UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a doctoral dissertation by

Brent Goplen

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by final
examining committee have been made.

Professor Sachin S. Sapatnekar

Name of the Faculty Advisor

Signature of the Faculty Advisor

Date

GRADUATE SCHOOL

Advanced Placement Techniques for Future VLSI Circuits

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Brent Goplen

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Sachin S. Sapatnekar, Advisor

October 2006

Acknowledgements

First of all, I would like to express my deepest gratitude to my advisor, Professor Sachin Sapatnekar, for his guidance and support though my graduate studies. Over the many years that he has been my advisor, he has served as an incredible inspiration and talented mentor in my educational and career development. His keen insights, direction, and persistent encouragement were instrumental in my success in solving tough technical problems. It has truly been a pleasure and an honor working with him.

I would also like to thank Professor Kia Bazargan, Professor Chris Kim, and Professor Antonia Zhai for reviewing my thesis and serving on my Ph.D. defense committee.

I am grateful to Dr. Prashant Saxena for mentoring me during my internship at Intel Labs (CAD Research) and contributing to the work on repeater count reduction. I would also like to thank T. Cao, A. Chowdhary, K. Rajagopal, B. Velikandanathan, D. Kirkpatrick, and P. Cocchini for their support while at Intel.

I owe many thanks to my colleagues in the VEDA Lab and the University of Minnesota for their help and many meaningful discussions: Rupesh, Shrirang, Venkat, Hongliang, Haifeng, Jaski, Tianpei, Sagar, Yong, Yan, Anup, Sanjay, Qunzeng, Baktash, Saeed, Pongstorn, Hushrav, and many others.

I am also grateful to the Semiconductor Research Corporation for funding my graduate studies through an SRC fellowship.

Finally, I would like to thank my family for their continued support and encouragement though the years.

Abstract

The design of future VLSI circuits is becoming more difficult in advanced technologies, and additional design constraints need to be considered in electronic design automation (EDA) tools. These new complexities are linked to the industry-wide trend of technology scaling along a Moore's law trajectory. The enhanced integration densities predicted by Moore's law may be achieved through two approaches: first, by decreasing the feature sizes, to allow a larger number of transistors to be accommodated in a smaller area, and second, through new technologies such as three-dimensional (3D) integration, which stack multiple active layers into a monolithic chip, thereby increasing the number of transistors per unit footprint.

A number of new challenges arise in the design of 3D circuits. These circuits have significantly larger power densities than their 2D counterparts and high thermal resistances between active layers. Unless 3D circuits are carefully designed, they can face severe thermal problems that can reduce their performance and reliability. A second important design concern in 3D ICs is related to the bottleneck in connecting wires between active layers: the maximum allowable density of interlayer vias is greatly restricted due to fabrication limitations.

Another issue that arises in the design of advanced VLSI circuits is related to the problem of interconnect delays. As feature sizes decrease, interconnect delays scale poorly, and must be managed: the delay of an interconnect wire increases quadratically with its length, but through appropriate repeater insertion, this dependency can be brought down from quadratic to linear. However, trends show that the number of repeaters will increase exponentially in future technology generations. Without methods to manage and reduce repeater counts, a breakdown in the design process could result due to the perturbations introduced by repeater insertion.

Several aspects of performance-driven physical design at and around the placement stage are examined in this thesis to address these issues that arise in the design of next-generation circuits. First, placement techniques for 3D ICs are developed, addressing the issue of interlayer via density limitations. It is observed that a tradeoff exists between the number of interlayer vias and wirelength for 3D ICs, and placement and legalization methods are developed using analytical and partitioning-based techniques to minimize both wirelength and interlayer via densities. This allows the wirelengths to be minimized for any desired interlayer via density that can be realized with the given fabrication technology. Thermal placement and legalization are then used to reduce thermal problems by judiciously placing standard cells, the primary sources of heat, within the chip. By using net weighting and additional thermal-directing nets in partitioning-based placement, nets are arranged and cells placed to reduce both power and temperature. During detailed placement, thermal considerations are retained to minimize degradation and make improvements based on the thermal objective. Next, an algorithm for reducing thermal problems in 3D ICs, through the insertion of thermal vias into placed designs, is developed. Thermal vias serve no electrical function, but are used for heat removal from the chip. Using temperature simulations obtained with finite element analysis (FEA), the arrangement of thermal vias is iteratively adjusted until the thermal objective is achieved, with minimal thermal via usage. Finally, a method for ameliorating the increasing number of repeaters, caused by scaling, is developed. This part is primarily directed towards the design of 2D chips, although the basic principles are also applicable to 3D designs. Repeater counts are reduced by dynamically modifying net weights in a context-sensitive manner during global placement and coarse legalization with layer assignment as well as valid inter-repeater distance ranges being modeled.

