Section 5.2.4.

In the same way, we can also derive the expression (46) for  $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k$ .

■

The key insight of equation (45) (or (46)) is that the integral of the product  $g(u, v) \odot (\partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k)(u, v)$  over the placement region is equivalent to the difference of the integrals of  $\widehat{g}(u, v)$  along a pair of opposing edges on the module boundary. The naive computation has to compute  $\partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$  for each  $k$ , but with the equation (45), we only need to compute  $\widehat{g}(u, v)$  once and reuse it for all the nodes. Thus, we are able to give a highly efficient computation of  $g \odot \nabla \widehat{D}(\bar{x}, \bar{y})$  as in Algorithm 4. The function

`interpolate({gij},(x,y))` computes  $g(x,y)$  at any  $(x,y)$  by bilinear interpolation. And the numerical integration is computed by the function `integrate({gij},(x1,y1),(x2,y2))` for equations (45) and (46).



<p><b>Input:</b> <math>(\bar{x}, \bar{y}), \{g_{ij}\}</math></p> <p><b>Output:</b> <math>g \odot \nabla \widehat{D}(\bar{x}, \bar{y})</math></p> <p><b>Algorithm:</b></p> <p>01: <math>\{D_{ij}\} \leftarrow \text{compute\_density}(\bar{x}, \bar{y});</math></p> <p>02: <math>\{\widehat{D}_{ij}\} \leftarrow \text{smooth}(\{D_{ij}\});</math></p> <p>03: <math>\{\widehat{g}_{ij}\} \leftarrow \text{smooth}(\{g_{ij}\});</math></p> <p>04: <b>for</b> <math>k = 1, 2, 3, \dots, n</math></p> <p>05: <math>x_L \leftarrow x_k - w_k/2, x_R \leftarrow x_k + w_k/2;</math></p> <p>06: <math>y_B \leftarrow y_k - h_k/2, y_T \leftarrow y_k + h_k/2;</math></p> <p>07: <math>g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k \leftarrow \text{integrate}(\{\widehat{g}_{ij}\}, (x_R, y_B), (x_R, y_T))</math>  <math>\quad - \text{integrate}(\{\widehat{g}_{ij}\}, (x_L, y_B), (x_L, y_T));</math></p> <p>08: <math>g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k \leftarrow \text{integrate}(\{\widehat{g}_{ij}\}, (x_L, y_T), (x_R, y_T))</math>  <math>\quad - \text{integrate}(\{\widehat{g}_{ij}\}, (x_L, y_B), (x_R, y_B));</math></p> <p>09: <b>end for</b></p>
<p><b>Function</b> <math>\text{integrate}(\{g_{ij}\}, (x_1, y_1), (x_2, y_2))</math></p> <p>01: Select the number <math>M</math> of intervals  dividing the segment <math>(x_1, y_1) - (x_2, y_2);</math></p> <p>02: <math>\Delta x \leftarrow (x_2 - x_1)/M, \Delta y \leftarrow (y_2 - y_1)/M;</math></p> <p>03: <math>I \leftarrow \text{interpolate}(\{g_{ij}\}, (x_1, y_1)) / 2</math>  <math>\quad + \text{interpolate}(\{g_{ij}\}, (x_2, y_2)) / 2;</math></p> <p>04: <b>for</b> <math>k = 1, 2, 3, \dots, M - 1</math></p> <p>05: <math>x \leftarrow x_1 + k\Delta x, y \leftarrow y_1 + k\Delta y;</math></p> <p>06: <math>I \leftarrow I + \text{interpolate}(\{g_{ij}\}, (x, y));</math></p> <p>07: <b>end for</b></p> <p>08: <math>I \leftarrow I \cdot \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} / M;</math></p> <p>09: <b>return</b> <math>I.</math></p>
<p><b>Function</b> <math>\text{interpolate}(\{g_{ij}\}, (x, y))</math></p> <p>assume <math>(a_i, b_j) - (a_{i+1}, b_{j+1})</math> is the bin containing <math>(x, y);</math></p> <p><b>return</b> <math>g_{i,j}(a_{i+1} - x)(b_{j+1} - y) / (w_{\text{bin}} h_{\text{bin}})</math>  <math>\quad + g_{i,j+1}(a_{i+1} - x)(y - b_j) / (w_{\text{bin}} h_{\text{bin}})</math>  <math>\quad + g_{i+1,j}(x - a_i)(b_{j+1} - y) / (w_{\text{bin}} h_{\text{bin}})</math>  <math>\quad + g_{i+1,j+1}(x - a_i)(y - b_j) / (w_{\text{bin}} h_{\text{bin}}).</math></p>

Algorithm 4. Efficient gradient computation for the density penalty function

During implementation,  $g(x, y)$  is represented as a matrix instead of a continuous function, so the function value at  $(x, y)$  is computed by bilinear interpolation from the neighboring points stored in the matrix.

**Theorem 5.2.** The computational complexity of Algorithm 4 is  $T(|V|)$ , no greater than the complexity of the smoothing operation.

**Proof.** Lines 01 to 03 consume  $T(|V|)$  time for smoothing. Integrals are computed numerically in each loop between line 04 and line 09. Because the lower limit and upper limit of these integrals depend on the module size, which is a constant in the problem, they only require  $O(1)$  time to compute each integral, and  $O(|V|)$  time in total. Therefore  $T(|V|) + O(|V|) = T(|V|)$  is the time complexity for Algorithm 4. ■

The advantage of Algorithm 4 is that  $\{\widehat{g}_{ij}\}$  can be reused in every loop, so that we only need to smooth  $\{g_{ij}\}$  once. This algorithm reduces the run time by a factor of  $|V|$  in terms of computational complexity compared to the naïve method.

## 5.5 Practical Implementation in a Global Placer

Integrating the gradient computation method in a global placer involves the practical implementation of the quadratic penalty method or the augmented Lagrangian method. The following subsections will give detailed considerations for implementing Algorithm 1 and Algorithm 2.

### 5.5.1 Density Grids and Smoothed Density

The placement region is scaled into a unit square, and the density function is implemented with discrete bin grids. The grids are constructed in such a way that the grid size is equal to the average area of “most” nodes, which consist of 90% of the movable nodes and excludes the largest 5% and the smallest 5%.

The smoothing operator is selected to be the square root of the Helmholtz smoothing operator. According to Section 5.4.2, a twice-smoothed density has to be computed, thus the Helmholtz smoothing is performed on the original density. The parameter  $\varepsilon$  referred to in Section 5.2.1 is set to 100.

### 5.5.2 Step Size Selection

In each iteration, the solver searches for a minimizer of the nonlinear function  $F(\bar{x}, \bar{y}) = \text{WL}_{\text{LSE}}(\bar{x}, \bar{y}) + P(\bar{x}, \bar{y})$  for both methods, where the penalizing term  $P(\bar{x}, \bar{y})$  represents  $P_Q(\bar{x}, \bar{y}; \mu)$  in the quadratic penalty method and represents  $P_A(\bar{x}, \bar{y}, \lambda; \mu)$  in the augmented Lagrangian method. Given an initial solution  $(\bar{x}^{(k)}, \bar{y}^{(k)})$  from the last iteration, the gradient descent method can be used to minimize  $F(\bar{x}, \bar{y})$  by iteratively updating

$$(\bar{x}_{\text{new}}, \bar{y}_{\text{new}}) \leftarrow (\bar{x}_{\text{old}}, \bar{y}_{\text{old}}) - \alpha \nabla F(\bar{x}_{\text{old}}, \bar{y}_{\text{old}}) \quad (56)$$

until it converges to a stationary point  $(\bar{x}^{(k+1)}, \bar{y}^{(k+1)})$ , which is a local minimizer of the function  $F(\bar{x}, \bar{y})$ .

In order to reduce the runtime, a constant step size is used instead of line search. If the gradient descent method cannot converge with step size  $\alpha$  for an initial solution  $(\bar{x}^{(k)}, \bar{y}^{(k)})$ , a smaller step size (e.g.,  $0.6\alpha$ ) will be tried.

In general, the constant step size  $\alpha$  at the  $k$ -th iteration is

$$\alpha = k\eta^m\alpha_0 \quad (57)$$

where  $\alpha_0$  is the initial step size,  $m$  is the number of disconvergent trials from the first iteration to the  $k$ -th iteration, and  $\eta < 1$  is the factor to reduce the step size. The current iteration number  $k$  is multiplied as a heuristic to provide an opportunity to increase the step size again. During the implementation,  $\alpha_0 = 1$  and  $\eta = 0.6$  are used.

### 5.5.3 Penalty Factor Update Scheme

This update scheme of the penalty factor  $\mu$  is applied to both the quadratic penalty method and the augmented Lagrangian method. The basic idea is to increase  $\mu$  when the overlap removal becomes slow.

$$\mu^{(k+1)} \leftarrow \begin{cases} \gamma\mu^{(k)} & \text{(overlap removal is slow)} \\ \mu^{(k)} & \text{(otherwise)} \end{cases} \quad (58)$$

where  $\gamma > 0$  and  $\gamma = 1.2$  are used during implementation.

To facilitate the decision concerning whether overlap removal is slow, two concepts are used: the percentage of overlapped area OVL is defined as

$$\text{OVL} = \frac{\int_0^1 \int_0^1 \max(0, D(u, v) - 1) dudv}{\int_0^1 \int_0^1 D(u, v) dudv} \quad (59)$$

and the overlap reduction rate  $r$  is defined as

$$r = (\text{OVL}_{k-1} - \text{OVL}_k) / \text{OVL}_{k-1} \quad (60)$$

We consider that the overlap removal is slow, when ( $OVL > 25\%$  and  $r < 5\%$ ) or ( $OVL \leq 25\%$  and  $r < 0.5\%$ ).

#### 5.5.4 Lagrangian Multiplier Update Scheme

Different from the scheme in Algorithm 2, the Lagrangian multiplier is not updated every iteration. Instead, it updates only when the penalty factor  $\mu$  is not increasing, which means that the Lagrangian multiplier is updated only when the overlap removal is fast enough.

To make the iterative method more stable, a damping factor  $\beta$  is introduced in the update:

$$\lambda^{(k+1)} \leftarrow \lambda^{(k)} + \beta \mu^{(k)} (\widehat{D}(\bar{x}^{(k+1)}, \bar{y}^{(k+1)}) - 1) \quad (61)$$

During implementation  $\beta = \gamma - 1$  is used where  $\gamma$  is defined in Section 5.5.3.

#### 5.5.5 Stopping Criterion

The percentage of cell overlap OVL is used as a stopping criterion. For most circuits,  $OVL \leq 10\%$  is small enough to be removed by the detailed placer. For the circuits with large module size variation (e.g., IBM-HB+ [68]), the stopping criterion can be set to  $OVL \leq 3\%$  to remove more overlaps in the global placement phase.

### 5.6 Experimental Results

To solve problem (28) defined in Section 5.1, we implemented the gradient computation with both the quadratic penalty method and augmented Lagrangian method, in a multilevel framework known as mPL6 [16].

The benchmarks used in experiments include IBM-HB+ [68] and the modified ISPD'05 and ISPD'06 placement contest benchmarks [66][64], which are summarized in Table 6.. The number of movable nodes, the number of nets, and the percentage of whitespace are listed in the table.

Table 6. Benchmark statistics

<b>IBM-HB+</b>				<b>modified ISPD'05 &amp; ISPD'06</b>			
circuit	#cell	#net	WS%	circuit	#cell	#net	WS%
ibm01	911	5829	20.1	adaptec2	254616	266009	21.4
ibm02	1471	8508	20.0	adaptec4	496141	515951	37.3
ibm03	1289	10279	21.2	bigblue1	277636	284479	45.8
ibm04	1584	12456	20.0	bigblue2	557962	577235	38.1
ibm06	749	9963	20.0	bigblue3	1096908	1123170	14.3
ibm07	1120	15047	20.0	bigblue4	2177449	2229886	34.7
ibm08	1269	16075	20.3	adaptec5	843224	867798	21.3
ibm09	1113	18913	20.1	newblue1	330137	338901	29.3
ibm10	1595	27508	16.7	newblue2	441586	465219	13.8
ibm11	1497	27477	20.0	newblue3	494123	552199	15.3
ibm12	1233	26320	20.6	newblue4	646219	637051	34.2
ibm13	954	27011	20.0	newblue5	1233154	1284251	25.4
ibm14	1635	43062	20.0	newblue6	1255135	1288443	40.7
ibm15	1412	52779	20.1				
ibm16	1091	47821	20.0				
ibm17	1442	56517	20.0				
ibm18	943	42200	20.0				

The first experiment was performed on the IBM-HB+ benchmark, which consists of hard instances with large node size variation. The placement results of mPL6, SCAMPI [68], the quadratic penalty method and the augmented Lagrangian method are shown in Table 7 and Table 8. The data of SCAMPI is obtained from [68], but the other global placements are fed into the detailed placer of NTUplace-DP [20] for the final placement. The overlap-free condition was verified for the final placement. The column “ovl%” indicates the percentage of overlap after global placement. The columns “WL” and “RT” are respectively the total wirelength and the total runtime for the whole placement. The

column “diff%” is the relative difference compared to mPL6. The results show that the methods integrated with the exact gradient computation shorten runtime by almost 70%, and improve wirelength by about 15%. These results reveal that for the circuits with large module size variation, the exact gradient computation leads to faster convergence and better placement quality.

Table 7. Experimental results on IBM-HB+ benchmark (mPL6 & SCAMPI)

IBM-HB+	mPL6 + NTUplace-DP			SCAMPI		
	ovl%	WL (10 <sup>7</sup> )	RT (min)	WL (10 <sup>7</sup> )	diff%	RT (min)
ibm01	1.42%	0.33	0.79	0.34	3.0%	1.03
ibm02	1.35%	0.74	4.29	0.8	8.1%	2.33
ibm03	1.12%	0.92	2.13	0.95	3.3%	1.74
ibm04	1.65%	1.12	4.60	1.23	9.8%	2.40
ibm06	0.89%	0.92	3.87	1.1	19.6%	2.83
ibm07	1.61%	1.66	3.30	1.57	-5.4%	1.65
ibm08	1.80%	2.06	7.43	2.05	-0.5%	3.14
ibm09	1.54%	2.11	6.96	2.22	5.2%	3.03
ibm10	8.60%	7.06	11.08	5.52	-21.8%	5.33
ibm11	1.97%	3.00	9.26	2.78	-7.3%	2.41
ibm12	1.47%	5.64	7.42	6.76	19.9%	6.77
ibm13	1.96%	3.91	9.70	4.22	7.9%	3.49
ibm14	1.83%	7.52	9.88	6.64	-11.7%	4.47
ibm15	1.62%	10.35	21.23	8.82	-14.8%	6.27
ibm16	1.98%	10.23	10.76	10.62	3.8%	5.11
ibm17	2.00%	16.32	17.40	15.27	-6.4%	6.43
ibm18	1.98%	9.10	15.68	7.78	-14.5%	3.21
average	2.05%				-0.1%	

Table 8. Experimental results on IBM-HB+ benchmarks (our methods)

IBMHB+	Quadratic Penalty					Augmented Lagrangian				
	ovl%	WL(10 <sup>7</sup> )	diff%	RT(min)	diff%	ovl%	WL(10 <sup>7</sup> )	diff%	RT(min)	diff%
ibm01	1.34%	0.30	-9.09%	0.49	-38.0%	3.86%	0.28	-15.15%	0.55	-30.4%
ibm02	1.08%	0.61	-17.57%	1.48	-65.5%	1.58%	0.58	-21.62%	1.25	-70.9%
ibm03	1.10%	0.86	-6.52%	1.16	-45.5%	1.67%	0.82	-10.87%	1.12	-47.4%
ibm04	1.28%	1.06	-5.36%	1.54	-66.5%	1.87%	1.01	-9.82%	1.64	-64.3%
ibm06	0.87%	0.87	-5.43%	1.17	-69.8%	1.27%	0.80	-13.04%	0.92	-76.2%
ibm07	1.62%	1.58	-4.82%	2.24	-32.1%	2.41%	1.56	-6.02%	1.98	-40.0%
ibm08	1.02%	1.85	-10.19%	2.11	-71.6%	1.55%	1.71	-16.99%	1.73	-76.7%
ibm09	1.54%	1.61	-23.70%	1.83	-73.7%	2.27%	1.61	-23.70%	1.64	-76.4%
ibm10	1.57%	4.38	-37.96%	3.72	-66.4%	2.38%	3.96	-43.91%	2.96	-73.3%
ibm11	1.64%	2.75	-8.33%	2.68	-71.1%	2.40%	2.58	-14.00%	2.05	-77.9%
ibm12	1.47%	4.86	-13.83%	3.14	-57.7%	2.18%	4.85	-14.01%	2.22	-70.1%
ibm13	1.60%	3.56	-8.95%	2.29	-76.4%	2.42%	3.47	-11.25%	2.05	-78.9%

ibm14	1.41%	6.70	-10.90%	3.90	-60.5%	2.72%	6.50	-13.56%	3.59	-63.7%
ibm15	1.05%	8.12	-21.55%	5.17	-75.6%	1.66%	8.01	-22.61%	4.86	-77.1%
ibm16	1.99%	9.55	-6.65%	4.45	-58.6%	3.00%	9.37	-8.41%	4.46	-58.6%
ibm17	1.76%	14.98	-8.21%	6.19	-64.4%	2.97%	14.86	-8.95%	5.24	-69.9%
ibm18	1.99%	8.35	-8.24%	4.17	-73.4%	2.98%	7.78	-14.51%	4.06	-74.1%
average	1.4%		-12.19%		-62.8%	2.3%		-15.79%		-66.2%

Moreover, this experiment also shows the augmented Lagrangian method performs better than the quadratic penalty method, both in terms of wirelength quality and runtime. Although the average overflow after global placement of the augmented Lagrangian method is slightly higher than the quadratic penalty method, the result shows that within such a small overflow, the global placement of the augmented Lagrangian method is indeed better because it leads to better final placement.

The second experiment was performed on the modified ISPD'05 and ISPD'06 benchmarks. The original benchmarks mostly consist of movable standard cells and fixed macros. We modified the benchmarks and made every object inside the placement region movable. In addition, the experiment is run for wirelength minimization only instead of density-aware wirelength minimization, as in the placement contests. The results of the modified benchmarks are showed in Table 9. Compared to the mPL6 results, the augmented Lagrangian method achieves an average of 3% shorter wirelength, and it is as much as 9% shorter. The runtime advantage of our method compared to mPL6 is not as significant as on IBM-HB+ benchmarks, because the macro size variations in the contest benchmarks are not as large. However, we still achieve a comparable or slightly better runtime.

Table 9. Experimental results on the modified ISPD'05 and ISPD'06 benchmarks

<b>modified ISPD'05</b>	<b>mPL6 + NTUplace-DP</b>	<b>Augmented Lagrangian + NTUplace-DP</b>
-------------------------	---------------------------	---



ISPD'06	WL (x 10 <sup>8</sup> )		RT (hour)		WL (x 10 <sup>8</sup> )			RT (hour)		
	global	final	global	total	global	final	diff%	global	total	diff%
<b>adaptec2</b>	0.84	0.86	0.72	0.77	0.82	0.81	-5.40%	0.79	0.83	6.80%
<b>adaptec4</b>	1.72	1.70	3.67	3.79	1.60	1.56	-8.40%	2.34	2.48	-34.50%
<b>bigblue1</b>	1.02	0.99	0.69	0.73	0.97	0.94	-5.60%	0.92	0.95	30.20%
<b>bigblue2</b>	1.19	1.14	4.16	5.02	1.10	1.07	-6.30%	2.89	3.55	-29.20%
<b>bigblue3</b>	3.36	3.24	5.73	6.18	3.48	3.36	3.60%	5.60	6.18	0.00%
<b>bigblue4</b>	7.94	7.73	7.12	7.84	7.99	7.62	-1.30%	8.48	9.15	16.70%
<b>adaptec5</b>	3.14	3.03	4.13	4.31	3.35	3.20	5.70%	3.89	4.08	-5.50%
<b>newblue1</b>	0.65	0.63	0.99	1.07	0.63	0.61	-2.50%	1.13	1.21	12.80%
<b>newblue2</b>	1.86	1.80	3.27	3.42	1.93	1.82	1.20%	2.89	3.07	-10.10%
<b>newblue3</b>	2.46	2.53	1.71	2.81	2.41	2.40	-5.00%	3.39	4.21	49.90%
<b>newblue4</b>	2.30	2.23	3.40	3.58	2.10	2.03	-9.00%	1.97	2.11	-40.90%
<b>newblue5</b>	4.08	3.92	5.95	6.36	3.98	3.80	-3.20%	5.46	5.83	-8.40%
<b>newblue6</b>	4.46	4.24	5.67	6.22	4.33	4.12	-2.80%	5.26	5.71	-8.20%
<b>average</b>							-3.00%			-1.60%

## 5.7 Conclusions

In this chapter we introduced a general class of density-smoothing operations, and developed the theory and efficient algorithms for computing the gradient of density penalty functions in the nonlinear programming framework. This is the first time that the density-constrained placement problem with a global smoothing technique has been solved exactly in the general nonlinear programming framework. We showed that such an approach supersedes the existing force-directed placement methods. The experiments on the IBM-HB+ benchmarks and the modified ISPD'05, ISPD'06 benchmarks with movable macros demonstrate the effectiveness of our technique.

In the next two chapters, our analytical placement approach with efficient computation of the density gradient will show promise in its application to the following problems: (i) 3D placement with non-overlapping constraints, and (ii) thermal-aware 3D placement with TSV distribution constraints.

## Chapter 6

### Multilevel Analytical Placement for 3D ICs

As mentioned in Chapter 2, all the existing 3D placement approaches are able to explore the trade-offs among the wirelength, the number of TSVs and the peak temperature. However, if the solution quality of those techniques is suboptimal, the conclusions drawn from their results are questionable. Our goal of this chapter is to develop a high-quality solver for a basic 3D placement problem that targets the wirelength and the number of TSVs, so that this solver can be used to include other constraints and objectives (e.g., maximum temperature) in 3D physical design with robust results.

In particular, we develop a 3D placement approach using a nonlinear optimization method to obtain a global placement. The main idea is to perform overlap removal and die assignment simultaneously with an area density penalty function for both node and TSV density constraints. The minimizer of this density penalty function is an overlap-free solution in the  $(x, y)$  directions, as well as a legal die assignment in the  $z$  direction. The objective is a weighted sum of the wirelength and the number of TSVs.

Existing 3D placement techniques are mainly used for standard-cell circuits, while mixed-size placement is needed to support placing with high-level functional modules and IP blocks. In this chapter we also present an analytical 3D placement method that is capable of handling mixed-size circuits. A multiple-stepscheme for the analytical solver is developed to handle standard cells and macros differently for stability and

efficiency. To relieve the difficulty of legalization, 3D floorplan-based initial solutions are used to guide the analytical solver. As far as we know, this is the first work that reports 3D placement results for mixed-size circuits.