Contents

List of Tables

List of Figures

1 Introduction

As the technology node progresses and new design paradigms such as three-dimensional integrated circuits (3D ICs) emerge, the design of future VLSI circuits is becoming more difficult and new objectives need to be considered in addition to traditional ones. For example, with increasing transistor packing and power densities, the resulting higher temperatures and thermal gradients are leading to performance and reliability concerns. Trends indicate that these thermal problems will be even more pronounced in the developing technology of 3D ICs. In addition, increased transistor counts make efficient algorithms a necessity in electronic design automation. Limitations on the number of interlayer vias that can be fabricated in 3D ICs causes restrictions on the wirelength improvement that can be obtained with three dimensional technology. In another vein, the increasing number of repeaters needed in future circuits is causing greater design perturbations. To address these issues during and around placement, our research has focused on the tradeoff between interlayer vias and wirelength in 3D ICs, thermal placement of 2D and 3D circuits, placement of thermal vias in 3D ICs, and net weighting to reduce repeater counts. This thesis begins with introductory and background material in Chapters 2 through 4, then presents its new contributions in Chapters 5 through 8, and ends with a concluding chapter.

One of the primary advantages of 3D ICs is the reduction in wirelength that can be achieved by using vertical interconnects between different layers. However in order to make vertical connections, interlayer vias must be created through device layers and greater thicknesses. Consequently, these vias are restricted in number because of their area requirements. They compete with transistors for area, are much larger in size than regular vias, and require a certain amount of area for tolerance in wafer alignment. With restrictions on interlayer via counts, 3D placement methods must be developed with that in mind so that the wirelength can be minimized without using too many interlayer vias. In Chapter 5, a partitioning-based placement method was developed for global placement that explores the tradeoff between interlayer via density and wirelength. Additional procedures were developed to prevent degradation and make improvements to the global placement results during detailed placement.

With thermal problems becoming increasingly prominent, thermal-aware physical design is becoming of particular interest. Temperatures are closely tied to power usage so thermal reduction methods should also reduce power whenever possible. In Chapter 6, a thermal placement method was presented to addresses thermal considerations by selectively moving cells to more favorable thermal environments and by selectively shrinking nets in order to reduce dynamic power. Net weighting is used to reduce power generation in thermally susceptible nets, and cells with high power are attracted to areas of the chip with lower thermal resistances to ambient in order to reduce temperatures. In addition, detailed placement procedures were developed so that the thermal improvements from global placement can be retained and enhanced during legalization. Benchmark circuits produced thermal placements with both lower power and temperatures while retaining the ability to tradeoff between interlayer via counts and wirelength reduction.

Other physical design paradigms and tools can be used to alleviate thermal problems. Incorporating thermal vias into chips is a promising way of mitigating thermal issues by lowering the thermal resistance of the chip itself, and thermal vias can have a larger impact on 3D ICs than on traditional 2D ICs. However, thermal vias take up valuable routing space, and therefore, algorithms are needed to minimize their usage and place them in areas where they would have the greatest impact. In Chapter 7, a thermal via placement method was presented in which thermal vias are assigned to specific areas of the chip and used to adjust the effective thermal conductivities of these areas. This method makes iterative adjustments to these thermal conductivities in order to achieve a desired thermal objective, such as a maximum allowable temperature or thermal gradient. Finite element analysis (FEA) was used in formulating this method and in calculating temperatures efficiently during each iteration. As a result, the method efficiently achieves its thermal objective while minimizing thermal via utilization.