## 6.1 Continuous Relaxation for Die Assignment

In this chapter we focus on the analytical algorithms to solve the following 3D placement problem:

$$\begin{aligned}
& \text{minimize} && \sum_{e \in E} (\text{WL}(e) + \alpha_{\text{TSV}} \cdot \text{TSV}(e)) \\
& \text{subject to} && \sum_{v_i \in V} \text{Overlap}(\text{bin}_{m,n,k}, v_i) \leq w_{\text{bin}} \cdot h_{\text{bin}} \quad \text{for } 1 \leq n \leq N \\
& && 1 \leq m \leq M \\
& && 1 \leq k \leq K
\end{aligned} \tag{62}$$

The feasible solution  $(\bar{x}, \bar{y}, \bar{z})$  of this problem must satisfy that  $\bar{x} \in [0, W_{\text{die}}]^{|V|}$ ,  $\bar{y} \in [0, H_{\text{die}}]^{|V|}$  and  $\bar{z} \in \{1, \dots, K\}^{|V|}$ . In 2D placement, although the non-overlapping constraints create some nonconvexity in the feasible solution  $(\bar{x}, \bar{y})$ , the density-based analytical placement methods are able to handle these constraints and achieve high-quality results. However in 3D placement, even ignoring the non-overlapping constraints, the variables  $\bar{z} \in \{1, \dots, K\}^{|V|}$  are non-continuous and require special consideration.

To make use of the analytical placement techniques, these non-continuous variables are relaxed and mapped to a continuous space. A virtual 3D placement region  $[0, W_{\text{die}}] \times [0, H_{\text{die}}] \times [1, K]$  becomes the feasible region for the 3D global placement. This definition is compliant to the discrete case, when  $z_i = k$  indicates the node  $v_i$  is placed on die  $k$ .

## 6.2 3D Global Placement with Bell-Shaped Area Projection

The 3D placement problem is handled by solving a sequence of unconstrained problems during the global placement phase. The unconstrained problem to be minimized is a penalized objective function, written as

$$\text{OBJ}^+(\bar{x}, \bar{y}, \bar{z}; \mu) = \text{OBJ}(\bar{x}, \bar{y}, \bar{z}) + \mu \cdot \text{Penalty}(\bar{x}, \bar{y}, \bar{z}) \quad (63)$$

where the wirelength objective function  $\text{OBJ}(\bar{x}, \bar{y}, \bar{z})$  is defined as in Section 2.4.1 with  $\gamma_e = 0$ , and the penalty factor  $\mu$  is to be increased heuristically to reduce the area density violations. The global placement flow is shown in Figure 15.

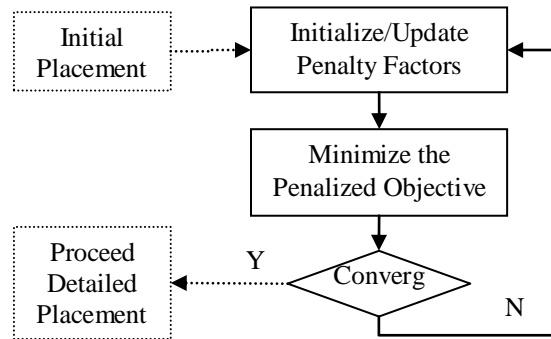


Figure 15. 3D analytical placement flow

As mentioned in Section 6.1, the placement variables of node  $v_i$  are represented by  $(x_i, y_i, z_i)$ , where  $z_i$  is initially a discrete variable. This  $z_i$  is relaxed from a discrete variable in  $\{1, 2, \dots, K\}$  to a continuous variable in  $[1, K]$ . After relaxation, a nonlinear optimization is applicable in the global placement phase. The relaxed variables are mapped back to the discrete values in the detailed placement phase.

The density penalty function  $\text{Penalty}(\bar{x}, \bar{y}, \bar{z})$  is for overlap removal in both the  $(x, y)$  direction and the  $z$  direction. The minimization of the density penalty function leads to a placement solution satisfying the non-overlapping constraints.

Assume that every node  $v_i$  has a legal die assignment (i.e.,  $z_i \in \{1, 2, \dots, K\}$ ), we can define  $K$  area density functions for this  $K$  dies. The area density function  $D_k(u, v)$  indicates the number of nodes that cover the point  $(u, v)$  on the  $k$ -th die. It is defined as:

$$D_k(u, v) = \sum_{v_i \in V} \delta(z_i, k) d_i(u, v) \quad (64)$$

where  $\delta(z_i, k)$  is an indicator function for the nodes on the  $k$ -th die, as defined in equation (9) of Chapter 2; the function  $d_i(u, v)$  is the density contribution of node  $v_i$  on the  $k$ -th die, which is defined in equation (27) of Chapter 5. The density contribution  $d_i(u, v)$  is one inside the area occupied by  $v_i$ , and is zero outside this area. An example of  $D_k(u, v)$  is given in Figure 16 showing the density function with two overlapping nodes.

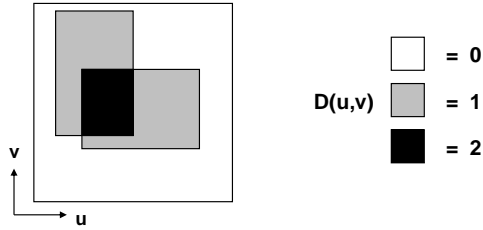


Figure 16. An example of the density function

During global placement, it is possible that node  $v_i$  is moving between two dies temporally, and the variable  $z_i \in [1, K]$  is not aligned to any of them. We borrow the idea from the bell-shaped function [67] to define the density function for this case:

$$D_k(u, v) = \sum_{v_i \in V} \eta_k(z_i) d_i(u, v) \quad (65)$$

where

$$\eta_k(z) = \begin{cases} 1 - 2(z - k)^2 & (|z - k| \leq 1/2) \\ 2(|z - k| - 1)^2 & (1/2 < |z - k| \leq 1) \\ 0 & (\text{otherwise}) \end{cases} \quad (66)$$

This is an extension of the density function, which we call it *bell-shaped density projection function*, and it is consistent with the previous definition in equation (64) when  $z = k$ .

An example of how this function works for a four-die 3D placement is given in Figure 17. The x-axis is the placement of a node in  $z$  direction, while the y-axis indicates the amount of area to be projected to the corresponding dies. The four dotted segments represent the four dies with colors red, green, blue and pink for the 1<sup>st</sup> die, the 2<sup>nd</sup> die, the 3<sup>rd</sup> die, and the 4<sup>th</sup> die, respectively. And the four colored curves are the four bell-shaped density projection functions. A node placed approximately at “die 2.3,” which means it is placed between the 2<sup>nd</sup> die and the 3<sup>rd</sup> die, is shown in the figure. By looking up the bell-shaped density projection function, we find that it should project 80% of its area to the 2<sup>nd</sup> die (solid curve), and project 20% of its area to the 3<sup>rd</sup> die (dash-dot curve). In this way, we establish a mapping between the relaxed 3D placement and a legalized 3D placement.

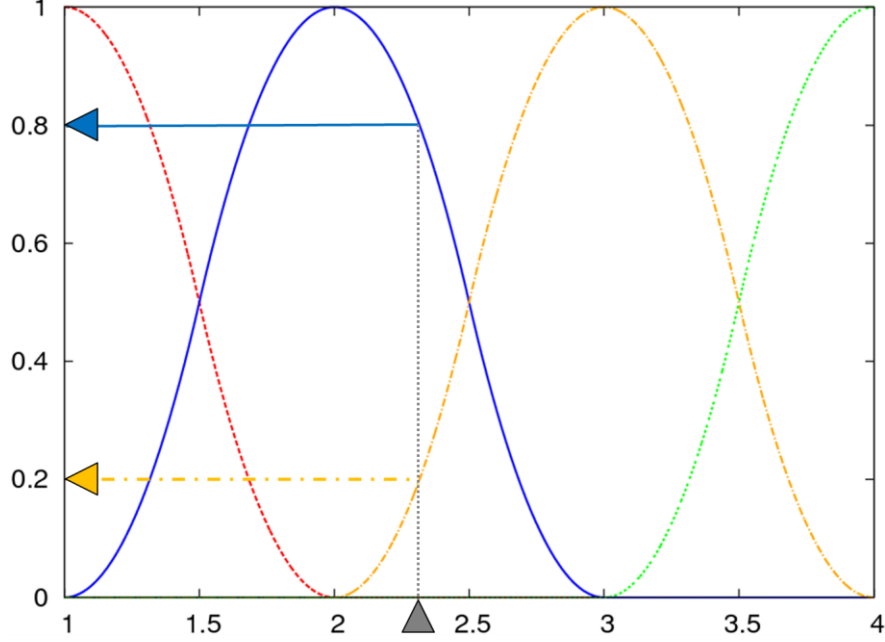


Figure 17. Bell-shaped density projection function

Inspired by the quadratic penalty terms in 2D placement methods, as discussed in Section 5.3.1, we define an area density penalty function to measure the amount of overlaps as the following:

$$P(\bar{x}, \bar{y}, \bar{z}) = \sum_{k=1}^K \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (D_k(u, v) - 1)^2 dudv \quad (67)$$

**Lemma 6.1.** Assume the total area of nodes equals the total placement area (i.e.,

$$\sum_{k=1}^K \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} D_k(u, v) dudv = KW_{\text{die}}H_{\text{die}} \text{ without empty space), every legal placement}$$

$(\bar{x}^*, \bar{y}^*, \bar{z}^*)$ , which satisfies  $D_k(u, v) = 1$  for every  $k$  and  $(u, v)$ , is a minimizer of  $P(\bar{x}, \bar{y}, \bar{z})$ .

The proof is trivial. For every legal placement the equality  $D_k(u, v) \leq 1$  holds. Since there is no empty space,  $D_k(u, v) = 1$  holds for every  $k$  and  $(u, v)$ ; otherwise there will

be overlaps. Because  $P(\bar{x}, \bar{y}, \bar{z}) \geq 0$  and  $P(\bar{x}^*, \bar{y}^*, \bar{z}^*) = 0$ , every legal placement satisfying the assumption in Lemma 6.1 is a minimizer of  $P(\bar{x}, \bar{y}, \bar{z})$ .

Therefore, minimizing  $P(\bar{x}, \bar{y}, \bar{z})$  provides a necessary condition for a legal placement. However, there exist minimizers that are not legal. These minimizers have non-integer values in the  $z$  direction. For example, if two nodes are placed in two neighboring dies, then placing both of them in the middle of these two dies will not increase the density penalty cost, and this illegal placement is still a minimizer.

To avoid reaching such minimizers, we introduce the inter-die density function:

$$E_k(u, v) = \sum_{v_i \in V} \eta_{k+0.5}(z_i) d_i(u, v) \quad \text{for } 1 \leq k \leq K-1 \quad (68)$$

and the corresponding density penalty function:

$$Q(\bar{x}, \bar{y}, \bar{z}) = \sum_{k=1}^{K-1} \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (E_k(u, v) - 1)^2 dudv \quad (69)$$

Similar to the density penalty function  $P(\bar{x}, \bar{y}, \bar{z})$ , the following lemma can be proved in the same way.

**Lemma 6.2.** Assume the total area of cells equals the placement area, every legal placement is a minimizer of  $Q(\bar{x}, \bar{y}, \bar{z})$ .

Combining the density penalty functions  $P(\bar{x}, \bar{y}, \bar{z})$  and  $Q(\bar{x}, \bar{y}, \bar{z})$ , we define the following density penalty function:

$$\text{Penalty}(\bar{x}, \bar{y}, \bar{z}) = P(\bar{x}, \bar{y}, \bar{z}) + Q(\bar{x}, \bar{y}, \bar{z}) \quad (70)$$

**Theorem 6.1.** Assume the total area of cells equals the total placement area, every legal placement  $(\bar{x}^*, \bar{y}^*, \bar{z}^*)$  is a minimizer of  $\text{Penalty}(\bar{x}, \bar{y}, \bar{z})$ , and vice versa.



**Proof.** It is obvious that every legal placement is a minimizer of  $\text{Penalty}(\bar{x}, \bar{y}, \bar{z})$  by combining *Lemma 1* and *Lemma 2*. We shall prove that every minimizer  $(\bar{x}^*, \bar{y}^*, \bar{z}^*)$  of  $\text{Penalty}(\bar{x}, \bar{y}, \bar{z})$  is a legal placement. Again, from the proof of *Lemma 1* and *Lemma 2*, the minimum value of  $\text{Penalty}(\bar{x}, \bar{y}, \bar{z})$  is achieved when  $D_k(u, v) = 1$  and  $E_k(u, v) = 1$  for every  $k$  and  $(u, v)$ . First, if all the components of  $\bar{z}^*$  are integers, it is easy to see the placement is legal, because all the nodes are assigned to a die, and at every point  $(u, v)$  on any die, there is only one node covering this point (no overlaps).

Next, we show that there does not exist a  $z_i^*$  with a non-integer value. If any  $z_i^*$  is not integer, there are  $K$  nodes that cover the point  $(x_i^*, y_i^*)$ , because  $\sum_{k=1}^K D_k(x_i^*, y_i^*) = K$ . The values in the  $z$  direction are in the range of  $[1, K]$ , thus among these  $K$  nodes there are at least two nodes with  $z$  values  $z_1$  and  $z_2$  that satisfy the distance  $|z_1 - z_2| < 1$ . Therefore, there exists a number  $k \in \{1, 2, \dots, K\}$  such that either  $z_1, z_2 \in [k, k+1)$ , or  $z_1, z_2 \in [k-0.5, k+0.5)$ . For the first case, if  $z_1, z_2 \in [k, k+1)$ , the value of the inter-die area density is  $E_k(x_i^*, y_i^*) \geq \eta_{k+0.5}(z_1) + \eta_{k+0.5}(z_2) > 1$ , which conflicts with the assumption that  $(\bar{x}^*, \bar{y}^*, \bar{z}^*)$  is a minimizer. For the second case, if  $z_1, z_2 \in [k-0.5, k+0.5)$ , the value of the die area density is  $D_k(x_i^*, y_i^*) = \eta_k(z_1) + \eta_k(z_2) > 1$ , which also results in conflict.

Therefore, there does not exist a  $z_i^*$  with a non-integer value, and every minimizer of  $\text{Penalty}(\bar{x}, \bar{y}, \bar{z})$  is a legal placement.



During implementation, the densities  $D_k(u, v)$  and  $E_k(u, v)$  are replaced by smoothed densities  $\widehat{D}_k(u, v)$  and  $\widehat{E}_k(u, v)$  for differentiability. These densities are smoothed by solving the Helmholtz equation (Section 5.2.1):

$$\begin{aligned}\widehat{D}_k(u, v) &= -\left(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2} - \varepsilon\right)^{-1} D_k(u, v) \\ \widehat{E}_k(u, v) &= -\left(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2} - \varepsilon\right)^{-1} E_k(u, v)\end{aligned}\tag{71}$$

And the corresponding smoothed density penalty function

$$\begin{aligned}\text{Penalty}(\bar{x}, \bar{y}, \bar{z}) &= \sum_{k=1}^K \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (\widehat{D}_k(u, v) - 1)^2 dudv \\ &\quad + \sum_{k=1}^{K-1} \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (\widehat{E}_k(u, v) - 1)^2 dudv\end{aligned}\tag{72}$$

is used in our implementation; its gradient is computed with the method presented in Chapter 5.

### 6.3 3D Global Placement with Huber-Based Smoothing

In the last section we presented a 3D global placement approach using a bell-shaped area projection, with 2D Helmholtz smoothing for the on-die area density and the inter-die area density. In this section we present a method that enables a 3D Helmholtz smoothing of the 3D area density. The results will show that the placement quality with 3D Helmholtz smoothing is as good as the multilayer 2D Helmholtz smoothing, and the method discussed in this section provides an alternative to the bell-shaped area projection.

The gradient computation method, which is discussed in Chapter 5 and later applied to the 3D placement approach in the previous section (Section 6.2), considers the area

density function continuous by assuming the resolution  $M \times N$  to be infinity. However, the area density map on the  $k$ -th die is implemented as a two-dimensional array  $\{D_{m,n,k}\}$  with  $1 \leq m \leq M$  and  $1 \leq n \leq N$ , instead of a continuous function  $D_k(u, v)$  with  $u \in [0, W_{\text{die}}]$  and  $v \in [0, H_{\text{die}}]$ . There is a gap between the formulation and the implementation. In this section, we are bridging this gap by introducing a local smoothing based on the Huber function, which provides an alternative to the bell-shaped local smoothing, and also gives another angle to understand the gradient computation method in Chapter 5 in a discrete formulation.

In this section, we will first present the formulation of the 3D area density constraints. The overlapping function can be expressed by a sum of absolute values, which are then approximated by the Huber function. Finally, the joint local smoothing and global smoothing formulations of the 3D area density constraints and the related penalty functions are presented.

### 6.3.1 3D Area Density Constraints

In order to measure the 3D area density, we define a virtual 3D placement region as  $[0, W_{\text{die}}] \times [0, H_{\text{die}}] \times [0, K]$ , which is divided into  $M \times N \times L$  bins. Each bin has a width of  $w_{\text{bin}} = W_{\text{die}}/M$ , a height of  $h_{\text{bin}} = H_{\text{die}}/N$ , and a depth of  $d_{\text{bin}} = K/L$ . Accordingly, node  $v_i$  consumes an area of  $[x_i - w_i/2, x_i + w_i/2] \times [y_i - h_i/2, y_i + h_i/2] \times [z_i - 1, z_i]$  in the virtual placement region, where  $x_i \in [0, W_{\text{die}}]$ ,  $y_i \in [0, H_{\text{die}}]$  and  $z_i \in [1, K]$ .

After inserting filler nodes (Section 5.1), the 3D placement problem with 3D area density constraints are formulated as:

$$\begin{aligned}
& \text{minimize} && \sum_{e \in E} (\text{WL}(e) + \alpha_{\text{TSV}} \cdot \text{TSV}(e)) \\
& \text{subject to} && \sum_{v_i \in V} \text{Overlap}(\text{bin}_{m,n,l}, v_i) = w_{\text{bin}} h_{\text{bin}} d_{\text{bin}} \quad \text{for } 1 \leq n \leq N \\
& && 1 \leq m \leq M \\
& && 1 \leq l \leq L
\end{aligned} \tag{73}$$

The number of bins  $L$  in the  $z$  direction is not necessarily the same as the number of dies  $K$ . As discussed in Section 6.2,  $L=K$  is not enough to capture the non-overlapping constraints, and  $L=2K$  is sufficient to reflect the non-overlapping constraints by the 3D area density constraints. Thus, we assume  $L=2K$  in the remainder of this section.

### 6.3.2 Local Smoothing of the Overlapping Function

The overlapping function between a node  $v_i$  and  $\text{bin}_{m,n,l}$  is defined as

$$\begin{aligned}
\text{Overlap}(\text{bin}_{m,n,l}, v_i) &= \text{Overlap}_x(\text{bin}_{m,n,l}, v_i) \\
&\quad \times \text{Overlap}_y(\text{bin}_{m,n,l}, v_i) \\
&\quad \times \text{Overlap}_z(\text{bin}_{m,n,l}, v_i)
\end{aligned} \tag{74}$$

The overlapping functions  $\text{Overlap}_x(\text{bin}_{m,n,l}, v_i)$  and  $\text{Overlap}_y(\text{bin}_{m,n,l}, v_i)$  are defined in equations (10) and (11) in Section 2.4.2, and the overlapping function in the  $z$  direction is defined as

$$\begin{aligned}
& \text{Overlap}_z(\text{bin}_{m,n,l}, v_i) \\
&= \text{common\_length}([(l-1) \cdot d_{\text{bin}}, l \cdot d_{\text{bin}}], [z_i - 1, z_i])
\end{aligned} \tag{75}$$

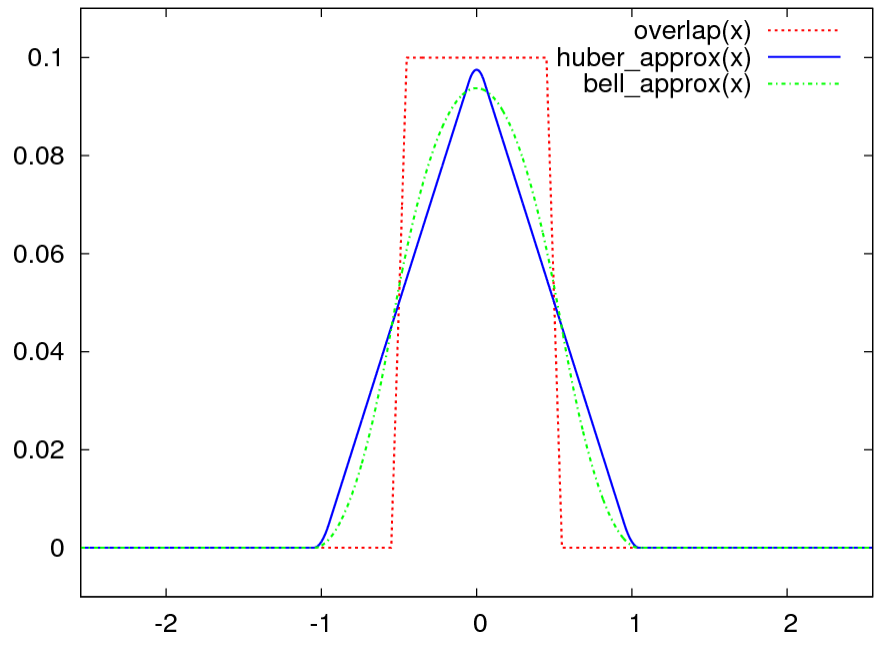
We observe that the common length function can be explicitly expressed as

$$\text{common\_length}([a, b], [c, d]) = \frac{1}{2} \times (|b-c| + |a-d| - |b-d| - |a-c|) \tag{76}$$

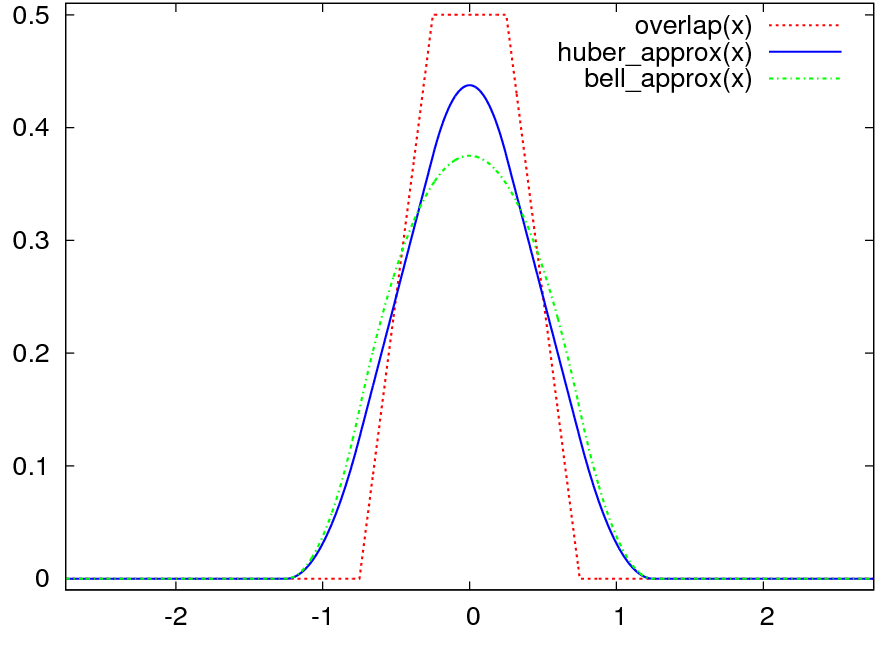
These overlapping functions are non-differentiable because of the absolute values. Similar to the max function (Section 5.1), there are multiple ways to approximate the absolute value function by a differentiable function. We use the Huber function [13] as an approximation, which is defined as

$$|x| \approx h_\mu(x) = \begin{cases} x^2/(2\mu) & (|x| \leq \mu) \\ |x| - \mu/2 & (|x| \geq \mu) \end{cases} \quad (77)$$