Poor interconnect scaling is expected to result in a repeater count explosion problem. In Chapter 8, this issue is addressed by using net weighting during the placement to reduce repeater counts by nudging nets away from repeater insertion and deletion thresholds. Net weights are iteratively adjusted and take into account layer assignment and inter-repeater distance ranges. Improvements made with global placement are retained after legalization by extending this net weight scheme into coarse legalization. As a result, repeater counts are significantly reduced with minimal impact on wirelength.

2 Three-Dimensional Integration

As the technology node progresses, chip areas and wire lengths continue to increase, causing such problems as increased interconnect delays, power consumption, and temperatures, all of which can have serious implications on reliability, performance, and design effort. Three dimensional technology attempts to overcome some of these limitations by stacking multiple active layers into a monolithic structure, using special fabrication technologies such as thin-film-transistors (TFT) or wafer bonding. Figure 1 shows an example of a 3D IC produced using wafer bonding technology with the *z* dimension being exaggerated to show details. With this particular technology, the bottom layer is created from a bulk wafer, and additional layers on top of this are created from SOI wafers with their bulk substrate removed. Device levels are located at the bottom of each layer, and metal levels are located above the device levels. Interlayer bonds, shown in black, are placed between layers and have interlayer vias passing through them. Interlayer vias electrically connect vertically adjacent areas allowing for greatly reduced wirelengths.

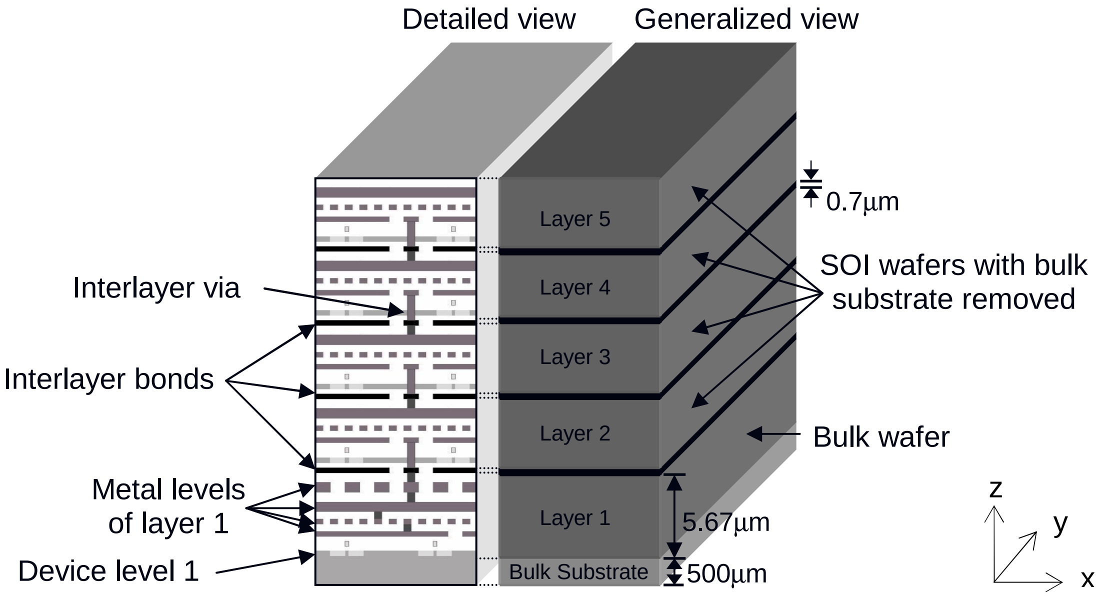

Figure 1. Composition of a 3D IC (adapted from [4])

2.1 Benefits and Obstacles

A number of papers have explored the advantages of 3D ICs over 2D ICs and the obstacles in realizing their potential [1] [2] [3] [4] [5] [6] [7]. By expanding vertically rather than spreading out over a larger surface area, the chip area is better utilized and more importantly, the interconnect lengths are decreased. Not only is the total wirelength reduced, but also the length of the longest wire [8]. As more layers are used, the net distribution is skewed toward shorter wirelengths [9] as shown in Figure 2.