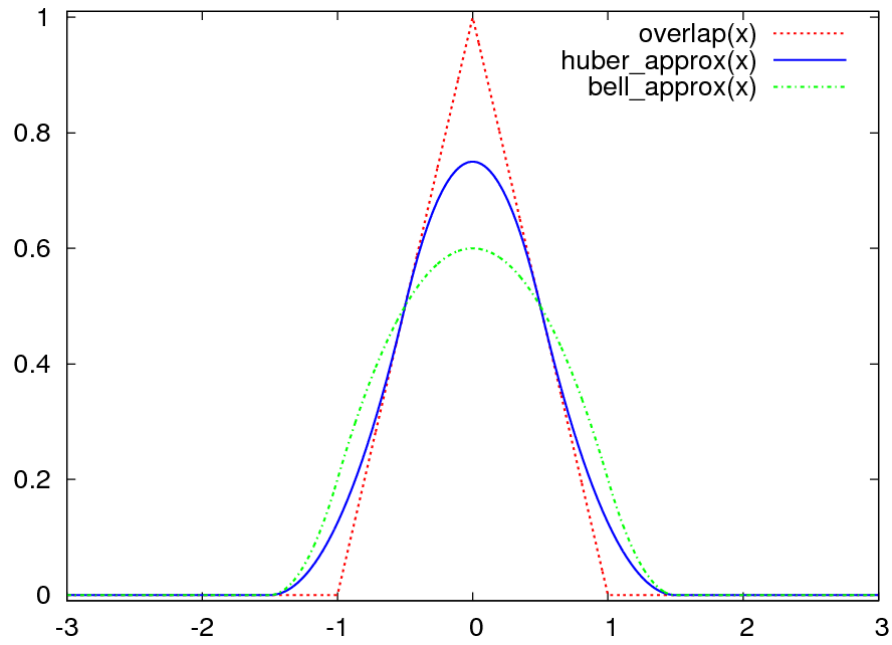
The overlapping function can also be approximated by a bell-shaped function [67]. To compare the overlapping function, the bell-shaped approximation, and the Huber-based approximation, we visualize these functions in Figure 18 as the overlaps between a node and a bin in one dimension. The bin is located at the origin point with bin width 1, and the nodes are with different widths from  $0.1 \times$  bin width to  $10 \times$  bin width, illustrated from Figure 18(a) to Figure 18(e). The bell-shaped approximation is scaled so that the area covered by the curve is equal to the node area. The Huber-based approximation is generated by setting the parameter  $\mu$  to be a half of the bin width. The x-axis is the placement of the node, and the y-axis presents the overlapping length. These figures show that the Huber-based approximation is more accurate than the bell-shaped approximation, especially when the node width is much greater (e.g.,  $10 \times$  greater) than the bin width.



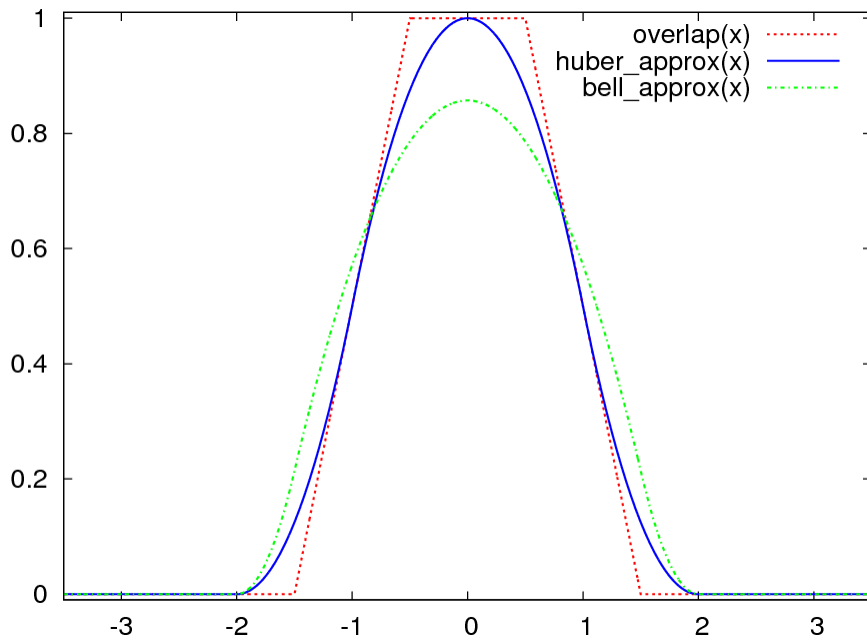
(a)  $node\_width = 0.1 \times bin\_width$



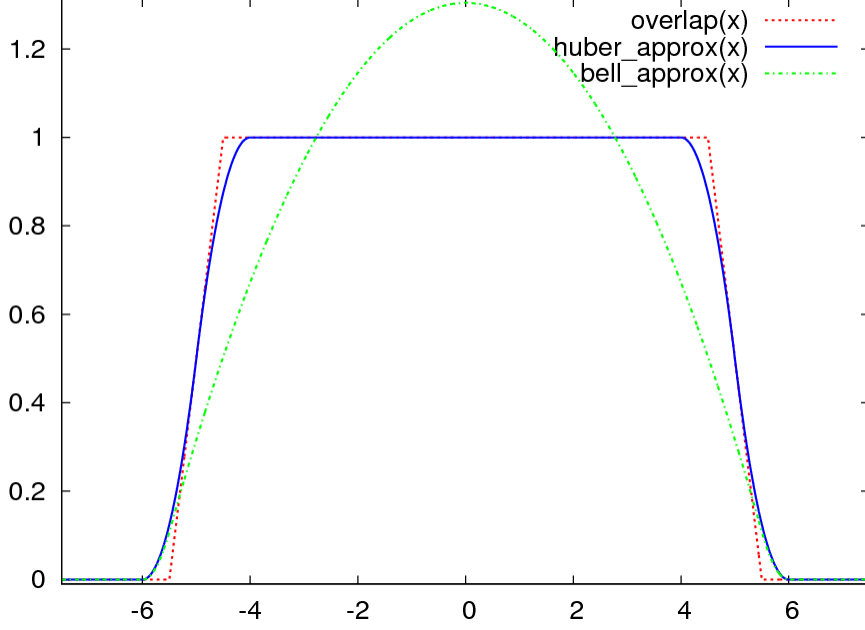
(b)  $node\_width = 0.5 \times bin\_width$



(c)  $node\_width = 1.0 \times bin\_width$



(d)  $node\_width = 2.0 \times bin\_width$



(e) node\_width = 10 × bin\_width

Figure 18. The overlapping function, the bell-shaped approximation, and the Huber-based approximation

### 6.3.3 Density Penalty Function and Global Smoothing

We define  $D(\bar{x}, \bar{y}, \bar{z})$  as a vectorized version of the 3D area density array  $\{\sum_{v_i \in V} \text{Overlap}(\text{bin}_{m,n,l}, v_i)\}$  with  $M \times N \times L$  elements, and  $C$  as a vectorized version of the 3D area capacity array  $\{w_{\text{bin}} h_{\text{bin}} d_{\text{bin}}\}$ . In the same way, we define  $D_l(\bar{x}, \bar{y}, \bar{z})$  as a vectorized version of the area density array  $\{\sum_{v_i \in V} \text{Overlap}(\text{bin}_{m,n,l}, v_i)\}$  with  $M \times N$  elements for a fixed  $l$ , and  $C_l$  as a vectorized version of the area capacity array  $\{w_{\text{bin}} h_{\text{bin}} d_{\text{bin}}\}$ .

If we replace the absolute value function by the Huber function, the overlapping functions become differentiable. Thus, the quadratic penalty function with Huber-based local smoothing for the area density constraints in formulation (73) is written as



$$\begin{aligned}
\text{Penalty}_{\text{local}}(\bar{x}, \bar{y}, \bar{z}) &= (D(\bar{x}, \bar{y}, \bar{z}) - C)^T (D(\bar{x}, \bar{y}, \bar{z}) - C) \\
&= \sum_{l=1}^L (D_l(\bar{x}, \bar{y}, \bar{z}) - C_l)^T (D_l(\bar{x}, \bar{y}, \bar{z}) - C_l)
\end{aligned} \tag{78}$$

We would like to apply the Helmholtz global smoothing as mentioned in Section 5.2.1. There are two ways to smooth the density functions: one is to smooth the vector  $D(\bar{x}, \bar{y}, \bar{z})$  by solving a 3D Helmholtz equation; the other is to smooth the vectors  $D_l(\bar{x}, \bar{y}, \bar{z})$  by solving a sequence of 2D Helmholtz equations for  $1 \leq l \leq L$ .

The Helmholtz smoothing can be implemented by solving a linear system [15]. We skip the details of the implementation, and use the symbol  $A_{\varepsilon,3D}$  as the 3D Helmholtz smoothing operator, and  $A_{\varepsilon,2D}$  as the 2D Helmholtz smoothing operator, both of which are constant matrices determined by the structure of the related linear system and the smoothing parameter  $\varepsilon$ . Thus, the globally smoothed area densities and the area capacities are expressed as

$$\begin{aligned}
\widehat{D}(\bar{x}, \bar{y}, \bar{z}) &= \mathbf{A}_{\varepsilon,3D} D(\bar{x}, \bar{y}, \bar{z}) \\
\widehat{C} &= \mathbf{A}_{\varepsilon,3D} C \\
\widehat{D}_l(\bar{x}, \bar{y}, \bar{z}) &= \mathbf{A}_{\varepsilon,2D} D_l(\bar{x}, \bar{y}, \bar{z}) \\
\widehat{C}_l &= \mathbf{A}_{\varepsilon,2D} C_l
\end{aligned} \tag{79}$$

Therefore, the two versions of the quadratic penalty functions with global smoothing are computed as the following,

$$\begin{aligned}
\text{Penalty}_{\text{global,3D}}(\bar{x}, \bar{y}, \bar{z}) &= (\widehat{D}(\bar{x}, \bar{y}, \bar{z}) - \widehat{C})^T (\widehat{D}(\bar{x}, \bar{y}, \bar{z}) - \widehat{C}) \\
&= (\mathbf{A}_{\varepsilon,3D} D(\bar{x}, \bar{y}, \bar{z}) - \mathbf{A}_{\varepsilon,3D} C)^T (\mathbf{A}_{\varepsilon,3D} D(\bar{x}, \bar{y}, \bar{z}) - \mathbf{A}_{\varepsilon,3D} C) \\
&= (D(\bar{x}, \bar{y}, \bar{z}) - C)^T \mathbf{A}_{\varepsilon,3D}^T \mathbf{A}_{\varepsilon,3D} (D(\bar{x}, \bar{y}, \bar{z}) - C)
\end{aligned} \tag{80}$$

and

$$\begin{aligned} \text{Penalty}_{\text{global,2D}}(\bar{x}, \bar{y}, \bar{z}) &= \sum_{l=1}^L \left( \widehat{D}_l(\bar{x}, \bar{y}, \bar{z}) - \widehat{C}_l \right)^T \left( \widehat{D}_l(\bar{x}, \bar{y}, \bar{z}) - \widehat{C}_l \right) \\ &= \sum_{l=1}^L \left( D_l(\bar{x}, \bar{y}, \bar{z}) - C_l \right)^T \mathbf{A}_{\varepsilon,2D}^T \mathbf{A}_{\varepsilon,2D} \left( D_l(\bar{x}, \bar{y}, \bar{z}) - C_l \right) \end{aligned} \quad (81)$$

The gradients of these area density penalty functions can be simply computed by

$$\nabla \text{Penalty}_{\text{global,3D}}(\bar{x}, \bar{y}, \bar{z}) = \left( \mathbf{A}_{\varepsilon,3D}^T \mathbf{A}_{\varepsilon,3D} \left( D(\bar{x}, \bar{y}, \bar{z}) - C \right) \right)^T \nabla D(\bar{x}, \bar{y}, \bar{z}) \quad (82)$$

and

$$\nabla \text{Penalty}_{\text{global,2D}}(\bar{x}, \bar{y}, \bar{z}) = \sum_{l=1}^L \left( \mathbf{A}_{\varepsilon,2D}^T \mathbf{A}_{\varepsilon,2D} \left( D_l(\bar{x}, \bar{y}, \bar{z}) - C_l \right) \right)^T \nabla D_l(\bar{x}, \bar{y}, \bar{z}) \quad (83)$$

These gradient expressions in the discrete area density formulation are consistent with the gradient computation method discussed in Chapter 5. The operators  $\mathbf{A}_{\varepsilon,3D}^T \mathbf{A}_{\varepsilon,3D}$  and  $\mathbf{A}_{\varepsilon,2D}^T \mathbf{A}_{\varepsilon,2D}$  are the twice-smoothing operators, which only need to be computed once and are reused in the computation of all the elements of the gradient.

We notice that the area density penalty function  $\text{Penalty}_{\text{global,2D}}(\bar{x}, \bar{y}, \bar{z})$  is similar to the penalty function defined in Section 6.2, based on the fact that the Huber-based smoothing is similar to the bell-shaped smoothing when the node size is twice as large as the bin size ( $L = 2K$ ). Therefore, applying  $\text{Penalty}_{\text{global,2D}}(\bar{x}, \bar{y}, \bar{z})$  is considered an approximated version of the approach discussed in Section 6.2.

In the experimental results in Section 6.6.2, we will show that both global smoothing techniques,  $\text{Penalty}_{\text{global,3D}}(\bar{x}, \bar{y}, \bar{z})$  with 3D global smoothing and  $\text{Penalty}_{\text{global,2D}}(\bar{x}, \bar{y}, \bar{z})$  with 2D global smoothing, generate similar results.

## 6.4 Mixed-Size 3D Placement Flow

Different from the standard-cell netlist, where all the nodes are of the same height, and the node areas do not differ very much from each other, the mixed-size netlist consists of nodes with large area variations. Although the problem formulation of a mixed-size 3D placement is the same as in Section 6.1, it requires additional considerations to handle the area variation. In order to solve the mixed-size 3D placement problem, we apply the flow in Figure 19, which mainly consists of a 3D floorplanner and our analytical 3D placer for mixed-size designs.

Although our analytical 3D placer presented in Section 6.2 is capable of handling mixed-size designs, large-macro legalization is still a problem. Thus, we use a 3D floorplanner to provide an initial 3D placement to guide the analytical placer (discussed in Section 6.4.1). Our analytical 3D placer works on a given 3D placement. A multiple-stepsize scheme is developed to handle mixed-size netlists efficiently and effectively (described in Section 6.4.2). The 3D global placement solution is rounded in the  $z$ -direction first to snap the movable node to the nearest die, and then the movable nodes are legalized die-by-die using the legalization and detailed placement algorithm in [27].

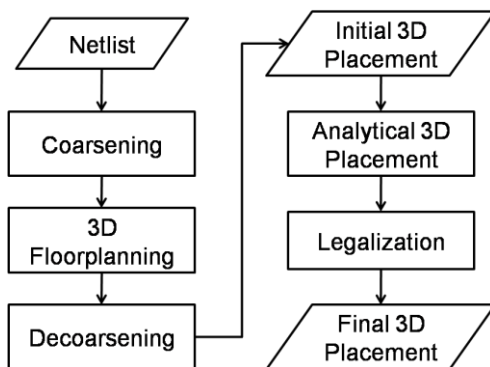
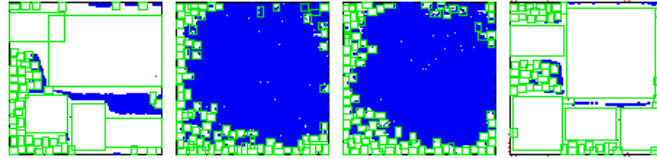


Figure 19. Our mixed-size 3D placement flow

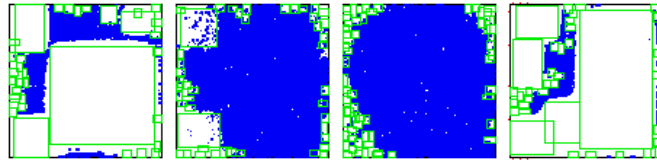
### 6.4.1 3D Floorplan-Based Initial Solution

Before describing the 3D floorplanner, we first analyze the cases where large-macro legalization fails for 3D global placements (as in Figure 20). In each case, the 3D placement of the bottom die (die 1) is shown on the left, and the topmost die (die 4) is shown on the right, where large boxes represent macros and small dots represent standard cells. From these cases, we see that some global placements are difficult to be legalized, even by hand. For example, there are two overlapping macros on die 1 of the *ibm03* case. Although there is enough empty space to hold the overlapping area, these two hard macros cannot be legalized unless one of them is moved to another die. This results in a great displacement of the overall solution.

Thus, a 3D global placement that roughly satisfies the area density constraints may still be difficult to legalize. We would like to start the analytical 3D placer with fixed large macros to prevent illegal solutions. We perform 3D floorplanning [32] on a coarsened netlist after partitioning (e.g., partitioning into 80 to 100 parts). These initial solutions guide the analytical 3D placer to a better placement of the large macros. Furthermore, to prevent these very large macros from being placed on the same die, we may fix the die assignment of very large macros (e.g., macros with width or height greater than 20% of the chip width or chip height, respectively). As demonstrated in Figure 21, with these initial solutions, we gain a higher probability of obtaining a solution that is easy to legalize.

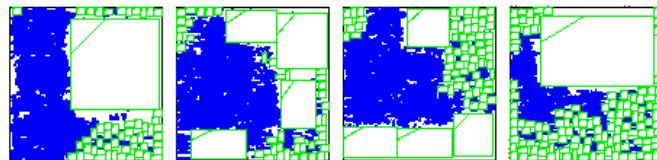


(a) Global placement of ibm03

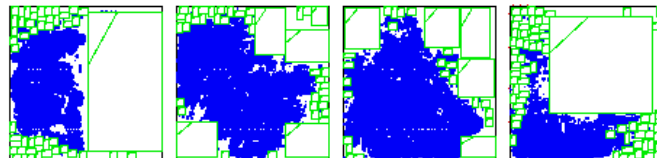


(b) Global placement of ibm06

Figure 20. Global placement results with difficulties in legalization



(a) Global placement of ibm03



(b) Global placement of ibm06

Figure 21. Global placement results guided by 3D floorplanning

### 6.4.2 Analytical Solver with Multiple-Stepsize Scheme

The multilevel analytical placement solver is used for 3D global placement of standard-cell circuits (Section 6.2). We adopt such an analytical solver and present the multiple-stepsize scheme to handle mixed-size designs.

The solution of the unconstrained minimization problem of equation (63) in Section 6.2 is equivalent to the steady solution to the following ordinary differential equation (ODE),

$$\begin{cases} d(x(t), y(t), z(t))/dt = -\nabla F_\mu(x(t), y(t), z(t)) \\ (x(0), y(0), z(0)) \text{ is a given initial placement} \end{cases} \quad (84)$$

where  $F_\mu(x, y, z) = \text{OBJ}(x, y, z) + \mu \cdot \text{Penalty}(x, y, z)$ .

This ODE can be solved by the explicit Euler method, which gives the following iterative scheme:

$$\begin{cases} (x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) = (x^{(k)}, y^{(k)}, z^{(k)}) - \tau \cdot \nabla F_\mu(x^{(k)}, y^{(k)}, z^{(k)}) \\ (x^{(0)}, y^{(0)}, z^{(0)}) \text{ is a given initial placement} \end{cases} \quad (85)$$

The stepsize  $\tau$  has to be small enough to guarantee convergence. The analytical upper bound for  $\tau$  depends on the Hessian of  $F_\mu(x, y, z)$  which is difficult to determine. In practice, the value of  $\tau$  is determined in an adaptive way: an initial stepsize  $\tau$  is tried and then the convergence is checked; if it does not converge, the stepsize is scaled down by a ratio between 0 and 1 (e.g., 0.6) and the trial and error process is repeated.

This scheme works fine for standard cell cases. However, the application of this scheme may cause trouble in mixed-size cases. We observe that if we use the same stepsize for all the variables, the stepsize has to be very small to guarantee convergence. Conversely, the stepsize will cause instability if it is set too large. Thus, we introduce scaling factors for every node according to its area, such that the stepsize ratio  $\tau_i/\tau_j$  is equal to the inversed area ratio  $A_j/A_i$ .

Here we justify the multiple-stepsizescheme by showing its equivalence to the *gradient projection* method for mixed-size placement problems. The following analysis only focuses on a small example of mixed-size linear placement, but it can be extended to a rigor proof for general mixed-size placement problems.

In the mixed-size linear placement example, there are two nodes  $v_1$  and  $v_2$  with widths  $w_1$  and  $w_2$ , respectively, where  $w_2 = 2w_1 = 2w$ . The placement region  $[0, W]$  and the interconnects are shown in Figure 22.



Figure 22. A mixed-size linear placement example

This linear placement problem can be solved by the *quadratic penalty* method,

$$\begin{cases} \text{minimize } \{ \text{OBJ}_2(x_1, x_2) + \mu \cdot \text{Penalty}_2(x_1, x_2) \} \\ \text{increase } \mu \text{ until converge} \end{cases} \quad (86)$$

where  $x_1$  and  $x_2$  are the centers of these two nodes,  $\text{OBJ}_2(x_1, x_2)$  is the total wirelength, and  $\text{Penalty}_2(x_1, x_2)$  is the density penalty function.

The mixed-size problem (86) can be transformed to a uniform-size problem (87), by decomposing  $v_2$  equally into two nodes  $v_3$  and  $v_4$  with an additional constraint  $x_3 + w = x_4$ , as shown in Figure 23.



Figure 23. The corresponding uniform-size linear placement

After decomposition, the problem becomes

$$\begin{cases} \underset{x_3+w=x_4}{\text{minimize}} \{ \text{OBJ}_3(x_1, x_3, x_4) + \mu \cdot \text{Penalty}_3(x_1, x_3, x_4) \} \\ \text{increase } \mu \text{ until converge} \end{cases} \quad (87)$$

where  $\text{OBJ}_3(x_1, x_3, x_4) = \text{OBJ}_2(x_1, (x_3 + x_4)/2)$ .

The *gradient projection* method [69] can be used to solve the constrained optimization problem in (87). Each iterative step consists of a descent step followed by a projection step. The descent step is as follows,

$$\begin{cases} x_1'' = x_1 - \tau \cdot \frac{\partial}{\partial x_1} (\text{OBJ}_3(x_1, x_3, x_4) + \mu \cdot \text{Penalty}_3(x_1, x_3, x_4)) \\ x_3'' = x_3 - \tau \cdot \frac{\partial}{\partial x_3} (\text{OBJ}_3(x_1, x_3, x_4) + \mu \cdot \text{Penalty}_3(x_1, x_3, x_4)) \\ x_4'' = x_4 - \tau \cdot \frac{\partial}{\partial x_4} (\text{OBJ}_3(x_1, x_3, x_4) + \mu \cdot \text{Penalty}_3(x_1, x_3, x_4)) \end{cases} \quad (88)$$

The projection step is to find a feasible solution  $(x_1', x_3', x_4')$  such that it is the feasible point with a minimal distance to the point  $(x_1'', x_3'', x_4'')$ . Formally, it is the solution of the following optimization problem:

$$\begin{aligned} & \underset{(x_1', x_3', x_4')}{\text{minimize}} \quad (x_1' - x_1'')^2 + (x_3' - x_3'')^2 + (x_4' - x_4'')^2 \\ & \text{subject to} \quad x_3' + w = x_4' \end{aligned} \quad (89)$$

The Lagrangian function for problem (89) is

$$L(x_1', x_3', x_4'; \lambda) = (x_1' - x_1'')^2 + (x_3' - x_3'')^2 + (x_4' - x_4'')^2 + \lambda(x_3' + w - x_4') \quad (90)$$

where the optimality condition requires that



$$\begin{cases} \frac{\partial L(x'_1, x'_3, x'_4; \lambda)}{\partial x'_1} = 2(x'_1 - x''_1) = 0 \\ \frac{\partial L(x'_1, x'_3, x'_4; \lambda)}{\partial x'_3} = 2(x'_3 - x''_3) - \lambda = 0 \\ \frac{\partial L(x'_1, x'_3, x'_4; \lambda)}{\partial x'_4} = 2(x'_4 - x''_4) - \lambda = 0 \end{cases} \quad (91)$$

Combining the descent step (88) and the projection step (89), we obtain

$$\begin{cases} x'_1 = x_1 - \tau \cdot \frac{\partial}{\partial x_1} (\text{OBJ}_3(x_1, x_3, x_4) + \mu \cdot \text{Penalty}_3(x_1, x_3, x_4)) \\ \frac{x'_3 + x'_4}{2} = \frac{x_3 + x_4}{2} - \frac{\tau}{2} \cdot \frac{\partial}{\partial x_3} (\text{OBJ}_3(x_1, x_3, x_4) + \mu \cdot \text{Penalty}_3(x_1, x_3, x_4)) \\ \quad \quad \quad - \frac{\tau}{2} \cdot \frac{\partial}{\partial x_4} (\text{OBJ}_3(x_1, x_3, x_4) + \mu \cdot \text{Penalty}_3(x_1, x_3, x_4)) \end{cases} \quad (92)$$

Since  $\text{OBJ}_3(x_1, x_3, x_4) = \text{OBJ}_2(x_1, (x_3 + x_4)/2)$ , we have

$$\frac{\partial \text{OBJ}_3(x_1, x_3, x_4)}{\partial x_1} = \frac{\partial \text{OBJ}_2(x_1, x_2)}{\partial x_1} \quad (93)$$

and 
$$\frac{\partial \text{OBJ}_3(x_1, x_3, x_4)}{\partial x_3} + \frac{\partial \text{OBJ}_3(x_1, x_3, x_4)}{\partial x_4} = \frac{\partial \text{OBJ}_2(x_1, x_2)}{\partial x_2} \quad (94)$$

when  $x_2 = (x_3 + x_4)/2$ .

According to the property of the density penalty functions discussed in 5.1, for the specific placements  $(x_1, x_2)$  and  $(x_1, x_3, x_4)$  where  $x_2 = (x_3 + x_4)/2$ , there exists a density-related function  $E(x)$  such that the gradient of the area density penalty function can be expressed in the following way:

$$\begin{cases} \frac{\partial \text{Penalty}_2(x_1, x_2)}{\partial x_1} = E(x_1 + w/2) - E(x_1 - w/2) \\ \frac{\partial \text{Penalty}_2(x_1, x_2)}{\partial x_2} = E(x_2 + w) - E(x_2 - w) \end{cases} \quad (95)$$

and

$$\begin{cases} \frac{\partial \text{Penalty}_3(x_1, x_3, x_4)}{\partial x_1} = E(x_1 + w/2) - E(x_1 - w/2) \\ \frac{\partial \text{Penalty}_3(x_1, x_3, x_4)}{\partial x_3} = E(x_3 + w/2) - E(x_3 - w/2) \\ \frac{\partial \text{Penalty}_3(x_1, x_3, x_4)}{\partial x_4} = E(x_4 + w/2) - E(x_4 - w/2) \end{cases} \quad (96)$$

Based on the equations (93) (94) (95) (96), we can transform (92) into

$$\begin{cases} x'_1 = x_1 - \tau \cdot \frac{\partial}{\partial x_1} (\text{OBJ}_2(x_1, x_2) + \mu \cdot \text{Penalty}_2(x_1, x_2)) \\ x'_2 = x_2 - \frac{\tau}{2} \cdot \frac{\partial}{\partial x_2} (\text{OBJ}_2(x_1, x_2) + \mu \cdot \text{Penalty}_2(x_1, x_2)) \end{cases} \quad (97)$$

This can be viewed as a descent step in the *gradient descent* method for the unconstrained optimization problem in (86). It indicates that the stepsize ratio between  $v_1$  and  $v_2$  is 2:1, which is inversely proportional to their area ratio  $w:2w=1:2$ .

**Theorem 6.2.** Assume in a mixed-size placement problem, the node area  $A_i = m_i A$  of  $v_i$  is a multiple of a unit area  $A$ . This problem can be transformed to a uniform-size placement problem by decomposing  $v_i$  into  $m_i$  small cells, and the use of additional constraints to glue these decomposed small cells. If the uniform-size placement problem with additional constraints is solved by the *gradient descent* method with step size  $\tau$ , it is

equivalent to solving the mixed-size placement problem by the *gradient descent* method with stepsize  $\tau/m_i$  for cell  $v_i$ .

The proof can be obtained by using the same idea from the previous analysis of the linear placement example, and thus is omitted.

As a result, the stepsize need not be very small for convergence. Conversely, the requirement on the stepsize is less strict, and it helps to implement a stable solver in practice. The effect of the multiple-stepsize scheme will be reported in Table 17.

## 6.5 Multilevel Scheme

To summarize, the optimization problem for our 3D placement method is

$$\begin{aligned}
& \text{minimize} && \sum_{e \in E} (\text{WL}(e) + \alpha_{\text{TSV}} \cdot \text{TSV}(e)) \\
& \text{subject to} && \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (\widehat{D}_k(u, v) - 1)^2 \, dudv = 0 \quad (\text{for } k = 1, \dots, K) \\
& && \int_0^{W_{\text{die}}} \int_0^{H_{\text{die}}} (\widehat{E}_k(u, v) - 1)^2 \, dudv = 0 \quad (\text{for } k = 1, \dots, K - 1)
\end{aligned} \tag{98}$$

where we minimize the weighted sum of wirelength and TSV number, and ensure that the projected area densities on the actual dies and the inter-die layers do not exceed their capacity.

Similar to [15], this nonlinear programming problem can be solved by a multilevel scheme. After clustering, we place the clusters as 2D cells, and then decluster and place them at a finer level. In the context of 3D placement, this multilevel scheme serves both placement and “partitioning.” In one extreme case, if we cluster until the number of clusters equals the number of dies, it becomes a partition. But it lacks the information of die-wise placement, and thus there is no simple way to evaluate the cost for this partition.

However, this information can be obtained in a suitable level with the multilevel placement scheme, which enables the evaluation of “partition” cost. The detailed results are given in Section 6.6.3.

Given the multilevel 3D global placement, the detailed placement first rounds the  $z$  variables to the nearest die. Then die-by-die detailed placement is performed by calling the detailed placer released at [27].

## 6.6 Experimental Results

### 6.6.1 Results on Standard-Cell 3D Placement with Bell-Shaped Area Projection

As described in Chapter 4, experiments are performed on the IBM-PLACE benchmarks [89], which are standard-cell circuits without I/O ports. In the experiments, we assume a 4-die implementation of 3D IC, as we did in Chapter 4. The floorplan size is scaled by dividing the original floorplan by 4, and then enlarging it to obtain 10% white space.

Table 10 presents the results for our multilevel analytical placer with TSV weight  $\alpha_{\text{TSV}} = 10$ . The same subset of circuits in Table 2 and Table 3 from Chapter 4 are used for experiments. Three sets of results are collected for the analytical method, for 1-level placement, 2-level placement and 3-level placement, respectively. The 1-level placement runs the analytical placement engine directly without any clustering, while 2-level or 3-level placements construct a 2-level or 3-level hierarchy by clustering. The wirelength after detailed placement (WL), the number of TSVs (#TSV), and the runtime (RT) are all collected.

In Table 10 we see that with the same TSV weight, 1-level placement achieves the shortest wirelength, while the 3-level placement achieves the fewest TSV numbers. The multilevel analytical 3D placement method and the previous transformation-based 3D placement method are compared. The 1-level placement is compared to “LST (r=10%)” (the best wirelength case), the 2-level placement is compared to “LST (8×8 win)”, and the 3-level placement is compared to “Folding-2” (the best TSV case). From the data shown in Table 2, Table 3 and Table 10, it is clear that the 1-level placement achieves on average a 12% shorter wirelength and 29% fewer TSVs than “LST (r=10%)”; the 3-level placement also achieves on average a 30% shorter wirelength with slightly fewer TSVs than “Folding-2.”

Table 10. Experimental results for the multilevel analytical 3D placement method with  $\alpha_{TSV}=10$

Circuits	1-Level (Flat)			2-Level			3-Level		
	WL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	RT (min)	WL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	RT (min)	WL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	RT (min)
<b>ibm01</b>	0.28	8.11	5.90	0.35	1.63	6.66	0.37	0.87	7.19
<b>ibm03</b>	0.67	16.83	10.77	0.81	4.06	12.09	0.84	2.92	12.31
<b>ibm04</b>	0.85	28.10	20.61	0.99	7.68	24.21	1.11	3.36	24.21
<b>ibm06</b>	1.00	38.48	22.65	1.22	10.03	26.26	1.45	3.40	27.07
<b>ibm07</b>	1.55	53.87	28.97	1.75	18.60	35.06	2.27	4.46	36.00
<b>ibm08</b>	1.68	53.86	26.60	1.93	18.48	29.80	2.36	4.43	30.81
<b>ibm09</b>	1.44	61.79	26.31	1.69	18.62	29.97	2.08	3.37	30.14
<b>ibm13</b>	2.64	110.51	39.18	2.96	36.86	50.41	4.14	4.37	45.29
<b>ibm15</b>	6.72	260.19	96.33	7.28	107.48	141.69	8.74	27.53	114.04
<b>ibm18</b>	9.69	333.02	128.09	10.33	140.53	165.71	12.88	38.35	156.42
<b>geomean</b>	1.00	1.00	1.00	1.16	0.31	1.21	1.38	0.09	1.18

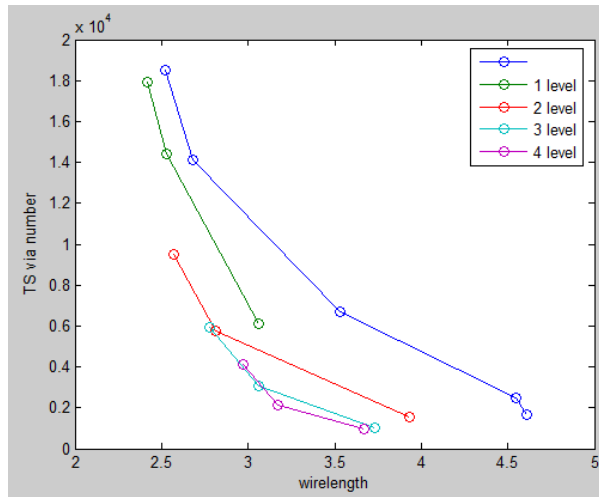
In order to obtain a more complete comparison, we show several trade-off curves for the circuit **ibm01** in Figure 24. The topmost curve is generated by the data in Chapter 4, with transformations LST(10%), LST(20%), LST(8x8 win), Folding-4 and Folding-2 (from left to right). We also plot the curve from the data we produced by varying the

number of clustering levels that we applied in the multilevel scheme, and by varying the weight factor for the TSV number penalty function  $\alpha_{\text{TSV}}$ .

In each curve generated by our method, the smaller TSV number is achieved by increasing the weight factor. But we can see that the increase will also degrade the wirelength quality. In the meantime, if we apply more levels of the multilevel clustering, we also achieve a smaller TSV number, but still guarantee a good wirelength. The results show that the multilevel scheme is an effective way to achieve trade-offs between wirelength and TSV number, which cannot be substituted by the weight factor.

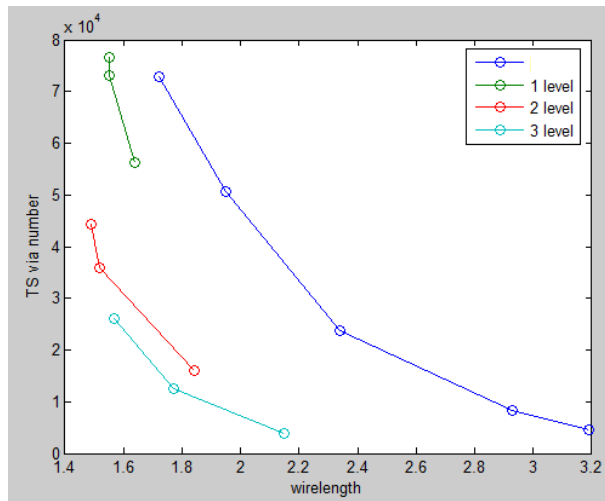
In this example we can obtain a wirelength that is over 30% shorter with a slightly smaller TSV number (around  $0.2 \times 10^4$ ) when compared to the transformation-based approach. Or, we can obtain a 30% smaller TSV number with a 10% shorter wirelength (around  $3 \times 10^6$ ) when compared to that approach.

For other circuits in the benchmarks, we obtain consistent results. Another example of `ibm09` is given in Figure 25.



#level	$\alpha_{TSV} = 0.1$		$\alpha_{TSV} = 1$		$\alpha_{TSV} = 10$	
	WL (x10 <sup>6</sup> )	#TSV	WL (x10 <sup>6</sup> )	#TSV	WL (x10 <sup>6</sup> )	#TSV
1	2.42	17902	2.53	14431	3.06	6124
2	2.57	9507	2.81	5766	3.93	1540
3	2.78	5929	3.06	3070	3.73	1052
4	2.97	4104	3.17	2162	3.67	971

Figure 24. Tradeoff curves for ibm01



#level	$\alpha_{TSV} = 0.1$		$\alpha_{TSV} = 1$		$\alpha_{TSV} = 10$	
	WL (x10 <sup>7</sup> )	#TSV	WL (x10 <sup>7</sup> )	#TSV	WL (x10 <sup>7</sup> )	#TSV
1	1.55	76714	1.55	73178	1.64	56277
2	1.49	44302	1.52	35957	1.84	16040
3	1.57	26072	1.77	12407	2.15	3832

Figure 25. Tradeoff curves for ibm09

### 6.6.2 Results on Standard-Cell 3D Placement with Huber-Based Smoothing

We test the 3D area density formulation as discussed in Section 6.3, using the same set of benchmarks discussed in the previous experimental section. The experimental results with 3D global smoothing are shown in Table 11, Table 12 and Table 13, for 1-level (flat), 2-level, and 5-level placements, respectively. The experimental results with 2D global smoothing are also shown in Table 14.

The results of various 3D analytical placement formulations are visualized in Figure 26. In addition to comparing the 3D and 2D smoothing methods discussed in Section 6.3, we also compare them to the bell-shaped area projection method discussed in Section 6.2. The data points show that the 3D global smoothing method with a moderate clustering level, labeled as “3D smoothing (2-lev),” provides the best placement quality on both HPWL and the TSV number. This is explained as follows:

If the weight of the TSVs is too small (e.g., 0.1), the placer tends to ignore the TSV quality, and generates the data points with similar HPWL quality but with various TSV numbers, as shown in the leftmost five data points in Figure 26. In such cases, clustering helps reduce the unnecessary TSVs without degrading the HPWL quality, because it reduces the inter-die connections at the coarse-level placement. On the other hand, if the clustering levels are deep (e.g., 5 levels), the inter-die connections are reduced too much; therefore the HPWL reduction cannot benefit much from the inter-die connections. The experimental results recommend that we perform a moderate level of clustering in order to obtain the results with fewer TSVs and shorter HPWL.



Table 11. Experimental results with 3D global smoothing of 1-level placement (flat)

Circuit	$\alpha_{TSV} = 0.1$		$\alpha_{TSV} = 10$		$\alpha_{TSV} = 100$		$\alpha_{TSV} = 1000$	
	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )
ibm01	0.26	20.21	0.28	13.50	0.34	2.50	0.38	0.87
ibm03	0.66	35.93	0.66	24.95	0.74	5.36	0.82	3.17
ibm04	0.80	44.24	0.85	33.29	0.98	7.15	1.16	2.75
ibm06	1.00	57.43	1.04	38.15	1.22	7.88	1.45	4.09
ibm07	1.60	76.88	1.69	55.10	1.91	11.78	2.30	5.10
ibm08	1.67	83.20	1.73	60.75	1.90	13.36	2.45	4.57
ibm09	1.47	86.46	1.63	64.46	1.82	12.18	2.21	3.16
ibm13	2.69	143.43	2.80	107.74	3.24	20.73	4.01	5.98
ibm15	7.21	285.10	7.43	236.32	8.05	51.22	9.70	12.08
ibm18	10.02	360.90	10.84	318.76	10.92	93.15	14.08	17.70
geomean	1.59	82.75	1.67	61.30	1.88	13.05	2.27	4.43

Table 12. Experimental results with 3D global smoothing of 2-level placement

Circuit	$\alpha_{TSV} = 0.1$		$\alpha_{TSV} = 10$		$\alpha_{TSV} = 100$		$\alpha_{TSV} = 1000$	
	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )
ibm01	0.26	10.96	0.26	9.26	0.31	2.32	0.36	1.04
ibm03	0.66	18.30	0.65	16.59	0.73	5.29	0.80	3.11
ibm04	0.86	24.28	0.86	21.69	1.00	5.73	1.11	2.95
ibm06	1.08	32.51	1.07	29.23	1.18	7.88	1.42	3.97
ibm07	1.60	41.06	1.57	37.43	1.83	11.67	2.17	4.68
ibm08	1.77	40.34	1.78	36.86	2.04	11.46	2.40	3.94
ibm09	1.57	46.22	1.48	41.21	1.79	10.15	2.03	3.24
ibm13	2.78	77.46	2.70	69.38	3.27	17.68	3.89	5.59
ibm15	7.10	158.12	7.11	146.48	8.48	30.66	9.32	10.52
ibm18	9.76	185.37	9.95	175.95	11.17	61.28	12.96	15.22
geomean	1.63	44.05	1.62	39.77	1.88	10.96	2.16	4.27

Table 13. Experimental results with 3D global smoothing of 5-level placement

Circuit	$\alpha_{TSV} = 0.1$		$\alpha_{TSV} = 10$		$\alpha_{TSV} = 100$		$\alpha_{TSV} = 1000$	
	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )
ibm01	0.30	4.93	0.30	4.50	0.31	2.63	0.38	0.55
ibm03	0.72	9.66	0.71	9.21	0.84	4.00	0.90	2.09
ibm04	0.88	13.52	0.89	12.68	0.99	6.22	1.16	2.31
ibm06	1.12	16.02	1.13	14.59	1.22	7.53	1.46	2.60
ibm07	1.83	19.27	1.81	17.07	1.97	11.31	2.34	4.07
ibm08	1.92	19.99	1.99	18.61	2.15	10.43	2.42	3.65
ibm09	1.80	16.38	1.75	14.55	1.84	8.24	2.11	2.40

<b>ibm13</b>	3.21	32.31	3.21	27.68	3.46	15.89	4.08	4.40
<b>ibm15</b>	8.08	58.77	8.14	55.43	8.46	33.46	9.35	8.76
<b>ibm18</b>	11.26	72.04	11.24	69.31	11.87	46.77	13.19	14.30
<b>geomean</b>	1.82	19.72	1.82	18.10	1.96	10.18	2.26	3.26

Table 14. Experimental results with 2D global smoothing with  $\alpha_{TSV} = 100$

Circuit	1-level placement		2-level placement		5-level placement	
	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )
<b>ibm01</b>	0.34	2.71	0.33	2.01	0.35	1.69
<b>ibm03</b>	0.77	5.75	0.76	5.07	0.83	3.34
<b>ibm04</b>	1.03	8.09	1.01	6.03	1.09	4.43
<b>ibm06</b>	1.21	10.00	1.26	7.13	1.30	6.35
<b>ibm07</b>	1.99	13.26	1.88	12.18	2.06	8.14
<b>ibm08</b>	1.99	14.56	2.14	11.90	2.18	8.49
<b>ibm09</b>	1.92	12.77	1.85	10.64	2.01	5.94
<b>ibm13</b>	3.52	23.59	3.37	18.48	3.81	9.20
<b>ibm15</b>	8.44	54.08	8.85	33.70	8.88	25.46
<b>ibm18</b>	11.27	102.39	11.63	66.19	12.87	37.13
<b>geomean</b>	1.96	14.48	1.96	11.08	2.09	7.50

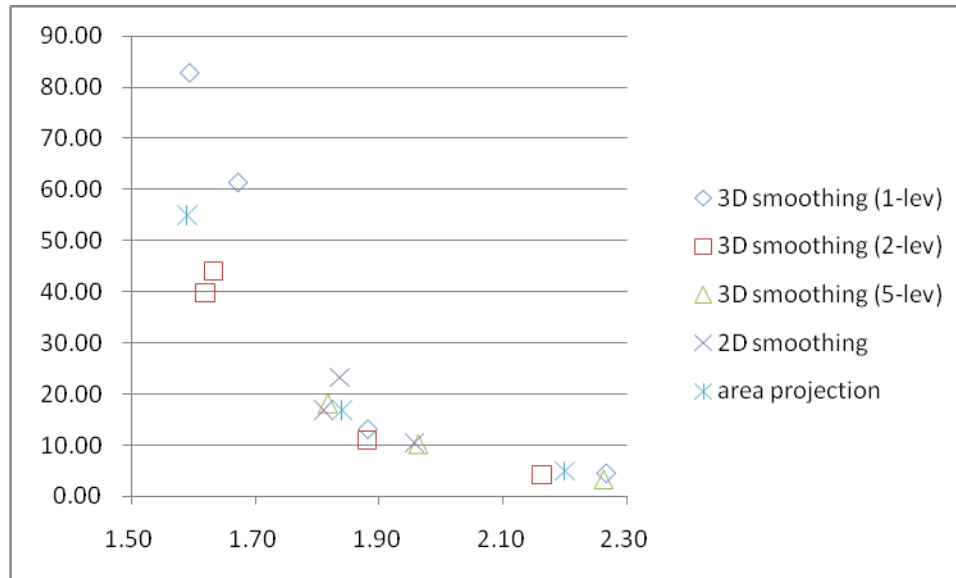


Figure 26. Comparisons of various analytical 3D formulations

A recent 3D analytical placer `ntuplace3d` [48] applies the bell-shaped function, instead of the Huber-based smoothing, to measure the area distribution in the virtual 3D

placement region (as in Section 6.3), but it does not use any global smoothing techniques. We compared their results<sup>‡</sup> with our bell-shaped area projection approach (Section 6.2) and our Huber-base smoothing (Section 6.3) in Table 15. The area projection column is the “3-Level” placement results from Table 10, and the Huber-based smoothing column is the placement results from Table 12 with  $\alpha_{\text{TSV}} = 1000$ . These data show that the Huber-based smoothing achieves more than 20% wirelength reduction on average than the ntplace3d approach with similar amount of TSVs; the Huber-based smoothing also achieves 14% TSV number reduction on average than the area projection approach with slightly shorter wirelength. Although the Huber-based smoothing approach runs slower than the ntplace3d approach, the average computational complexity of Huber-based smoothing approach is  $\Theta(N^{1.07})$ , which is faster than the ntplace3d’s complexity of  $\Theta(N^{1.48})$  asymptotically.