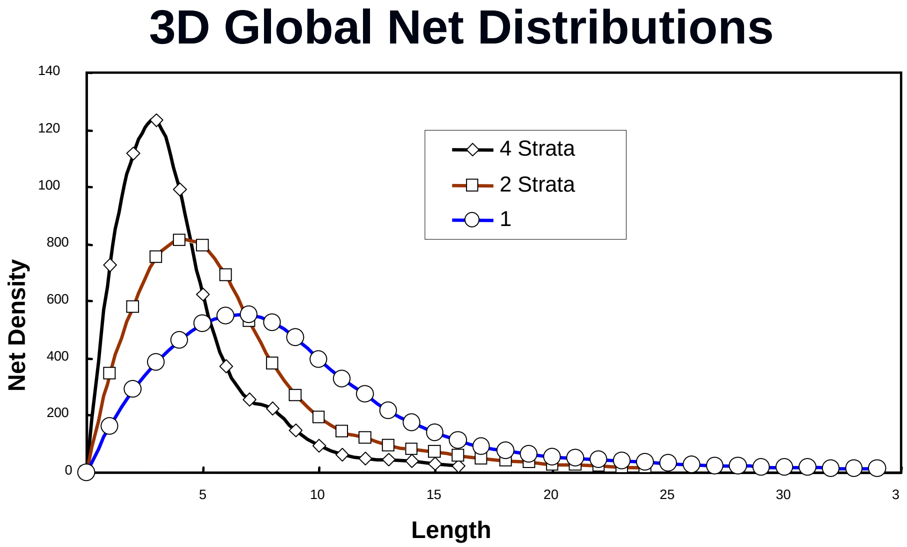

Figure 2. Improvement in the wirelength for 3D ICs from [9].

Adding more layers produces greater wirelength reduction, and larger circuits would benefit more from vertical integration. Consequently, the reduced wirelengths produce greater power efficiency by lowering the power dissipation per transistor and the total power dissipation for the entire chip. In addition, interconnect delays are also decreased, providing improved performance. Stacking transistors in 3D ICs allows the transistor packing densities to be increased without even decreasing feature sizes; this allows for a smaller chip footprint. With classical scaling slowing down [10], 3D technology allows a way for Moore’s Law to keep progressing. 3D ICs also permit previously incompatible technologies (such as digital, analog, RF, SiGe, etc) to be integrated into a single chip as a system-on-a-chip (SoC).

Despite the advantages that 3D ICs have over 2D ICs, there are some obstacles to 3D integration. Fabrication technology needs to be improved to provide more layers, better quality, and lower cost. Interlayer vias are difficult to fabricate and can have significant negative effects on the circuit. They are much larger than regular vias in terms of both cross-sectional area and length [7] [11], and because they go through the silicon, they take area away from transistors. Their size produces larger interconnect capacitances that can have detrimental effects on performance and power. Generally, their usage should be minimized, but doing this limits the potential wirelength reduction that can be achieved with 3D ICs. The tradeoff between wirelength and interlayer via counts will be considered in Chapter 5.

Even though power and temperatures are also rapidly increasing in 2D ICs, thermal problems (higher temperatures and thermal gradients) are expected to be more pronounced in 3D ICs [3] [12] [13] for two reasons. First, despite the power dissipation per transistor being smaller in 3D ICs, the higher packing densities will inevitably cause higher power densities in 3D ICs [14]. Second, 3D ICs have greater thermal resistance along heat conduction paths to the heat sink causing larger temperature rises. Heat has further to travel to reach ambient because of the additional integration layers, and insulating materials created during fabrication between layers also impedes heat dissipation. Higher temperatures and larger thermal gradients can cause variations in the performance across the chip and reliability issues.

These issues make thermal and interlayer via density management central to the development of CAD tools for 3D ICs. With the advent of better processing technologies for 3D ICs, design tools are needed to realize their full potential. Current design tools used for 2D ICs cannot be easily extended to 3D ICs [3] especially when taking into account interlayer vias and thermal effects. In addition, efficient thermal methods are lacking even with 2D ICs.

2.2 Fabrication Technologies

Three-dimensional integration can be achieved in a number of ways ranging from three-dimensional multi-chip-modules (3D MCM) to monolithic 3D ICs. With 3D MCMs, chips are arranged vertically in a 3D package [15] [16]. This is a proven method of vertical integration, but the electrical connections between dies are limited and usually are only made at the chip edges. On the other hand, 3D ICs are composed of tightly stacked device layers in a single chip with vias making direct connections between device layers. Monolithic 3D ICs can be fabricated using crystallization of additional silicon layers or by bonding multiple wafers together.

There are a number of ways that additional layers of silicon can be deposited and crystallized on top of a chip to produce 3D ICs: beam recrystallization [17], silicon epitaxial growth [18], and solid-phase crystallization [19]. Earlier work focused on beam recrystallization and silicon epitaxial growth, but high temperatures used in fabrication tended to degrade the performance of previously fabricated layers. In beam recrystallization, a laser beam is used to heat up and recrystallize a previously deposited layer of silicon. With silicon epitaxial growth, crystallized silicon is deposited at high temperatures or vacuum. Solid-phase crystallization (SPC) can also creates multiple layers of TFTs for vertical integration. With solid-phase crystallization, amorphous silicon is deposited, a seeding agent such as germanium [20] or nickel [21] is used to nucleate grains for Si islands, grains are grown with lateral crystallization, and transistors are fabricated in the Si islands.

Despite the progress that has been made with crystallization methods, wafer bonding is currently the most promising 3D IC fabrication method for three reasons: first, high-performance transistors can easily be made on other wafers prior to bonding; second, underlying device layers are not damaged by the creation of additional layers; and third, heterogeneous integration for SoCs is possible using wafers fabricated with different technologies. With wafer bonding, layers are fabricated separately on different wafers and subsequently bonded together. During the bonding process, the wafer substrates are removed, wafers are precision aligned, and high aspect ratio vias are created for interlayer connections. Wafers can be bonded using polymer adhesive [22] [23], oxide bonding [24], or copper bonding [25] [26].

3 Temperature Calculation

3.1 Introduction

At steady state, heat conduction at a point (*x*,*y*,*z*) within a chip can be described by the following differential equation:

$$K_x \frac{\partial^2 T(x,y,z)}{\partial x^2} + K_y \frac{\partial^2 T(x,y,z)}{\partial y^2} + K_z \frac{\partial^2 T(x,y,z)}{\partial z^2} + Q(x,y,z) = 0 \qquad (1)$$

where T is the temperature, K_x, K_y, and K_z are the thermal conductivities, and Q is the heat generated per unit volume. A unique solution exists when convective, isothermal, and/or insulating boundary conditions are appropriately applied. The nature of the packaging and heat sink determines these boundary conditions. A number of different numerical methods have been used to solve this differential equation for the thermal simulation of integrated circuits, such as finite element analysis (FEA) [27], finite-difference method (FDM) [28], finite volume-based method [29], Fourier method [30], alternating direction implicit (ADI) method [31], and Green function method using discrete cosine transforms [32]. FEA has a number of advantages such as its ability to handle complex geometries and nonhomogeneous materials with fewer nodes, greater efficiency, and more flexibility. This is particularly important when considering thermal vias and other structures that can change the thermal properties of 3D ICs. Flexibility is also needed as 3D fabrication technologies develop and different structures are produced. Therefore, the FEA method from [27] was used in calculating temperatures in these methods. An overview of FEA and its application to 3D ICs is presented in the remainder of this chapter.