Table 15. Comparisons of the analytical 3D placement algorithms

Circuit	ntplace3d [48]			Area projection (Section 6.2)			Huber-based smoothing (Section 6.3)		
	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	RT (min)	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	RT (min)	HPWL ( $\times 10^7$ )	#TSV ( $\times 10^3$ )	RT (min)
<b>ibm01</b>	0.48	0.75	0.38	0.37	0.87	7.19	0.36	1.04	2.95
<b>ibm03</b>	1.05	2.70	1.07	0.84	2.92	12.31	0.80	3.11	4.72
<b>ibm04</b>	1.43	2.98	1.08	1.11	3.36	24.21	1.11	2.95	6.41
<b>ibm06</b>	1.85	4.04	1.48	1.45	3.40	27.07	1.42	3.97	6.20
<b>ibm07</b>	2.92	5.40	2.37	2.27	4.46	36.00	2.17	4.68	8.64
<b>ibm08</b>	2.85	5.38	3.52	2.36	4.43	30.81	2.40	3.94	11.23
<b>ibm09</b>	2.57	3.44	3.03	2.08	3.37	30.14	2.03	3.24	14.61
<b>ibm13</b>	5.10	4.07	5.40	4.14	4.37	45.29	3.89	5.59	19.62
<b>ibm15</b>	12.21	16.08	15.95	8.74	27.53	114.04	9.32	10.52	46.82
<b>ibm18</b>	17.76	11.27	28.62	12.88	38.35	156.42	12.96	15.22	52.09
<b>geomean</b>	1.28	0.85	0.09	1.00	1.00	1.00	0.99	0.86	0.35

<sup>‡</sup> We obtain the executable of ntplace3d from the authors, and rerun the experiments with our modified benchmarks. We remove the row spacing and double the row height to match the original chip size. Please note that the results in [48] only removed the row spacing but kept the original row height, so they reported shorter wirelengths.

### 6.6.3 Results on Mixed-Size 3D Placement

To evaluate the quality of our analytical 3D placer for mixed-size circuits, experiments are performed on the modified version of the ICCAD'04 mixed-size placement benchmarks [91]. The netlists and the cell sizes remain the same, and the 3D placement regions are scaled from the 2D regions by a factor of  $\sqrt{K}$  on each side, where  $K$  is the number of dies. Thus the total white spaces are not changed. The I/O port locations are also scaled linearly along with the placement regions, and the I/O ports are assumed open at the topmost die (the die with the largest die number). We will set  $K = 4$  for all the experiments in this section.

In this suite of benchmarks, the number of standard cells and nets ranges from 10k to 50k, and there are hundreds of macros in each circuit. The readers may refer to [2][3] for these numbers in detail. Here, we only show the statistics of areas and wirelengths in Table 16. The first sub-column under “2D HPWL” is the half-perimeter wirelength produced by the placer mPL6 [16], where the geometric mean is computed at the last row for comparison with 3D placement results. A breakdown of the wirelengths is found in the following sub-columns: there are, on average, 76% wires connecting between standard cells only, 4% wires connecting between macros only, and 20% wires connecting between standard cells and macros. The area breakdown is also shown afterwards, where the standard cells consume 41% of total area, macros consume 39% and there is 20% white space on average.

Table 16. Statistics of the benchmarks

	2D HPWL				Area	
	total ( $\times 10^6$ )	std ratio	mac ratio	mix ratio	std ratio	mac ratio
<b>ibm01</b>	2.20	0.655	0.053	0.292	0.37	0.43
<b>ibm02</b>	4.73	0.670	0.066	0.264	0.25	0.55
<b>ibm03</b>	6.81	0.726	0.013	0.261	0.3	0.5
<b>ibm04</b>	7.31	0.697	0.046	0.257	0.38	0.42
<b>ibm05</b>	9.36	1.000	0.000	0.000	0.80	0.00
<b>ibm06</b>	5.73	0.830	0.002	0.168	0.35	0.45
<b>ibm07</b>	9.85	0.830	0.010	0.160	0.44	0.36
<b>ibm08</b>	11.67	0.659	0.107	0.234	0.39	0.41
<b>average</b>	<i>6.50</i>	<b>0.76</b>	<b>0.04</b>	<b>0.20</b>	<b>0.41</b>	<b>0.39</b>

Before we show the experimental results, we shall first present the data in Table 17 on the effect of the multiple-stepsize scheme discussed in Section 6.4.2. We compare the multiple-stepsize scheme to the single-stepsize scheme on 2D placements ( $K=1$ ). The different schemes are tested as an additional run on the same given placements, and we expect that the final placement will have the same or a slightly better wirelength (the normalized quality in Table 17). Three implementations are tested: the first one is a single-stepsize scheme with a moderate increasing penalty factor  $\mu$ ; the second one is a single-stepsize scheme with an aggressive increasing penalty factor; and the last one is a multiple-stepsize scheme with an aggressive increasing penalty factor. The quality of the results are normalized, which is equal to the final HPWL divided by the given HPWL, and the number of iterations spent is also reported. The results show that if only single-stepsize is used, it takes more time for the adaptive scheme to search for a small enough stepsize, or the quality degrades significantly. The multiple-stepsize scheme helps the stability and the runtime, and maintains the best quality with the fewest number of iterations to converge, compared to the other two single-stepsize schemes.

Table 17. Effect of the multiple-stepsize scheme

	moderate single stepsize		aggressive single stepsize		aggressive multiple stepsizes	
	quality	#iter	quality	#iter	quality	#iter
<b>ibm01</b>	1.01	45	1.01	45	1.01	45
<b>ibm02</b>	1.00	195	1.13	200	0.97	60
<b>ibm03</b>	0.99	200	1.12	135	0.97	70
<b>ibm04</b>	0.99	200	1.06	140	0.97	75
<b>ibm05</b>	0.99	150	0.99	70	0.99	70
<b>ibm06</b>	1.00	40	1.00	40	1.00	40
<b>ibm07</b>	1.00	200	1.09	145	0.98	65
<b>ibm08</b>	1.03	200	1.04	200	0.97	70
<b>geomean</b>	<b>1.01</b>	<b>112.88</b>	<b>1.06</b>	<b>94.17</b>	<b>0.99</b>	<b>57.58</b>

Experimental results for the two modes of our 3D placer, “Pseudo 3D” and “3D (mac fixed),” are summarized in Table 18, as is the folding method in [34]. As pointed out in Section 6.4.1, large macros create troubles for the legalization. Thus, 3D floorplanning is performed on the coarsened netlist (100 nodes) before 3D placement. The “large macros” in the experiments are the macros whose width or height is greater than 20% of the chip width or 20% of the chip height, respectively. Both modes start with 3D floorplan-based initial solutions. The “pseudo 3D” mode fixes the large macros, disables the movement in the  $z$  direction, and runs the 3D placement only for standard cells and small macros in the  $(x, y)$  direction. And the “3D (mac fixed)” mode fixes the large macros, but allows the movement of standard cells and small macros in both the  $(x, y)$  direction and the  $z$  direction.

Table 18. 3D placement results

	Pseudo 3D				3D (mac fixed)				Folding (Chapter 4)	
	gp-WL ( $\times 10^6$ )	dp-WL ( $\times 10^6$ )	#TSV ( $\times 10^3$ )	RT (min)	gp-WL ( $\times 10^6$ )	dp-WL ( $\times 10^6$ )	#TSV ( $\times 10^3$ )	RT (min)	dp-WL ( $\times 10^6$ )	#TSV ( $\times 10^3$ )
ibm01	1.47	1.63	2.30	1.55	1.49	1.64	2.39	2.82	1.89	1.88

ibm02	4.12	3.90	3.31	4.04	3.83	3.79	4.98	5.81	4.12	3.23
ibm03	5.46	5.24	4.23	4.01	4.89	4.70	4.36	9.16	failed	failed
ibm04	6.04	5.88	4.90	4.72	5.72	5.56	5.46	9.33	6.80	3.36
ibm05	5.53	5.40	13.98	4.43	5.72	5.65	8.26	5.72	6.92	9.41
ibm06	5.02	5.09	5.62	11.90	4.77	4.86	4.87	9.02	failed	failed
ibm07	7.98	8.03	6.78	7.65	7.11	7.46	7.28	33.49	9.26	5.11
ibm08	9.65	10.00	8.95	10.68	8.12	8.48	9.40	18.25	11.79	7.01
geomean	5.06	<b>5.07</b>	5.43	5.17	4.73	<b>4.80</b>	5.45	9.03	-	-

The wirelength after global placement (gp-WL), the wirelength after detailed placement (dp-WL), the number of TSVs (#TSV), and the total runtime (RT) are all reported. Both modes produce a similar amount of TSVs. Compared to the “pseudo 3D” mode, the “3D (mac fixed)” mode reduces wirelength by 5.3% on average, by allowing the movement of small objects in the z-direction. Compared to the folding method, the “3D (mac fixed)” mode reduces wirelength by 18% on average with 35% more TS vias. Compared to the 2D placement, the “3D (mac fixed)” mode provides a 27% wirelength reduction on average for these mixed-size benchmarks.

## 6.7 Conclusions

In this chapter a multilevel analytical 3D placement algorithm is discussed, which considers the wirelength minimization, overlap removal, die assignment and TSV number control by optimizing a penalized objective function. The overlap removal and die assignment are handled by a density penalty function, whose minimizer is guaranteed to be a legal placement. Experimental results demonstrate that the multilevel scheme is effective for controlling the TSV number.

We also discussed several techniques for enabling an analytical 3D placement to support mixed-size designs. The multiple-stepsizes method gains efficiency by allowing as

large a stepsize as possible for each standard cell, while enabling large macros updated with small stepsizes for stability. The 3D floorplanning is used to generate initial solutions for very large macros and gives a higher possibility of obtaining a legalized solution. The experimental results show that the 3D mixed-size placement is able to reduce wirelength by 27% compared to 2D placements. The results also show that the 3D mixed-size placement achieves 5.3% shorter wirelength than the pseudo 3D placement with a similar amount of TSVs.



## Chapter 7

### Thermal-Aware Cell and TSV Co-Placement for 3D ICs

Existing thermal-aware 3D placement methods assume that the temperature of 3D ICs can be optimized by properly distributing the power dissipations, and ignoring the heat conductivity of TSVs. However, our study indicates that this is not exactly correct. While considering the thermal effect of TSVs during placement appears to be quite complicated, we are able to prove that when the TSV area in each bin is proportional to the lumped power consumption in that bin, together with the bins in all the dies directly above it, the peak temperature is minimized. Based on this criterion, we implement a thermal-aware 3D placement tool. Compared to the methods that prefer a uniform power distribution that only results in an 8% peak temperature reduction, our method reduces the peak temperature by 34% on average with even slightly less wirelength overhead. These results suggest that considering thermal effects of TSVs is necessary and effective during the placement stage. At the time when this dissertation was written, to the best of our knowledge, this was the first thermal-aware 3D placement tool that directly takes into consideration the thermal and area impact of TSVs.

#### 7.1 Introduction

There are several works that address the thermal issue during 3D placement. The work in [43] applies a force-directed method with thermal forces to move nodes away from high temperatures. The transformation-based 3D placement [34] relieves the thermal issues at the legalization stage, where it is preferable to place hot nodes close to

the heat sink. The partitioning-based 3D placement [45] uses net weights to shorten the high switching nets to reduce power, and uses pseudo nets to pull hot nodes to the heat sink to reduce temperature. The work in [82] models and minimizes the unevenness of thermal distribution, in addition to minimizing the wirelength and the unevenness of node area distribution. A detailed survey of 3D physical design can be found in [29][31].

It is well known that for 2D ICs, properly distributed power dissipations (e.g., uniform distribution) can result in low temperatures. Most of the aforementioned work simply extends this conclusion to 3D and still focuses on properly distributing power dissipations for temperature reduction. However, as detailed in Section 7.2, uniform power distribution is no longer a good heuristic for temperature reduction in 3D ICs. Since TSVs are the major channel for heat flow, their distribution also has a significant impact on the temperature. A survey on concurrent TSV planning within thermal-aware 3D floorplanning and 3D routing is given in [83]. Unfortunately, none of the existing work in thermal-aware 3D placement takes the thermal effect of TSVs into consideration, mainly due to the high complexity of such a practice.

In this chapter we discuss a thermal-aware 3D placement method that considers both the thermal effect and the area impact of TSVs. We first devise a simple criterion to guide the placement of TSVs for achieving the lowest temperature. Based on the assumption that the dielectric layer is an ideal heat insulator, *we are able to prove that when the TSV area in each bin is proportional to the lumped power consumption in that bin, together with the bins in all the dies directly above it, the peak temperature is minimized.* We then use this result to guide our analytical 3D placer. Experimental

results show that compared to methods that prefer a uniform power distribution (which only results in an 8% peak temperature reduction), our method reduces the peak temperature by 34% on average with even slightly less wirelength overhead. At this time, and to the best of our knowledge, this is the first thermal-aware 3D placement tool that directly takes into consideration the thermal and area impact of TSVs.

The remainder of the chapter is organized as follows. Section 7.2 provides the motivation for our work. Section 7.3 discusses the optimal distribution of TSVs to minimize the temperature, which is then integrated into a 3D placement framework in Section 7.4. Experimental results are given in Section 7.5, and concluding remarks are given in Section 7.6.

## **7.2 Motivation**

The stack-die structure has dramatically increased power density compared to conventional 2D ICs, and thus threatens the thermal reliability of 3D ICs. In addition, the low thermal conductivity of the dielectric layers in face-to-back bonding dies prohibits the heat from flowing vertically. Accordingly, as pointed out in [44], TSVs are the major channels for vertical heat flow.

Such an observation results in the fundamental difference between the thermal-aware placement for 2D ICs and for 3D ICs. In 2D placement, by properly distributing the power dissipations across the chip, heat can flow uniformly through the entire substrate to the heat sink, and the temperature can be minimized [79]. However, in 3D ICs, it is the correlation between the distributions of the TSVs and the power density that has a direct impact on the temperature. For example, compare the two artificial placement results

with the relative power values shown in Figure 27. In Figure 27(a), the power distribution is uniform while the TSVs are clustered in the center; while in Figure 27(b), the power distribution is non-uniform with 2 to 8 times higher power density in some regions than shown in the previous case, and the TSVs are clustered proportional to the regional power density. The corresponding temperature maps are shown in Figure 27(c) and (d), respectively, assuming a 4-die 3D chip with 6W power in a  $1.5\text{mm}^2$  area with about 1200 TSVs per die, where the 3D technology parameters for temperature evaluation are the same as those in Section 7.5. From this artificial example, we can see that the locations of the TSVs play a very important role in the thermal integrity of 3D ICs.

As expected, it is suboptimal for existing thermal-aware 3D placement to be targeted at distributing power dissipations and neglect the thermal effect of TSVs. To improve this, a naïve approach would be to compute the optimal locations of the TSVs that can result in the minimum temperature during each iteration of placement. However, this will result in an optimization-in-the-loop with significant runtime overhead. Since thermal-aware placement mainly targets large designs, this method is less practical. On the other hand, if we adjust the locations of the TSVs after placement is done to minimize the temperature, it will bring about significant wirelength overhead because these TSVs are also part of the signal nets. We will address this dilemma in this chapter.

There are many different 3D integration technologies as introduced in Section 1.1, including face-to-face bonding, face-to-back bonding, via-first, via-middle, via-last. Different techniques can have totally different thermal models. Here, we focus on the face-to-back bonding with via-last technology, where the TSVs are fabricated separately

on each die through both the silicon layer and the metal layers. In addition, although it is possible to insert additional thermal TSVs [44] after placement to further suppress the temperature, it brings in extra area overhead. In this chapter we focus on exploring the opportunities of temperature reduction by utilizing the signal TSVs in 3D placement after TSV insertion (Section 3.3.4). Our experimental results show that signal TSVs alone can already reduce the temperature significantly, with minimal wirelength or runtime overhead.

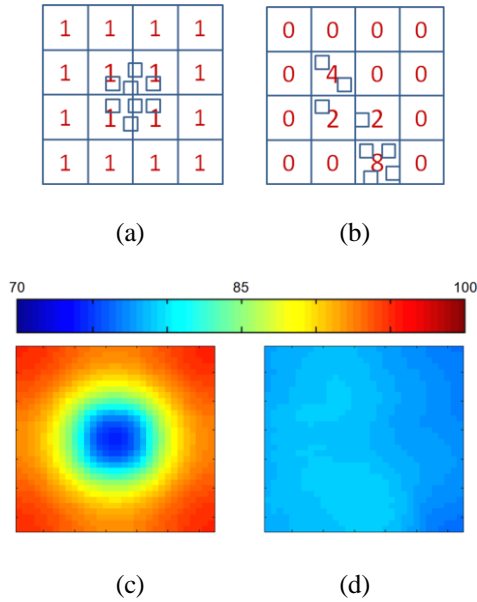


Figure 27. Uniform power with clustered TSVs vs. consistent TSV and power distribution

### 7.3 Properties of a Thermally-Optimal TSV Distribution

As discussed in Section 7.2, the fundamental problem in thermal-aware placement can be stated as follows: Given a power distribution, what will be the optimal distribution of TSVs so that the temperature is minimized? While this problem seems to be complicated, we will show that the answer is surprisingly simple. We can derive an

analytical solution without involving any optimization tools. For simplicity of presentation, Table 19 summarizes the key notations used in this section.

Table 19. Major notations

$B$	thermal conductance matrix
$B_0$	thermal conductance matrix without TSVs
$T$	vectorized temperature map
$P$	vectorized power map
$t_i (t_i^j)$	the temperature in bin $i$ for single-die case (in bin $i$ , die $j$ for multi-die case)
$p_i (p_i^j)$	the power in bin $i$ for single-die case (in bin $i$ , die $j$ for multi-die case)
$A_{tot} (A_{tot}^j)$	total TSV area for single-die case (in die $j$ for multi-die case)
$a_i (a_i^j)$	TSV area in bin $i$ for single-die case (in bin $i$ die $j$ for multi-die case)
$M_i (M_i^j)$	stamping matrix of the lumped TSV in bin $i$ for single-die case (in bin $i$ , die $j$ for multi-die case)
$n$	number of bins in each die
$K$	number of dies
$g_{TSV}$	thermal conductance of a unit area TSV

To start, we assume a steady-state analysis to calculate the temperature, where the chip is thermally modeled as a resistive network. We also lump the TSVs in each bin as a thermal conductor, with its conductance proportional to the total TSV area. The power-temperature relation can be expressed as

$$BT = P \quad (99)$$

A two-die example of the thermal resistive network is illustrated in Figure 28, where the nodes (labeled with numbers) are connected by thermal conductors (labeled with subscripted symbols), and the bin numbers are in large gray colors. Take node 3 (bin 3, die 1) for example, the power-temperature relation is expressed as

$$g_{1,3}(t_3^1 - t_1^1) + g_{3,4}(t_3^1 - t_4^1) + g_{3,7}(t_3^1 - t_3^2) = p_3^1 \quad (100)$$

Thus, the network can be written in a matrix form as equation (99), where each row corresponds to one node.

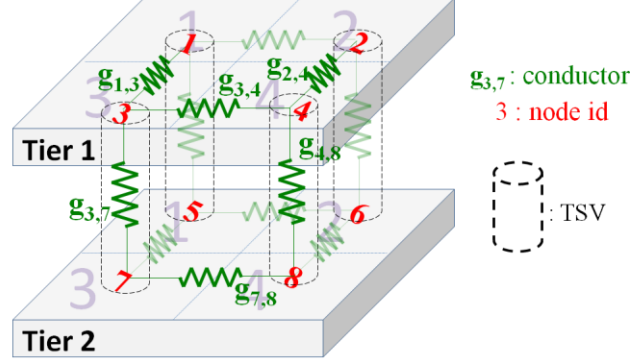


Figure 28. A two-die example of the thermal resistive network

If we treat TSV size as variables, the thermal conductance matrix  $B$  of the network can be expressed in a parameterized form as

$$B = B_0 + \sum_{i=1}^n \sum_{j=1}^K g_{TSV} \cdot a_i^j M_i^j \quad (101)$$

where  $B_0$  is the constant thermal conductance matrix without TSVs, and the variable  $a_i^j$  is the total area of a lumped TSV in bin  $i$ , die  $j$ . The stamping matrix  $M_i^j$  indicates the connectivity of a lumped TSV from bin  $i$ , die  $j$  to bin  $i$ , die  $j+1$ .<sup>§</sup> If we denote  $k_1$  and  $k_2$  to be the node id corresponding to bin  $i$ , die  $j$  and bin  $i$ , die  $j+1$ , in the thermal resistive network, then  $M_i^j(k_1, k_2) = M_i^j(k_2, k_1) = -1$ ,  $M_i^j(k_1, k_1) = M_i^j(k_2, k_2) = +1$ , and all the other elements in  $M_i^j$  are zeros.

---

<sup>§</sup>  $M_i^K$  is the stamping matrix for the lumped TSV in the last die connecting to the heat sink.

Again, take node 3 for example. Let  $b_{3,7}$  be the thermal conductance between node 3 and node 7 when there is no TSVs,  $g_{TSV}$  be the conductance of a unit-area TSV, and the variable  $a_3^1$  be the area of a lumped TSV in bin 3. The conductance becomes

$$g_{3,7} = b_{3,7} + g_{TSV} \cdot a_3^1 \quad (102)$$

In this example, the stamping matrix  $M_3^1$  only has non-zero elements  $M_3^1(3,3) = M_3^1(7,7) = +1$ , and  $M_3^1(3,7) = M_3^1(7,3) = -1$ .