3.2 FEA Background

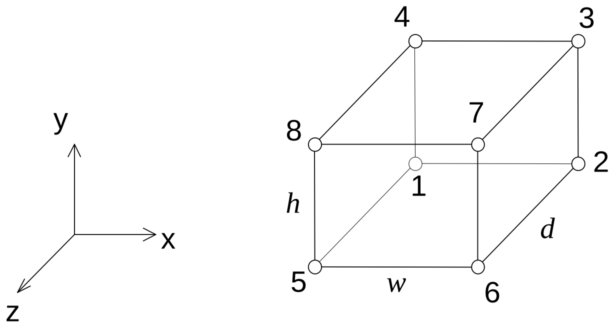

Figure 3. An eight-node hexahedral element.

In finite element analysis, the design space is first discretized or meshed into elements. Different element shapes can be used such as tetrahedra and hexahedra. A four-node tetrahedral element is the simplest possible three-dimensional element, but it does not simulate heat conduction in rectangular structures well. An eight-node hexahedral element, or more specifically a rectangular prism as shown in Figure 3, can simulate heat conduction in lateral directions without aberrations in the prime directions. Higher order elements, which have larger numbers of nodes, generally produce better results, but their derivations are more involved [33]. With FEA, temperatures are calculated at discrete points (the nodes of the element), and the temperatures elsewhere within the element are interpolated using a weighted average of the temperatures at the nodes. In deriving the finite element equations, the differential equation describing heat conduction is approximated within the elements using this interpolation. For an eight-node hexahedral element, a trilinear interpolation function is used to describe the temperatures within each element based on the nodal temperatures:

$$T(x,y,z) = [N]\{t\} = \sum_{i=1}^{8} N_i t_i \tag{2}$$

where $[N\,]=[N_1 \;\; N_2 \;\; \dots \;\; N_8\,]$, $\{t\}=\{t_1 \;\; t_2 \;\; \dots \;\; t_8\}^{\mathrm{T}}$, t_i is the temperature at node i, and N_i is the shape function for node i. The shape functions are determined by the coordinates of the element's center, (x_c, y_c, z_c), and the coordinates at the nodes, (x_i, y_i, z_i), the width, w, height, h, and depth, d, of the element.

$$N_i = \left(\frac{1}{2} + \frac{2(x_i - x_c)}{w^2}(x - x_c)\right)\left(\frac{1}{2} + \frac{2(y_i - y_c)}{h^2}(y - y_c)\right)\left(\frac{1}{2} + \frac{2(z_i - z_c)}{d^2}(z - z_c)\right) \tag{3}$$

From the shape functions, the thermal gradient, $\{g\}$, can be found as follows:

$$\{g\} = \left\{\frac{\partial T}{\partial x} \;\; \frac{\partial T}{\partial y} \;\; \frac{\partial T}{\partial z}\right\}^{\mathrm{T}} = [B]\{t\} \;\text{ where } [B] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \cdots & \frac{\partial N_8}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \cdots & \frac{\partial N_8}{\partial y} \\ \frac{\partial N_1}{\partial z} & \frac{\partial N_2}{\partial z} & \cdots & \frac{\partial N_8}{\partial z} \end{bmatrix}. \tag{4}$$

Similar to circuit simulation using the modified nodal formulation [34], stamps are created for each element and added to the global system of equations. In FEA, these stamps are called element stiffness matrices, $[k]$, and can be derived as follows using the variational method for an arbitrary element type [33]:

$$[k] = [k_c] + \sum_{i=1}^{\#\text{sides}} [k_h^i] \tag{5}$$

$$[k_c] = \iiint_V [B]^{\mathrm{T}} [D][B] dV \tag{6}$$

$$[k_h^i] = \iint_{S_i} h_c^i [N]^{\mathrm{T}} [N] dS \tag{7}$$

where $[D] = \begin{bmatrix} K_x & 0 & 0 \\ 0 & K_y & 0 \\ 0 & 0 & K_z \end{bmatrix}$, $[k_c]$ is the conduction matrix, $[k^i_h]$ is the element convection matrix, and h^i_c is the convective heat transfer coefficient on surface S_i. The symbols K_x, K_y, and K_z represent, respectively, the thermal conductivities in the *x*, *y*, and *z* directions.