Now, we can mathematically state the problem for optimal TSV placement as

$$\begin{aligned}
 \text{(P1)} \quad \min \quad T_L &= \left\| \left( B_0 + \sum_{i=1}^n \sum_{j=1}^K g_{TSV} \cdot a_i^j M_i^j \right)^{-1} P \right\|_{\infty} \\
 \text{s.t.} \quad \sum_{i=1}^n a_i^j &= A_{tot}^j \quad 1 \leq j \leq K \\
 a_i^j &\geq 0 \quad 1 \leq i \leq n, 1 \leq j \leq K
 \end{aligned} \quad (103)$$

where  $A_{tot}^j$  is the total area of the TSV connecting die  $j$  and die  $j+1$ , and is determined once the floorplanning is done. The infinity norm is defined as  $\|\mathbf{x}\|_{\infty} = \max\{|x_1|, |x_2|, \dots, |x_n|\}$ . The objective function is obtained by simply substituting (101) into (99). The two constraints are also self-evident: the total TSV area in each die is a fixed number, and the lumped TSV area in each bin should be non-negative. Note that we have relaxed the constraint that the TSV area  $a_i^j$  in each bin should be discrete. Accordingly, the TSV areas mentioned in the theorems and corollaries described below should be rounded.



Problem **(P1)** is non-linear in nature. Integrating nonlinear optimization engines in a placement tool directly would be impractical due to the high complexity.

Before we directly tackle **(P1)**, we resort to a simpler version of the problem: For a one-die 3D IC\*\* with a given power distribution, what will be the optimal locations of TSVs so that the temperature is minimized?

In this case, each TSV is directly connected to the heat sink. As such, **(P1)** can be rewritten as

$$\begin{aligned}
 \text{(P2)} \quad \min \quad T_L &= \left\| \left( B_0 + \sum_{i=1}^n g_{TSV} \cdot a_i M_i \right)^{-1} P \right\|_{\infty} \\
 \text{s.t.} \quad \sum_{i=1}^n a_i &= A_{tot} \\
 a_i &\geq 0 \quad 1 \leq i \leq n
 \end{aligned} \tag{104}$$

where  $M_i$  is the stamping matrix for the TSV in bin  $i$ ,  $a_i$  is the total TSV area in bin  $i$ , and  $A_{tot}$  is the total TSV area.

At first look, this problem is still non-linear and difficult to solve. But intuitively we should place more TSVs into the bins with higher power density to provide lower impedance to thermal ground. This leads to the conjecture that the optimal TSV area  $a_i^*$  in bin  $i$  should be proportional to the power consumption  $p_i$ . This conjecture is indeed correct, as stated in the following theorem:

**Theorem 7.1 (Single-Die Case).** To minimize the peak temperature, the TSV area in bin  $i$  should be proportional to the power in that bin; i.e., the optimal solution of problem (P2) is

---

\*\* This case is where TSVs are only used to connect the IC and the heat sink.

$$a_i^* = A_{tot} \cdot p_i / \sum_{i=1}^n p_i \quad (105)$$

In the interest of conciseness, we will only outline the proof for the theorem. From the fact that TSVs are the major vertical heat flow channel ( $g_{TSV} a_k \gg b_{k,l}$  where  $b_{k,l}$  is the inter-die conductance without TSVs), we can get

$$\sum_i p_i \approx \sum_j g_{TSV} a_j t_j = g_{TSV} a^T T \quad (106)$$

where  $a = [a_1, a_2, \dots, a_n]^T$ . Based on Hölder's inequality, we have

$$a^T T \leq \|a\|_1 \|T\|_\infty \quad (107)$$

Combining (106) and (107), we have

$$g_{TSV} \|T\|_\infty \geq \sum_{i=1}^n p_i / \|a\|_1 = \sum_{i=1}^n p_i / \sum_{i=1}^n a_i = \sum_{i=1}^n p_i / A_{tot} \quad (108)$$

In order for  $\|T\|_\infty$  to attain the above minimum, the inequalities in (107) must become equality. According to Holder's inequality, such a condition is

$$T_1 = T_2 = \dots = T_n \quad (109)$$

Substitute it back to (106), and we can get

$$p_1 / a_1 = p_2 / a_2 = \dots = p_n / a_n \quad (110)$$

which, along with the second constraints in **(P2)**, yields

$$a_i = A_{tot} p_i / \sum_{i=1}^n p_i \quad (111)$$

Note that in the above theorem, we neglected the fact that the total TSV area in each area is discrete, that the dielectric layer is not an ideal thermal insulator, and that the total TSV area allocated in each bin cannot exceed the area of that bin. In reality, the optimal

condition needs to be tailored to fit into these constraints. We can also easily derive a corollary based on this theorem.

**Corollary 7.1.** When the TSVs are placed proportional to the power consumption in each bin, the temperature in each bin is identical, i.e.,

$$t_i^* = \sum_{i=1}^n p_i / (g_{TSV} A_{tot}) \quad (112)$$

Corollary 7.1 has a particularly important meaning, as it allows us to generalize Theorem 7.1 (which is limited to the single-die case) to the general multi-die cases. Take a two-die case as an example. According to Theorem 7.1, as long as we place the TSVs connecting the bottom die and the package proportional to the power density in each bin, the temperature is minimized in the bottom die. Now, since such optimized temperature distribution is also uniform based on Corollary 7.1, the bottom die can be treated as ground. Accordingly, we can again apply Theorem 7.1 to the top die, and place the TSVs connecting the top die and the bottom die according to the power distribution. Such an observation leads to the following theorem.

**Theorem 7.2 (Multi-Die Case).** If we denote the bottom die (with connection to the heat sink) as die K, and the top die as die 1, then to minimize the temperature, the TSV area in bin i of die j connecting to die j+1 should be proportional to the lumped power in bin i of die 1, 2, ..., j. In other words, the optimal solution of problem (P1) shall satisfy

$$(a_i^j)^* = A_{tot}^j \cdot \sum_{k=1}^j p_i^k / \sum_{i=1}^n \sum_{k=1}^j p_i^k \quad (113)$$

The proof can be derived based on the induction on the number of dies using Theorem 7.1; this is because the optimized temperature in a die is uniform and can be

treated as thermal ground to further optimize upper dies. Figure 29 shows a simple two-die ( $K = 2$ ) example to illustrate the theorem, where each die is divided into four bins ( $n = 4$ ).

Similar to Corollary 7.1 for the single-die case, we also have the following corollary for the multi-die case.

**Corollary 7.2.** When the TSVs in each die are placed proportional to the lumped power consumption in each bin and the same bins in all the dies above, then each die shall have a uniform temperature distribution. The temperature in the die  $j$  can be expressed as

$$(t_i^j)^* = \sum_{i=1}^n \sum_{k=1}^j p_i^k / (g_{TSV} A_{tot}^j) \quad (114)$$

To summarize this section, we would like to point out that all the theorems and corollaries are based on the assumption that TSVs are much more effective in conducting heat than the dielectric layer. And accordingly, we have treated the dielectric layer as an ideal heat insulator. In reality this is not correct, and thus the theorems are only an approximation. However, our experimental results show that they work pretty well.

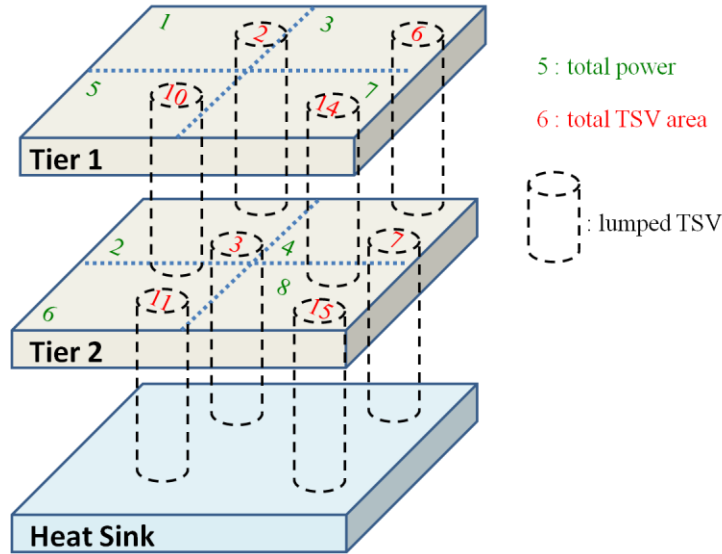


Figure 29. A two-die example to illustrate Theorem 7.2

## 7.4 Thermal-Aware 3D Placement

Our 3D placement flow is similar to the one in [54], but in this section we mainly focus on the 3D placement step in the TSV co-placement flow. We assume the die assignment of each cell is given, either by manually partitioning or automatic partitioning. An automatic partitioning method by 3D floorplanning will be explained in Section 7.4.2. The 3D placement step is called after 3D net splitting and TSV insertion.

### 7.4.1 Thermal-Aware Cell/TSV Co-Placement

Based on the optimality condition in Theorem 7.2, we are able to effectively reduce the temperature during the 3D placement step by an analytical method, as in the following formulation,

$$\begin{aligned}
(\mathbf{P3}) \quad & \min \quad \text{HPWL}(\mathbf{x}, \mathbf{y}) + \beta \cdot \text{COST}(\mathbf{x}, \mathbf{y}) \cdot \frac{\|\nabla \text{HPWL}^{(\text{init})}\|}{\|\nabla \text{COST}^{(\text{init})}\|} \\
& s.t. \quad D_b(\mathbf{x}, \mathbf{y}) = C_b \quad \text{for each area bin } b
\end{aligned} \tag{115}$$

$(\mathbf{x}, \mathbf{y})$  is the placement variable;  $D_b(\mathbf{x}, \mathbf{y}) = C_b$  is the area density constraints for overlap removal;  $\text{HPWL}(\mathbf{x}, \mathbf{y})$  is the total half-perimeter wirelength as the objective function; the TSV distribution cost  $\text{COST}(\mathbf{x}, \mathbf{y})$  measures the “distance” between the current solution and a thermally optimal distribution, and  $\beta$  is a user-defined parameter for trade-offs between wirelength quality and temperature reduction. The TSV distribution cost is also normalized by a factor as the ratio between the gradient norm of the initial HWPL function and the gradient norm of the initial COST function.

Please refer to Chapters 7, 10 and 11 in [65] for the algorithms that solve problem  $(\mathbf{P3})$  by the quadratic penalty method when  $\beta = 0$ , and refer to [23] for the parameter tunings when  $\beta > 0$ . In this section we focus on the definition of the TSV distribution cost function  $\text{COST}(\mathbf{x}, \mathbf{y})$ .

The TSV distribution cost is constructed with the property that  $\text{COST}(\mathbf{x}, \mathbf{y}) = 0$  if and only if the optimality condition in Theorem 7.2 is satisfied. In detail, the cost is constructed as the following:

Let  $N_i^j$  be the number of TSVs in the bin  $i$ , die  $j$ , and we assign a *negative “power”* value  $\tilde{p}_{TSV}^j$  to all the TSVs on die  $j$ . The negative power value is defined as

$$\tilde{p}_{TSV}^j = (-1) \cdot \frac{\sum_{i=1}^n \sum_{k=1}^j P_i^k}{\sum_{i=1}^n N_i^j} \tag{116}$$

Under this assignment, the total negative power of the TSVs in the bin  $i$ , die  $j$  is

$$\tilde{p}_i^j = \tilde{p}_{TSV}^j \cdot N_i^j \quad (117)$$

Therefore, the total TSV power and the lumped cell power in the bin  $i$ , die  $j$  is

$$\begin{aligned} \sum_{k=1}^j P_i^k + \tilde{p}_i^j &= \sum_{k=1}^j P_i^k + \tilde{p}_{TSV}^j \cdot N_i^j \\ &= \sum_{k=1}^j P_i^k + (-1) \cdot \sum_{i=1}^n \sum_{k=1}^j P_i^k \cdot N_i^j / \sum_{i=1}^n N_i^j \\ &= \sum_{k=1}^j P_i^k + (-1) \cdot \sum_{i=1}^n \sum_{k=1}^j P_i^k \cdot a_i^j / A_{tot}^j \end{aligned} \quad (118)$$

It is obvious that this amount of power value is equal to zero if and only if the TSVs are optimally distributed, as in Theorem 7.2. Thus, the TSV distribution cost can be defined as

$$COST(x, y) = \sum_{i=1}^n \sum_{j=1}^K \left( \sum_{k=1}^j P_i^k(x, y) + \tilde{p}_i^j(x, y) \right)^2 \quad (119)$$

which is a sum of squares of the total TSV power and the lumped cell power in each bin. This quadratic penalty method is an easy-to-use, common method in engineering practice to satisfy the equality constraints. Since the existence of a solution that satisfies both the area density constraint and the TSV distribution constraint is not easy to determine, we only penalize the COST function by a finite number  $\beta$  instead of pushing it to  $+\infty$ .

#### 7.4.2 Overall Thermal-Aware Placement Flow

To obtain the die assignment, we use a 3D floorplanner [32] on a coarsened circuit, which is produced by doing an 80-way partition using hMetis [51]. The number of partitions for floorplanning is determined empirically, so that the runtime is under control

for circuits of various sizes. The die assignment obtained in the 3D floorplan is locked before 3D placement.

Given the die assignment, we split the circuit into dies, and insert one TSV per 3D net. The cells and inserted TSVs are co-placed by solving problem **(P3)** with a modified placement engine. Finally, the cells and TSVs are legalized, die-by-die, to complete the flow using the XDP [27] detailed placement engine.

## 7.5 Experimental Results

We implement the algorithm in C++ and run our experiments on an Intel Xeon 2.0 GHz machine with Linux. The experiments are performed on seven open-source IP cores in the IWLS 2005 benchmarks [97]. The circuits are summarized in Table 20, where the utility rate (Util.) is the total cell area divided by the total chip area.

We synthesize the circuits with a standard cell library for the MIT Lincoln Lab 130nm 3D SOI technology. The target 3D technology is a 4-die 3D IC, with TSV size  $6\mu m \times 6\mu m$  and TSV pitch  $12\mu m \times 12\mu m$ . The 3D chip temperature is measured by the compact model in [80], assuming that the height of the silicon layer is  $300\mu m$  on the bottom die and  $25\mu m$  on the other dies.

The placement area is set as a square with 20% to 28% whitespace in total, and the I/O pins are placed uniformly along the boundaries in alphabetical order. The power dissipation of each cell is generated as follows: The circuit is partitioned into eight parts by hMetis. Each part is assigned a random number between 0 and 1 as a relative power number. These relative numbers are scaled to power values such that the overall power



density is on the order of a magnitude of  $1\text{ W/mm}^2$ , which is the projected power density for high-performance chips at the 14nm generation by ITRS [96].

Table 20. Circuit statistics

Circuit	#Cell	#TSV	Power (W)	Cell Area (mm <sup>2</sup> )	Util.
<b>aes_core</b>	20397	1362	1.31	1.31	0.80
<b>wb_conmax</b>	25883	2166	1.87	1.87	0.80
<b>ethernet</b>	49332	3782	4.46	4.46	0.78
<b>des_perf</b>	69494	3678	5.28	5.28	0.77
<b>vga_lcd</b>	82843	7356	7.04	7.04	0.80
<b>netcard</b>	478502	9112	40.37	40.37	0.72
<b>leon3mp</b>	509793	14742	43.86	43.86	0.73

The advantage of our thermal-aware 3D placement is compared to other thermal optimization methods. The results are presented in Table 21, and the results of **wb\_conmax** are visualized in Figure 30. The x-axis shows the normalized half-perimeter wirelength (HPWL), and the y-axis shows the temperature. The “baseline” is a wirelength-driven placement generated by solving **(P3)** with  $\beta = 0$ .

Three thermal optimization methods, *uniform power*, *post-processing*, and *co-placement* are compared.

The uniform-power method mimics the thermal-aware 3D placement methods [43][82] that do not consider the thermal effects of TSVs. Although a uniform-power distribution is not a thermal optimal solution, the difference is only a few degrees according to the Hotspot [49] simulation for 3D ICs. Thus, uniform power is a fair replacement for the previous thermal-aware 3D placement methods. It is able to be implemented by solving problem **(P3)** with TSV power  $\tilde{p}_{TSV} = 0$ . In this way, the TSV distribution cost becomes purely a power distribution cost. When the per-die total power

is assumed to be a constant, the minimizer of the cost function in equation (119) is a uniform per-die power distribution. The cost weight  $\beta$  is set to 1 in the implementation. The post-processing method is a direct application of Theorem 7.2 at the post-placement stage. After 3D global placement, an optimal TSV distribution is computed according to the power distribution, regardless of overlaps. The assignment of TSVs to the TSV slots in the target distribution is computed by a linear assignment method to minimize the wirelength overhead. The resulting overlaps are removed by a legalization step.

Co-placement reflects our method, which optimizes the TSV distribution during 3D placement. In Figure 30, the left endpoint of the curve is the result with TSV distribution cost weight  $\beta = 0.00$ , and the right endpoint is the result with  $\beta = 1.00$ .

From Figure 30, it is clear that co-placement outperforms the other two optimization methods, and reduces more temperature within a similar amount of wirelength overhead. As discussed in Section 7.2, if the thermal benefits of TSVs are not being considered, a uniform-power distribution is not effective for temperature reduction. The “average” rows in Table 21 show the average results normalized by the baseline results. Our co-placement method is able to reduce temperature by 34%, which is 4X greater than the uniform-power method that reduces temperature by only 8%. Although the post-processing method makes use of the heat conductivity of TSVs, it is likely to cause congestion due to displacement. Thus, the legalized results have either higher temperature, or longer wirelength.

Moreover, our co-placement method provides a mechanism for wirelength and temperature trade-offs, as shown in Figure 30. The data points are generated with

different  $\beta$  values labeled in the figure. When the performance is critical, the acceptable wirelength degradation is limited. Our method is still able to reduce temperature with a negligible amount of wirelength degradation (e.g., 2%).

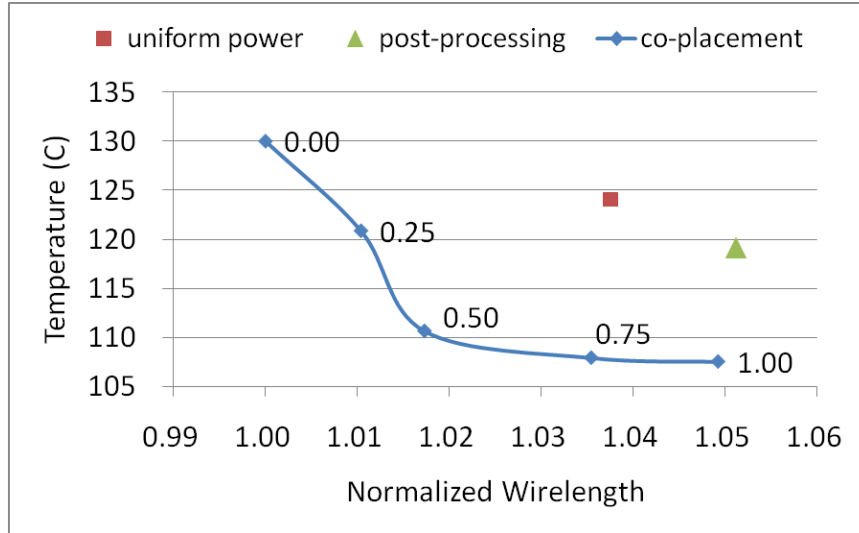


Figure 30. Comparison of thermal optimizations on wb\_conmax

Table 21. Results of our co-placement method and comparisons with other methods

Circuit		baseline	uniform power	post-processing	co-placement
aes_core	HPWL (m)	1.43	1.58	1.54	1.55
	T ( °C )	108	103	105	101
	RT (s)	206	180	208	208
wb_conmax	HPWL (m)	2.34	2.42	2.46	2.45
	T ( °C )	130	124	119	108
	RT (s)	214	289	220	257
ethernet	HPWL (m)	3.77	3.95	4.08	3.89
	T ( °C )	124	113	85	87
	RT (s)	490	395	506	502
des_perf	HPWL (m)	4.24	4.61	4.83	4.55
	T ( °C )	173	158	112	103
	RT (s)	689	639	702	759
vga_lcd	HPWL (m)	5.94	6.26	6.62	6.13
	T ( °C )	108	112	80	79
	RT (s)	772	815	854	812
netcard	HPWL (m)	37.17	39.31	40.67	40.21
	T ( °C )	461	415	288	194
	RT (s)	5121	4620	5439	5693
leon3mp	HPWL (m)	40.10	43.42	45.38	43.05

	T (°C)	437	347	201	160
	RT (s)	5440	4846	6480	5152
<b>Average</b>	HPWL	1.00	1.07	1.10	1.06
	T	1.00	0.92	0.72	0.66
	RT	1.00	0.97	1.06	1.06

## 7.6 Conclusions

In this chapter we identified a simple criterion for thermally optimal TSV distribution, where the TSV should follow the lumped power distribution. Based on this condition, we implement a thermal-aware 3D placement method. Experimental results show that it outperforms the uniform-power method that mimics existing thermal-aware placement methods.

## Chapter 8

### Case Study: 3D Physical Design of LEON3 Microprocessor

In this chapter we evaluate 3D-Craft, our 3D physical design flow, using the open-source microprocessor LEON3 [94] as a design driver. LEON3 is a 32-bit processor core compliant with the SPARC V8 architecture. The architectural diagram is shown in Figure 31. We reconfigure the VHDL source code, set the 4KB data cache and the 4KB instruction cache as direct-mapped caches, disable the local IRAM and local DRAM, and disable the FPU since it is not open-sourced.

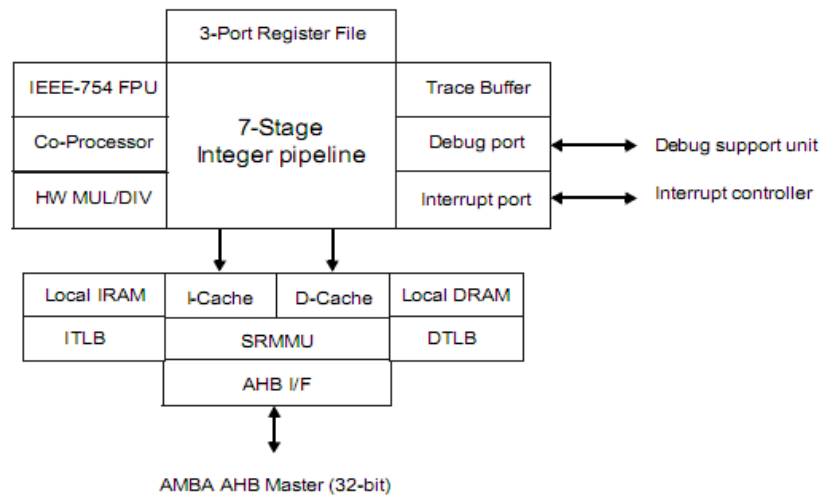


Figure 31. LEON3 processor core block diagram [94]

### 8.1 Standard-Cell Implementation

First, we evaluate the benefit of wirelength reduction using 3D integration technology and the capability of our 3D-Craft tool with a standard-cell version of LEON3. We synthesize a single-core LEON3 processor based on the NCSU standard cell kit [36] for the MITLL 0.18 $\mu\text{m}$  FD-SOI technology. The size of a TSV is  $3 \times 3 \mu\text{m}^2$  with an extra

1.5  $\mu\text{m}$  spacing on each side; thus a TSV consumes an area of  $36 \mu\text{m}^2$ . We perform physical design and evaluate the wirelengths for a 2D implementation and a 3-die 3D implementation.

The synthesized netlist consists of 95061 standard cells, 97880 nets, and 150 I/O ports. The total cell area is  $11.05 \text{ mm}^2$ . We define the 2D and 3D placement regions such that there is 20% of white space in total, excluding the area consumed by TSVs. In detail, we define a  $3.72 \times 3.72 \text{ mm}^2$  placement region for the 2D implementation, and a  $2.15 \times 2.15 \text{ mm}^2$  placement region for the 3-die 3D implementation. Since 2D physical design can be treated as a special case of 3D physical design, we use 3D-Craft to generate placement and global routing for both implementations. The placement is generated by mPL-3D, and the global routing is generated by Cadence Encounter.

We list the statistics of the physical design results in Table 22. The half-perimeter wirelength (HPWL) and the wirelength estimated after global routing (routed WL) are computed by Cadence Encounter. The TSVs are inserted by mPL-3D assuming there is only one TSV for every inter-die net, and for each die we only count the TSVs (#TSV) that consume silicon area. The area utilization is also reported in the last column. In this example, the HPWL of the 3-die 3D implementation is 29% shorter than the 2D implementation, and the routed wirelength is 37% shorter; this demonstrates the benefits of the 3D implementation.

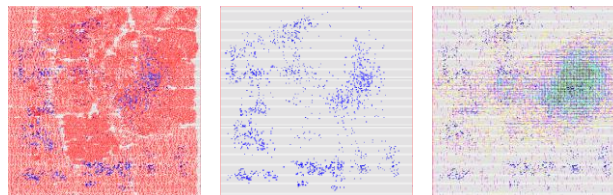
Table 22. Statistics of the 2D and 3D implementations

		HPWL (mm)	routed WL (mm)	#TSV	utilization
2D		11.23	17.11	N/A	0.80
3D	Bottom die	2.26	3.33	0	0.80
	Middle die	3.57	4.5	729	0.81
	Top die	2.2	2.96	1191	0.81
	<b>Total</b>	<b>8.03</b>	<b>10.79</b>	<b>1920</b>	<b>0.81</b>

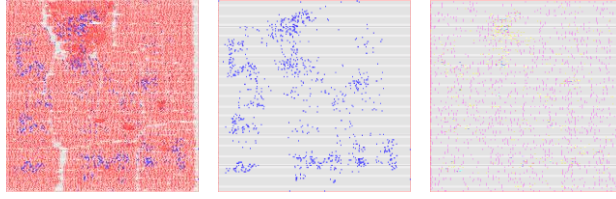
The placement, TSV distribution and routing congestion analysis are shown in Figure 32. The placements and the TSV distribution are shown in the first two pictures of each die, respectively. The routing congestions on the right are analyzed by Cadence Encounter, where the shaded spots represent congested regions. Although there are congested regions in the middle die, we observe similar congestions in the 2D implementation; thus the comparison is fair. Since the MITLL 3D IC technology provides only three metal layers for each die, the routing congestions can be probably solved by adopting 3D IC technologies that provide more routing resources, or by integrating a routability-driven 3D placer in 3D-Craft.



(a) Bottom die



(b) Middle die



(c) Top die

Figure 32. The 3D placement by mPL-3D, the TSV distribution, and the routing congestion analysis by Cadence Encounter

## 8.2 Mixed-Size Implementation and Physical Hierarchy Exploration

Most of the existing 3D IC designs constrain each functional block in the logical hierarchy to be in a single die, which may not generate the best 3D physical hierarchy (see the discussion in [24] about physical hierarchy vs. logical hierarchy). Therefore, it is worthwhile to apply 3D-oriented physical design methods instead of only performing floorplanning with existing 2D units. The study in [61] explores the 3D design space using an architectural planner and a timing and power model for 3D implementation of cache-like structures, and shows that the performance improvement and power reduction is significant when adapting 3D functional units. Thus, it is useful to consider flattening some or all levels of logical hierarchy to obtain a more optimal 3D implementation with fewer constraints on the physical hierarchy. In general, we were motivated to study the benefits of removing the logical hierarchical restrictions on 3D physical design.

We synthesize a single-core LEON3 processor with the UMC 90nm digital cell library and the memory macros generated by Faraday’s memory compiler. In Table 23, the cell number, macro number, net number and total area are extracted from the synthesized netlist, and we estimate the 2D HPWL using the Cadence SoC Encounter 6.2.

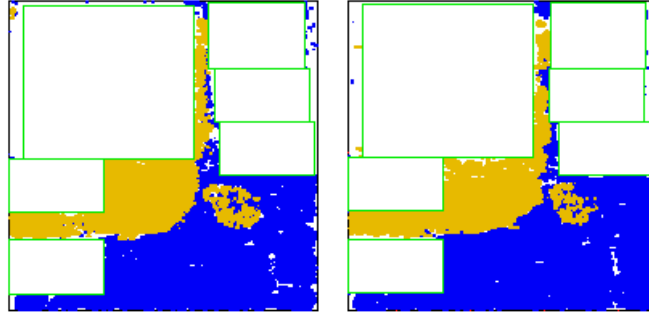


To obtain the 2D HPWL, we create a square placement region with 10% whitespace, and then run the placer in Encounter without congestion effort and timing optimization for a wirelength-driven placement. As a comparison, we run our mixed-size 3D placement to get the 3D HPWL, which is performed on a 2-die placement region with 10% total white space. The 2-die 3D implementation provides a potential of more than 40% wirelength reduction compared to the 2D implementation, with about 4000 TSVs. As an estimate of the TSV cost, we may assume each TSV consumes a  $3 \times 3 \mu\text{m}^2$  pitch, and the capacitance of one TSV is approximately equal to an  $8\mu\text{m}$  metal-2 wire [36]. With such a small amount of TSV cost, the capacity overhead is equivalent to a wirelength cost of  $0.032 \times 10^6 \mu\text{m}$  (less than 2% of the total wirelength), so the 3D implementation results in a great reduction in power.

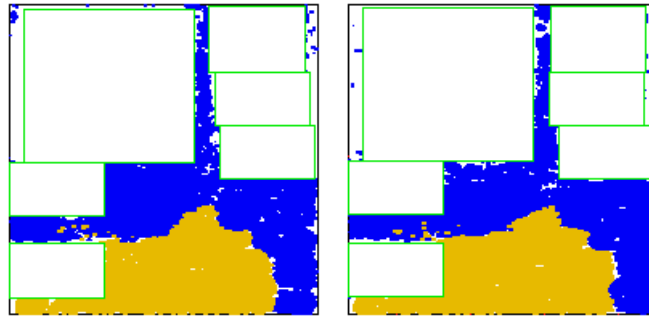
Table 23. Overall statistics of the synthesized netlist

#cell	#mac	#net	total area ( $\mu\text{m}^2$ )	2D HPWL ( $\mu\text{m}$ )	3D HPWL ( $\mu\text{m}$ )	3D TSV
34225	12	36789	$6.67 \times 10^5$	$1.70 \times 10^6$	$0.99 \times 10^6$	3835

This 3D placement is performed with the flattened netlist, which obtains a great reduction in wirelength by removing all the logical hierarchical restrictions. The placement of the processor core and the register file is shown in Figure 33(a) and Figure 33(b), respectively. In each set of figures, the left one is the placement on the bottom die, and the right one is the placement on the top die. The processor core and the register file are placed on both dies. It is obvious that these units are not confined to one die, and are not even in a cuboid shape.



(a) Processor core `/leon3/p/` (in lighter color)



(b) Register file `/leon3/rf/` (in lighter color)

Figure 33. Placement of logical units

It remains a question whether such wirelength reduction is achievable by maintaining part of the logical hierarchy. To conduct this study, we shall first summarize the logical hierarchy of the synthesized netlist. Table 24 and Figure 34 show the per-unit area consumption of the logical units. The logical hierarchy is not exactly the same as the architectural diagram in Figure 31, but there exist corresponding logical units for the blocks in the diagram. There are 12 hard macros consuming more than 60% area; these are the memory blocks for the cache memory, the TLB memory and the trace buffer. The major logical units include the processor core (`/leon3/p/`, 11.1%), the register file (`/leon3/rf/`, 16.6%), the cache memory (`/leon3/cmем/`, 38.1%), the TLB memory

(/leon3/tbmem/, 12.0%) and the debug support unit (/dsu/, 13.4%), which are visualized in Figure 34.

Table 24. Statistics of the per-unit area consumption

unit name	area%	description
/leon3/p/	11.1%	processor core: pipeline (6.6%), MMU (2.2%), MUL (1.6%) and DIV (0.7%)
/leon3/rf/	16.6%	register file
/leon3/cmем/	38.1%	cache memory blocks: 4KB data cache (15.4%), 512B data cache tag (3.7%); 4KB instruction cache (15.4%), 512B instruction cache tag (3.7%)
/leon3/tbmem/	12.0%	TLB memory blocks (4 x 256B)
/dsu/	13.4%	debug support unit with trace buffer (4 x 256B)
/mctrl/	1.8%	memory controller
/irqctrl/	0.3%	interrupt controller
/uart/	0.7%	UART serial interface
/ahb/	2.4%	AMBA AHB bus
/apb/	1.7%	AMBA APB bus
/gptimer/	1.4%	general purpose timer unit
/grgpio/	0.3%	general purpose I/O port

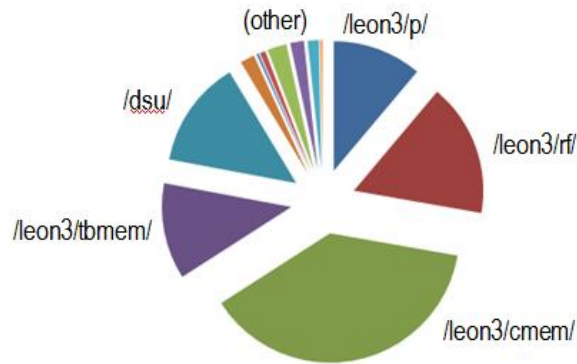
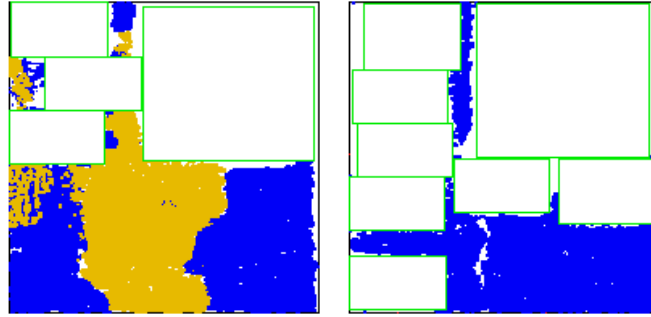


Figure 34. Plot of the per-unit area consumption

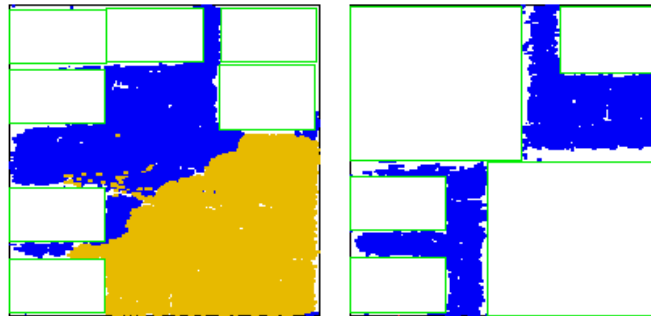
To restrict one logical module placed on only one die, we first create a square macro for that module to replace the cells in the flat netlist, run the mixed-size placer, and then run another round of mixed-size placement with the flat netlist by fixing the die

assignment of all the movable cells and macros. Figure 35 shows two results: Figure 35(a) is the 3D placement which is a result of placing the entire processor core on a single die, and Figure 35(b) is the 3D placement which is a result of placing the entire register file on a single die. The HPWL and TSV number are compared in Table 25, which shows that the restricted placement for the processor core brings in 10% longer HPWL, and the restricted placement for the register file brings in 20% longer HPWL.

From these results on the LEON3 example, we see that the 3D placement with flattened netlist benefits the most from 3D integration technology. The number of cells and macros in a modern design can be in the scale of billions, but computational cost can be compensated for by using parallel computing.



(a) Restricted placement for the processor core (in lighter shading)



(b) Restricted placement for the register file (in lighter shading)

Figure 35. Restricted placement for some logical modules

Table 25. HPWL and TSV comparisons for different placements

Figure 33		Figure 35(a)		Figure 35(b)	
HPWL	TSV	HPWL	TSV	HPWL	TSV
$0.99 \times 10^6$	3835	$1.09 \times 10^6$	1715	$1.20 \times 10^6$	845

### 8.3 Mixed-Size Implementation and Timing Characteristics

Previously, the work in [78] applied our 3D-Craft to implement the logic-on-logic 3D integration of standard cell circuits. The target 3D chip uses Tezzaron's 130nm 3D technology process [71], where the two logic dies are bonded face-to-face with microbumps as inter-die connections. Different implementations of FFT processing elements (PE) is listed in Table 26, including a 2D implementation (2D), a 3D

implementation using sequential execution of a commercial placer (3D-Seq.), a 3D implementation using the partition-first approach (3D-Part.), and the 3D implementation by 3D-Craft. The results show that 3D-Craft achieves a 23% higher performance than the 2D implementation, and achieves a 15% higher performance than the ad hoc 3D implementation using commercial 2D placers sequentially.

Table 26. Four implementations of the PE unit

<b>PE Impl.</b>	<b>Total wirelength (mm)</b>	<b>normalized</b>	<b>Max frequency (MHz)</b>	<b>normalized</b>
<b>2D</b>	588	1.00	31.61	1.00
<b>3D-Seq.</b>	487.3	0.83	33.84	1.07
<b>3D-Part.</b>	484.1	0.82	36.72	1.16
<b>3D-Craft</b>	464.8	0.79	38.74	1.23

In this section we are interested in studying the performance benefits from 3D integration in the presence of cache blocks.

The design driver is a 4-core LEON3 microprocessor, synthesized with the UMC 90nm standard cell library and Faraday’s memory compiler. The target 3D chip is assumed to be a 2-die chip bonded face-to-face. Each core of the 4-core LEON3 processor is similar to the netlist described in Section 8.2, and the cores are connected with an *Advanced High-performance Bus (AHB)*. We use the timer in OAGear [102] to measure the delay of the longest path, and convert the delay into maximum frequency, where the net load is approximated by a simple linear wire model as in [70] with the electrical parameters as on the 3<sup>rd</sup> metal layer.

Several 3D implementations of the 4-core LEON3 are listed in Table 27; which are generated with different weights of the TSV number in the objective function. Different

from the standard cell circuits, the wirelength is not always reduced in the extra cost of TSVs. Instead, the minimum wirelength presents only with a suitable amount of TSVs. The 3D implementation with the best performance ( $\alpha_{\text{TSV}} = 8\mu\text{m}$ ) is 40% higher than the 2D implementation, while there is only a 23% advantage in wirelength. This supports the prediction [84] that the longest wires will benefit more than the average wire, even for this mixed-size netlist that is out of the scope of their regular circuit model.

Table 27. 3D Implementations of a 4-core LEON3

	HPWL (m)	#TSV	Delay (ns)	Frequency (MHz)
<b>2D Impl.</b>	5.51	0	7.15	140
<b><math>\alpha_{\text{TSV}} = 2\mu\text{m}</math></b>	5.02	38899	6.69	149
<b><math>\alpha_{\text{TSV}} = 4\mu\text{m}</math></b>	4.89	23155	7.26	138
<b><math>\alpha_{\text{TSV}} = 8\mu\text{m}</math></b>	4.29	15965	5.15	194
<b><math>\alpha_{\text{TSV}} = 16\mu\text{m}</math></b>	4.41	6725	5.73	175
<b><math>\alpha_{\text{TSV}} = 32\mu\text{m}</math></b>	4.50	4029	5.59	179
<b><math>\alpha_{\text{TSV}} = 64\mu\text{m}</math></b>	4.65	3221	5.38	186

## Chapter 9

### Conclusions and Future Work

Along with the development of 3D integration technologies in the past decade, a significant amount of advancement has been made in the 3D IC physical design automation. In this dissertation we cover important problems and algorithms developed in the 3D physical design flow, especially the 3D placement problem. A high-level overview of the basic concepts in the 3D physical design flow is presented, and the necessary references are included for readers who would like to dig deeper into a specific topic.

The industrial tools and services for 3D integration are emerging. A recent article [37] summarizes the industrial efforts on *Electronic Design Automation (EDA)* tools for 3D integration. With TSV-related technology awareness, Synopsys developed a method [77] to address the TSV-induced stress. To analyze the thermal issues of 3D ICs, Gradient developed the thermal simulator HeatWave 3DIC [95]. 3D testing has different requirements than 2D testing, and the Mentor Graphics Tessent Platform [76] supports the before-packaging and after-packaging testing of 3D ICs. In order to implement a 3D IC, the design tools include Micro Magic's MAX-3D Layout Editor [99] for custom design, and R3Logic's products [100] for both custom design and die-level integration. Cadence's Encounter Digital Implementation (EDI) System [93] supports integration and analysis (timing, thermal, signal integrity) of a 3D system with TSVs. Last, but not least, Atrenta is actively developing its Spyglass [92] physical 3-D prototyping tools to



collaborate with AutoESL, CEA-Leti, IMEC and Qualcomm for early-stage design exploration. The features and corresponding tools are listed in Table 28.

Table 28. Reported 3D-aware EDA tools

Features	EDA tools
Design exploration	Atrenta Spyglass
Manual Layout and Integration	Micro Magic MAX-3D Layout Editor R3Logic's projects Cadence EDI System
Timing and signal integrity analysis	Cadence EDI System
Thermal analysis	Gradient HeatWave 3DIC Cadence EDI System
Stress analysis and optimization	Synopsys patent
Prebond and postbond testing	Mentor Graphics Tessent Platform

However, the commercial progress of 3D EDA flows has a chicken-and-egg problem: A complete commercial 3D EDA flow will not happen before there is a large market, but the market does not grow large before there is a complete EDA flow. To facilitate wide adoption of 3D integration technologies, the following issues must be addressed: lack of EDA tools, complexity of 3D designs, manufacturing cost, and lack of standards. There have been several organizations that formed groups to investigate and develop standards, including JEDEC, SEMI, SEMATECH, SIA, SRC, IEEE-SA, IEEE-ISTO, GSA, and Si2 [8]. In the perspective of 3D physical design, research needs to address the following issues:

- Format standards for data exchange and interoperability of 3D physical design tools: The commercial 3D EDA support is only at an emerging stage. Most tools are developed by academia and start-ups. Format standards will facilitate the integration of a complete 3D physical design flow with thermal awareness and stress awareness, including the processes of partitioning/floorplanning, placement,

power grid synthesis, clock tree synthesis, buffer insertion, circuit tuning, and routing,

- 3D physical hierarchy optimization: The study in [24] highlighted the difference in physical hierarchy and logic hierarchy, and underscored the need for physical design generation. This is even more important for 3D designs. Early 3D architecture exploration work using block stacking only led to disappointing performance gain due to the simple adoption of logic and physical hierarchy optimized for 2D designs [33]. More performance gain is shown to be possible on the same architecture by applying 3D designs further down to the logic hierarchy [61].
- Strong linkage between the architecture level analysis tool and 3D physical planning tools: This link is required to take advantage of 3D IC technologies with new architectures and physical implementations. Physical design and microarchitecture co-design is needed.
- Heterogeneous system modeling and optimization: A heterogeneous 3D IC can be viewed as a miniaturized PCB boards, and the boards components are integrated in a single 3D IC. For 2D systems, the chip-package-board co-design has already been strongly recommended by industry [40]. The stacking structure of 3D ICs creates different electrical and mechanical situations, and requires additional efforts on system-level modeling and optimization.

## Appendix: Manual of 3D-Craft

The 3D-Craft is a set of programs that complete the place and route of a 3D design. The programs and an example flow can be obtained at [http://cadlab.cs.ucla.edu/three\\_d/](http://cadlab.cs.ucla.edu/three_d/). The flow works as in the following steps:

### 1) Setting Up the Standard Cell Library and the TSV Library

In this manual we use the standard cell library (*gscl45nm.lef*) [90] developed by Oklahoma State University based on North Carolina State University's FreePDK45nm [88]. The TSV is defined in the LEF file *gscl45nm\_TSV.lef*, with TSV pitch of 2.47 $\mu$ m and TSV size of 2.00 $\mu$ m. The commands to create an OpenAccess cell library and TSV library are as the following:

```
Linux> lef2oa -lib gscl45nm -lef ./lib/gscl45nm.lef
```

```
Linux > lef2oa -lib gscl45nm_TSV -lef ./lib/gscl45nm_TSV.lef -techLib gscl45nm
```

The program *lef2oa* is provided by OpenAccess. It converts the LEF files to OpenAccess libraries. After running these commands, a cell library *gscl45nm/* and a TSV library *gscl45nm\_TSV/* are created.

### 2) Setting Up the 3D Design Library

We use the design *aes\_core* in the IWLS 2005 benchmarks [97] and synthesize it with the *gscl45nm* standard cell library. The steps to set up a 3D design library include: (i) converting the design to OpenAccess format, (ii) setting up the chip size, and (iii) setting up the bonding order. The commands for these steps are as the following:

```
Linux> setenv lib work
```

```
Linux> setenv cell aes_core
```

```
Linux> setenv TSV TSV_STD
```

```
Linux> verilog2oa -lib $lib -verilog ./verilog/$cell.v -refLibs gscl45nm
```

```
Linux> genPhysical -techlib gscl45nm -lib $lib -cell $cell -outView T4 -util 2.80
```

```
Linux> crBonding --cell "$lib $cell T4" --set "FFFF"
```

The program *verilog2oa* is provided by OpenAccess, which converts a verilog file to the OpenAccess format. The program *genPhysical* is modified from OA Gear [102] with 3D awareness. It sets up the chip size by specifying the utilization rate (*-util*), which equals the total cell area divided by the chip area per die. For example, to reserve 30% of white spaces for a 3D chip with four dies, the utilization rate should be set to  $(1-30\%) \times 4 = 2.80$ . The command *crBonding* is a program of 3D-Craft, which sets up the bonding order. In this example, face-to-back stacking is used for all the four dies, so a string “FFFF” is set. Please refer to Section 3.2.3 for a detailed description of this extension of OpenAccess.

This step creates an OpenAccess design “work aes\_core T4,” where work is the library name, aes\_core is the cell name, and T4 is the view name.

### **3) Running the 3D Floorplanner or the Real 3D Placer**

In this step we are going to determine the die assignment using either the 3D floorplanner or the real-3D placer. Please refer to Section 3.3.2 for a description of the 3D floorplanner, and refer to Chapter 6 for a detailed discussion of the analytical 3D placer.

The command to run the 3D floorplanner is as the following:

```
Linux> mPL-R3D -fplan 1 -lib $lib -cell $cell -view T4 -TSV $TSV \  
-fp.areaWt 0.6 -fp.wireWt 0.7 -fp.tsvWt 0.2 -writePl T4-fplan
```

The command to run the real-3D placer is as the following:

```
Linux> mPL-R3D -lib $lib -cell $cell -view T4 -TSV $TSV \  
-viaWeight 8000 -clusteringDepth 1 -writePl T4-fplan
```

The command *mPL-R3D* is a program of 3D-Craft. It integrates both the floorplanner and the real-3D placer. The option “-fplan 1” enables the floorplanning mode. The weights for the area (fp.areaWt), the wirelength (fp.wireWt) and the TSV number (fp.tsvWt) can be tuned in the floorplanning mode. In the placement mode, the TSV weight (viaWeight), and the clustering depth (clusteringDepth) can be tuned. In this example, we run the placer with a TSV weight of  $8\mu m$  in a 2-level scheme (clustering once).

This step creates an OpenAccess design “work aes\_core T4-fplan.”

#### **4) TSV Insertion, Net Splitting and Chip Resizing**

The commands to complete the TSV insertion and net splitting are as the following:

```
Linux> crCopyDesign "$lib $cell T4-fplan" "$lib $cell T4-split"
```

```
Linux> crSplitNets "$lib $cell T4-split" --TSV $TSV
```

```
Linux> genPhysical -techlib gscl45nm -lib $lib -cell $cell -view T4-split \  
-phys -outView T4-resize -util 2.80
```

The commands *crCopyDesign* and *crSplitNets* are programs of 3D-Craft. The program *crSplitNets* finishes the tasks of both TSV insertion and net splitting. It directly modifies the given OpenAccess design. In order to keep the intermediate result from the previous step, a copy of the design after real 3D placement is created by *oaCopyDesign*. Please refer to Section 3.3.4 for a detailed description of TSV insertion and net splitting. After TSV insertion, the chip size may be adjusted during the design exploration stage using the *genPhysical* program.

This step creates an OpenAccess design “work aes\_core T4-resize.”

## 5) TSV Planning and Legalization

The step is implemented by running a pseudo 3D placer as the following:

```
Linux> mPL-P3D "$lib $cell T4-resize" --out T4-mPL --TSV $TSV
```

```
Linux> mPL-xdp --noglobal "$lib $cell T4-mPL" --out T4-xdp --TSV $TSV
```

The command *mPL-P3D* (pseudo 3D) and *mPL-xdp* are programs of 3D-Craft. The pseudo 3D placer only places the cells and TSVs in the horizontal directions with a fixed die assignment. The detailed placer legalizes both the TSVs and cells.

This step creates an OpenAccess design “work aes\_core T4-xdp.”

## 6) 3D Design Export for Commercial EDA tools

The following commands export a 3D design into several 2D sub-designs:

```
Linux> crSplitDesign "$lib $cell T4-xdp " "$lib $cell mPL" --TSV $TSV
```

```
Linux> oa2verilog -lib $lib -cell ${cell}_die0 -view mPL -verilog ${cell}_die0.v
```

```
Linux> oa2verilog -lib $lib -cell ${cell}_die1 -view mPL -verilog ${cell}_die1.v
```

```
Linux> oa2verilog -lib $lib -cell ${cell}_die2 -view mPL -verilog ${cell}_die2.v
```

```
Linux> oa2verilog -lib $lib -cell ${cell}_die3 -view mPL -verilog ${cell}_die3.v
```

```
Linux> oa2def -lib $lib -cell ${cell}_die0 -view mPL -def ${cell}_die0.placed.def
```

```
Linux> oa2def -lib $lib -cell ${cell}_die1 -view mPL -def ${cell}_die1.placed.def
```

```
Linux> oa2def -lib $lib -cell ${cell}_die2 -view mPL -def ${cell}_die2.placed.def
```

```
Linux> oa2def -lib $lib -cell ${cell}_die3 -view mPL -def ${cell}_die3.placed.def
```

The command *crSplitDesign* is a program of 3D-Craft, which is able to split a 3D design into several 2D sub-designs, as long as the TSVs are inserted and fixed. In this example, four 2D sub-designs are created by this program, one for each die. Using the programs *oa2verilog* and *oa2def* provided by OpenAccess, these sub-designs can be converted to several verilog files and DEF files that are ready for commercial tools.

## References

- [1] C. Ababei, H. Mogal, and K. Bazargan, "Three-dimensional place and route for FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 25, Jun. 2006, pp. 1132-1140.
- [2] S.N. Adya and I.L. Markov, "Consistent placement of macro-blocks using floorplanning and standard-cell placement," *Proceedings of the 2002 International Symposium on Physical Design (ISPD)*, 2002, p. 12.
- [3] S.N. Adya, S. Chaturvedi, J.A. Roy, D.A. Papa, and I.L. Markov, "Unification of partitioning, placement and floorplanning," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2004, pp. 550-557.
- [4] K. Arrow, L. Hurwicz, H. Uzawa, "Studies in linear and non-linear programming," *Stanford University Press*, 1958.
- [5] K. Balakrishnan, V. Nanda, S. Easwar, and S.K. Lim, "Wire congestion and thermal aware 3D global placement," *Proceedings of the 2005 Conference on Asia South Pacific Design Automation (ASP-DAC)*, 2005, p. 1131.
- [6] K. Banerjee, A. Mehrotra, and A. Sangiovanni-Vincentelli, "On thermal effects in deep sub-micron VLSI interconnects," *Proceedings of the 1999 Conference on Design Automation (DAC)*, 1999, pp. 885-891.
- [7] K. Banerjee, S.J. Souri, P. Kapur, and K.C. Saraswat, "3-D ICs: a novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration," *Proceedings of the IEEE*, vol. 89, May. 2001, pp. 602-633.
- [8] K. Bartleson, "Standards for 3D?" *EE Times*, Mar. 28, 2011.
- [9] K. Bernstein, R. Puri, A. Young, P. Andry, J. Cann, P. Emma, D. Greenberg, W. Haensch, M. Ignatowski, S. Koester, and J. Magerlein, "Interconnects in the third dimension," *Proceedings of the 44th Annual Conference on Design Automation (DAC)*, 2007, p. 562.
- [10] B. Black, D.W. Nelson, C. Webb, and N. Samra, "3D processing technology and its impact on iA32 microprocessors," *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, 2004, pp. 316-318.
- [11] Paul Boldt, Don Scansen, "A5: All Apple, part mystery", *EE Times*, Apr. 13, 2011.



- [12] S. Borkar, "3D Integration technology for energy efficient system design," *Proceedings of the 2010 International Symposium on VLSI Technology, System and Application (VLSI-TSA)*, 2010, pp. 100-103.
- [13] S. Boyd and L. Vandenberghe, "Convex Optimization," *Cambridge University Press*, 2004.
- [14] T.F. Chan, J. Cong, T. Kong, and J.R. Shinnerl, "Multilevel optimization for large-scale circuit placement," *Proceedings of the 2000 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2000, pp. 171-176.
- [15] T.F. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," *Proceedings of the 2005 International Symposium on Physical Design (ISPD)*, 2005, p. 185.
- [16] T.F. Chan, J. Cong, J.R. Shinnerl, K. Sze, and M. Xie, "mPL6: enhanced multilevel mixed-size placement," *Proceedings of the 2006 International Symposium on Physical Design (ISPD)*, 2006, p. 212.
- [17] T.F. Chan, J. Cong, and E. Radke, "A rigorous framework for convergent net weighting schemes in timing-driven placement," *Proceedings of the 2009 International Conference on Computer-Aided Design (ICCAD)*, 2009, p. 288.
- [18] R. Chanchani, "3D Integration Technologies – An Overview," *Materials for Advanced Packaging*, D. Lu and C.P. Wong, eds., Springer US, 2009, pp. 1-50.
- [19] C.-C. Chang and J. Cong, "An efficient approach to multi-layer layer assignment with application to via minimization," *Proceedings of the 34th Annual Conference on Design Automation (DAC)*, 1997, pp. 600-603.
- [20] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "A high-quality mixed-size analytical placer considering preplaced blocks and density constraints," *Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2006, p. 187.
- [21] Y. Chen, D. Niu, Y. Xie, and K. Chakrabarty, "Cost-effective integration of three-dimensional (3D) ICs emphasizing testing cost analysis," *Proceedings of the 2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2010, pp. 471-476.
- [22] C. Chiang and S. Sinha, "The road to 3D EDA tool readiness," *Proceedings of the 2009 Conference on Asia South Pacific Design Automation (ASP-DAC)*, 2009, pp. 429-436.

- [23] Y.-L. Chuang, P.-W. Lee, and Y.-W. Chang, "Voltage-drop aware analytical placement by global power spreading for mixed-size circuit designs," *Proceedings of the 2009 International Conference on Computer-Aided Design (ICCAD)*, 2009, p. 666.
- [24] J. Cong, "Timing closure based on physical hierarchy," *Proceedings of the 2002 International Symposium on Physical Design (ISPD)*, 2002, p. 170.
- [25] J. Cong and Y. Zhang, "Thermal-driven multilevel routing for 3-D ICs," *Proceedings of the 2005 Conference on Asia South Pacific Design Automation (ASP-DAC)*, 2005, p. 121.
- [26] J. Cong and Y. Zhang, "Thermal via planning for 3-D ICs," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2005, pp. 745-752.
- [27] J. Cong and M. Xie, "A Robust Mixed-Size Legalization and Detailed Placement Algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, Aug. 2008, pp. 1349-1362.
- [28] J. Cong and G. Luo, "A 3D physical design flow based on Open Access," *Proceedings of the 2009 International Conference on Communications, Circuits and Systems (ICCCAS)*, 2009, pp. 1103-1107.
- [29] J. Cong and G. Luo, "Advances and Challenges in 3D Physical Design," *IPSJ Transactions on System LSI Design Methodology*, vol. 3, 2010, pp. 2-18.
- [30] J. Cong and Y. Ma, "Thermal-Aware 3D Floorplan," *Three Dimensional System Integration: IC Stacking Process and Design*, Y. Xie, J. Cong, and S. Sapatnekar, eds., Springer US, 2010, pp. 63-102.
- [31] J. Cong and G. Luo, "Thermal-Aware 3D Placement," *Three Dimensional System Integration: IC Stacking Process and Design*, Y. Xie, J. Cong, and S. Sapatnekar, eds., Boston, MA: Springer US, 2010, pp. 103-144.
- [32] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2004, pp. 306-313.
- [33] J. Cong, A. Jagannathan, Y. Ma, G. Reinman, J. Wei, and Y. Zhang, "An automated design flow for 3D microarchitecture evaluation," *Proceedings of the 2006 Conference on Asia South Pacific Design Automation (ASP-DAC)*, 2006, p. 384.

- [34] J. Cong, G. Luo, J. Wei, and Y. Zhang, "Thermal-aware 3D IC placement via transformation," *Proceedings of the 2007 Conference on Asia South Pacific Design Automation (ASP-DAC)*, Jan. 2007, pp. 780-785.
- [35] S. Das, "Design Automation and Analysis of Three-Dimensional Integrated Circuits," *PhD Dissertation*, Massachusetts Institute of Technology, 2004.
- [36] W.R. Davis, J. Wilson, S. Mick, C. Mineo, A.M. Sule, M. Steer, and P.D. Franzon, "Demystifying 3D ICs: The Pros and Cons of Going Vertical," *IEEE Design and Test of Computers*, vol. 22, Jun. 2005, pp. 498-510.
- [37] M. Demler, "EDA Tools Pave Path to 3-D ICs," *EDN*, Jun. 2011, pp. 26-34.
- [38] X. Dong and Y. Xie, "System-level cost analysis and design exploration for three-dimensional integrated circuits (3D ICs)," *Proceedings of the 2009 Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2009, pp. 234-241.
- [39] H. Eisenmann and F.M. Johannes, "Generic global placement and floorplanning," *Proceedings of the 35th Annual Conference on Design Automation (DAC)*, 1998, pp. 269-274.
- [40] J.-W. Fang, K.-H. Ho, and Y.-W. Chang, "Routing for chip-package-board co-design considering differential pairs," *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 512-517.
- [41] C. Ferri, S. Reda, and R.I. Bahar, "Parametric yield management for 3D ICs," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 4, Oct. 2008, pp. 1-22.
- [42] P.D. Franzon, S. Berkeley, B. Shani, K. Obermiller, W.R. Davis, M.B. Steer, S. Lipa, E.C. Oh, T. Thorolfsson, S. Melamed, S. Luniya, and T. Doxsee, "Design and CAD for 3D integrated circuits," *Proceedings of the 45th Annual Conference on Design Automation (DAC)*, 2008, p. 668.
- [43] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," *Proceedings of the 2003 International Conference on Computer-Aided Design (ICCAD)*, 2003, pp. 86-89.
- [44] B. Goplen and S. Sapatnekar, "Thermal via placement in 3D ICs," *Proceedings of the 2005 International Symposium on Physical Design (ISPD)*, 2005, p. 167.
- [45] B. Goplen and S. Sapatnekar, "Placement of 3D ICs with thermal and interlayer via considerations," *Proceedings of the 44th Annual Conference on Design Automation (DAC)*, 2007, p. 626.

- [46] R. Hentschke, G. Flach, F. Pinto, and R. Reis, "3D-vias aware quadratic placement for 3D VLSI circuits," *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2007, pp. 67-72.
- [47] D. Hill, "Method and System for High Speed Detailed Placement of Cells within an Integrated Circuit Design," *US Patent 6370673*, 2002.
- [48] M.-K. Hsu, Y.-W. Chang, and V. Balabanov, "TSV-Aware Analytical Placement for 3D IC Designs," *Proceedings of the 48th Annual Conference on Design Automation (DAC)*, 2011, pp. 664-669.
- [49] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M.R. Stan, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 14, May. 2006, pp. 501-513.
- [50] A.B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 24, May. 2005, pp. 734-747.
- [51] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning," *Proceedings of the 34th Annual Conference on Design Automation (DAC)*, 1997, pp. 526-529.
- [52] I. Kaya, S. Salewski, M. Olbrich, and E. Barke, "Wirelength Reduction Using 3-D Physical Design," *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, E. Macii, V. Paliouras, and O. Koufopavlou, eds., Springer Berlin / Heidelberg, 2004, pp. 453-462.
- [53] A.A. Kennings and I.L. Markov, "Analytical minimization of half-perimeter wirelength," *Proceedings of the 2000 Conference on Asia South Pacific Design Automation (ASP-DAC)*, 2000, pp. 179-184.
- [54] D.H. Kim, K. Athikulwongse, and S.K. Lim, "A study of Through-Silicon-Via impact on the 3D stacked IC layout," *Proceedings of the 2009 International Conference on Computer-Aided Design (ICCAD)*, 2009, p. 674.
- [55] J.-S. Kim, C.S. Oh, H. Lee, D. Lee, H.-R. Hwang, S. Hwang, B. Na, J. Moon, J.-G. Kim, H. Park, J.-W. Ryu, K. Park, S.-K. Kang, S.-Y. Kim, H. Kim, J.-M. Bang, H. Cho, M. Jang, C. Han, J.-B. Lee, K. Kyung, J.-S. Choi, and Y.-H. Jun, "A 1.2V 12.8GB/s 2Gb mobile Wide-I/O DRAM with 4×128 I/Os using TSV-based stacking," *Proceedings of the 2011 IEEE International Solid-State Circuits Conference (ISSCC)*, 2011, pp. 496-498.

- [56] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing," *Science (New York, N.Y.)*, vol. 220, May. 1983, pp. 671-680.
- [57] J.M. Kleinbans, G. Sigl, F.M. Johannes, and K.J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 10, Mar. 1991, pp. 356-365.
- [58] J.U. Knickerbocker, P.S. Andry, B. Dang, R.R. Horton, M.J. Interrante, C.S. Patel, R.J. Polastre, K. Sakuma, R. Sirdeshmukh, E.J. Sprogis, S.M. Sri-Jayantha, A.M. Stephens, A.W. Topol, C.K. Tsang, B.C. Webb, and S.L. Wright, "Three-dimensional silicon integration," *IBM Journal of Research and Development*, vol. 52, Nov. 2008, pp. 553-569.
- [59] C. Li and C.-K. Koh, "Recursive Function Smoothing of Half-Perimeter Wirelength for Analytical Placement," *Proceedings of the 8th International Symposium on Quality Electronic Design (ISQED)*, 2007, pp. 829-834.
- [60] J.C. Lin, W.C. Chiou, K.F. Yang, H.B. Chang, Y.C. Lin, E.B. Liao, J.P. Hung, Y.L. Lin, P.H. Tsai, Y.C. Shih, T.J. Wu, W.J. Wu, F.W. Tsai, Y.H. Huang, T.Y. Wang, C.L. Yu, C.H. Chang, M.F. Chen, S.Y. Hou, C.H. Tung, S.P. Jeng, and D.C.H. Yu, "High density 3D integration using CMOS foundry technologies for 28 nm node and beyond," *Technical Digest of the 2010 International Electron Devices Meeting (IEDM)*, 2010, pp. 2.1.1-2.1.4.
- [61] Y. Liu, Y. Ma, E. Kursun, G. Reinman, and J. Cong, "Fine grain 3D integration for microarchitecture design through cube packing exploration," *Proceedings of the 25th International Conference on Computer Design (ICCD)*, 2007, pp. 259-266.
- [62] D.G. Luenberger, "Optimization by Vector Space Methods," *John Wiley & Sons, Inc.*, 1969.
- [63] P. Mercier, S.R. Singh, K. Iniewski, B. Moore, and P. O'Shea, "Yield and Cost Modeling for 3D Chip Stack Technologies," *Proceedings of the 2006 IEEE Custom Integrated Circuits Conference (CICC)*, 2006, pp. 357-360.
- [64] G.-J. Nam, "ISPD 2006 Placement Contest," *Proceedings of the 2006 International Symposium on Physical Design (ISPD)*, 2006, p. 167.
- [65] G.-J. Nam and J. Cong, "Modern Circuit Placement," *Boston, MA: Springer US*, 2007.

- [66] G.-J. Nam, C.J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ISPD2005 placement contest and benchmark suite," *Proceedings of the 2005 International Symposium on Physical Design (ISPD)*, 2005, p. 216.
- [67] W.C. Naylor, R. Donnelly, and L. Sha, "Non-linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer," *US Patent 6301693*, 2001.
- [68] A.N. Ng, I.L. Markov, R. Aggarwal, and V. Ramachandran, "Solving hard instances of floorplacement," *Proceedings of the 2006 International Symposium on Physical Design (ISPD)*, 2006, p. 170.
- [69] J. Nocedal and S.J. Wright, "Numerical Optimization," *Springer New York*, 2006.
- [70] R.H.J.M. Otten and R.K. Brayton, "Planning for performance," *Proceedings of the 35th Annual Conference on Design Automation (DAC)*, 1998, pp. 122-127.
- [71] R.S. Patti, "Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs," *Proceedings of the IEEE*, vol. 94, Jun. 2006, pp. 1214-1224.
- [72] A.D. Polyani, "Handbook of Linear Partial Differential Equations for Engineers and Scientists," *Chapman and Hall/CRC*, 2002.
- [73] S.S. Sapatnekar, "Thermal Via Insertion and Thermally Aware Routing in 3D ICs," *Three Dimensional System Integration: IC Stacking Process and Design*, Y. Xie, J. Cong, and S. Sapatnekar, eds., Springer US, 2010, pp. 145-160.
- [74] N. Sillon, A. Astier, H. Boutry, L. Di Cioccio, D. Henry, and P. Leduc, "Enabling technologies for 3D integration: From packaging miniaturization to advanced stacked ICs," *Technical Digest of the 2008 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2008, pp. 1-4.
- [75] P. Spindler, U. Schlichtmann, and F.M. Johannes, "Kraftwerk2—A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, Aug. 2008, pp. 1398-1411.
- [76] S. Pateras, "3D-IC Testing with the Mentor Graphics Tessent Platform," *Mentor Graphics White Paper*, 2011.
- [77] Q. Su, M. Ni, Z. Tang, J. Kawa, J.D. Sproch, "ESD/Antenna Diodes for Through-Silicon Vias", *Patent Application US 2011/0095367 A1*.

- [78] T. Thorolfsson, G. Luo, J. Cong, and P.D. Franzon, "Logic-on-logic 3D integration and placement," *Proceedings of the 2010 IEEE International 3D Systems Integration Conference (3DIC)*, 2010, pp. 1-4.
- [79] C.-H. Tsai and S.-M. Kang, "Cell-level placement for improving substrate thermal distribution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 19, 2000, pp. 253-266.
- [80] P. Wilkerson, M. Furmanczyk, and M. Turowski, "Compact thermal modeling analysis for 3D integrated circuits," *Proceedings of the 11th International Conference Mixed Design of Integrated Circuits and Systems (MIXDES)*, 2004, pp. 277-282.
- [81] C. Xu, L. Jiang, S.K. Kolluri, B.J. Rubin, A. Deutsch, H. Smith, and K. Banerjee, "Fast 3-D thermal analysis of complex interconnect structures using electrical modeling and simulation methodologies," *Proceedings of the 2009 International Conference on Computer-Aided Design (ICCAD)*, 2009, p. 658.
- [82] H. Yan, Q. Zhou, and X. Hong, "Thermal aware placement in 3D ICs using quadratic uniformity modeling approach," *Integration, the VLSI Journal*, vol. 42, Feb. 2009, pp. 175-180.
- [83] Y. Zhan, S.V. Kumar, and S.S. Sapatnekar, "Thermally Aware Design," *Foundations and Trends® in Electronic Design Automation*, vol. 2, 2007, pp. 255-370.
- [84] R. Zhang, K. Roy, C.-K. Koh, and D.B. Janes, "Stochastic interconnect modeling, power trends, and performance characterization of 3-D circuits," *IEEE Transactions on Electron Devices*, vol. 48, Apr. 2001, pp. 638-652.
- [85] "3D Packaging" Magazine, Issue #13, *Yole Development*, Dec. 2009.
- [86] "POWER7: IBM's Next Generation Server Processor," *ASIA Power Architecture Conference Series*, 2009.
- [87] "Strategy of the 3D-integration program," *IMEC Scientific Report*, 2010.
- [88] <http://www.eda.ncsu.edu/wiki/FreePDK>
- [89] <http://er.cs.ucla.edu/benchmarks/ibm-place>
- [90] <http://vcag.ecen.okstate.edu/>
- [91] <http://vlsicad.eecs.umich.edu/BK/ICCAD04bench>
- [92] <http://www.atrenta.com/solutions/spyglass-family/spyglass.htm>

- [93] <http://www.cadence.com>
- [94] <http://www.gaisler.com>
- [95] <http://www.gradient-da.com/products/heatwave.php>
- [96] <http://www.itrs.net>
- [97] <http://www.iwls.org/iwls2005/benchmarks.html>
- [98] <http://www.magma-da.com>
- [99] <http://www.micromagic.com/tools/MAX-3D.html>
- [100] <http://www.r3logic.com>
- [101] <http://www.si2.org/openaccess>
- [102] <http://www.si2.org/openeda.si2.org/projects/oagear>