3.3 Conductive Stamp

For a right prism with a width of *w*, a height of *h*, and a depth of *d* as shown in Figure 3, the element stiffness matrix is given in Equation (8) as an 8×8 symmetrical matrix with rows and columns corresponding to the nodes 1 through 8 [27]. The matrix describes heat conduction between the eight nodes, and its entries depend only on *w*, *h*, *d*, and the *K*'s as shown in the following matrix. In this matrix, there are conductance terms between every node in the element including diagonally adjacent nodes.

$$[k_c] = \begin{bmatrix} +A & +B & +C & +D & +E & +F & +G & +H \\ +B & +A & +D & +C & +F & +E & +H & +G \\ +C & +D & +A & +B & +G & +H & +E & +F \\ +D & +C & +B & +A & +H & +G & +F & +E \\ +E & +F & +G & +H & +A & +B & +C & +D \\ +F & +E & +H & +G & +B & +A & +D & +C \\ +G & +H & +E & +F & +C & +D & +A & +B \\ +H & +G & +F & +E & +D & +C & +B & +A \end{bmatrix} \tag{8}$$

$$\text{where } A = \frac{K_x hd}{9w} + \frac{K_y wd}{9h} + \frac{K_z wh}{9d},\ B = -\frac{K_x hd}{9w} + \frac{K_y wd}{18h} + \frac{K_z wh}{18d}$$

$$C = -\frac{K_x hd}{18w} - \frac{K_y wd}{18h} + \frac{K_z wh}{36d},\ D = \frac{K_x hd}{18w} - \frac{K_y wd}{9h} + \frac{K_z wh}{18d}$$

$$\text{E} = \frac{K_x hd}{18w} + \frac{K_y wd}{18h} - \frac{K_z wh}{9d},\ F = -\frac{K_x hd}{18w} + \frac{K_y wd}{36h} - \frac{K_z wh}{18d}$$

$$G = -\frac{K_x hd}{36w} - \frac{K_y wd}{36h} - \frac{K_z wh}{36d},\ H = \frac{K_x hd}{36w} - \frac{K_y wd}{18h} - \frac{K_z wh}{18d}$$

3.4 Convective Stamp

If a surface, S_i, of the element is exposed to convective boundary conditions, the convective heat transfer coefficient, $h^i{}_c$, and the element convection matrix, $[k^i{}_h]$, are nonzero. An element convection matrix is created for each side of the element exposed to convective boundary conditions as a 4×4 symmetrical matrix dependent on $h^i{}_c$, w, h, and d. For example, if the surface containing nodes 1, 2, 3, and 4 of the element in Figure 1 is exposed to convective boundary conditions with a convective coefficient of h_c, the stamp would be as follows:

$$[k_c] = \frac{h_c wh}{36}\begin{bmatrix} +4 & +2 & +1 & +2 \\ +2 & +4 & +2 & +1 \\ +1 & +2 & +4 & +2 \\ +2 & +1 & +2 & +4 \end{bmatrix} \tag{9}$$

and $h_c whT_a/4$ would be added to the power dissipation of nodes 1, 2, 3, and 4 where T_a is the ambient temperature. Similar stamps can be obtained for other surfaces of the element.

3.5 Element Mesh and Global Matrix

For the entire mesh, the elements are aligned in a grid pattern with nodes being shared among at most eight different elements. The element stiffness matrices are combined into a global stiffness matrix, $[K_{global}]$, by adding the components of the element matrices corresponding to the same node together. The global power vector, $\{P\}$, contains power dissipated or heat generation as represented at the nodes. This is produced by distributing the heat generated by the standard cells among its closest nodes. A linear system of equations is produced, $[K_{global}]\{T\} = \{P\}$ with $\{T\}$ being a vector of all the nodal temperatures. The system of equations is sparse and can be solved efficiently.

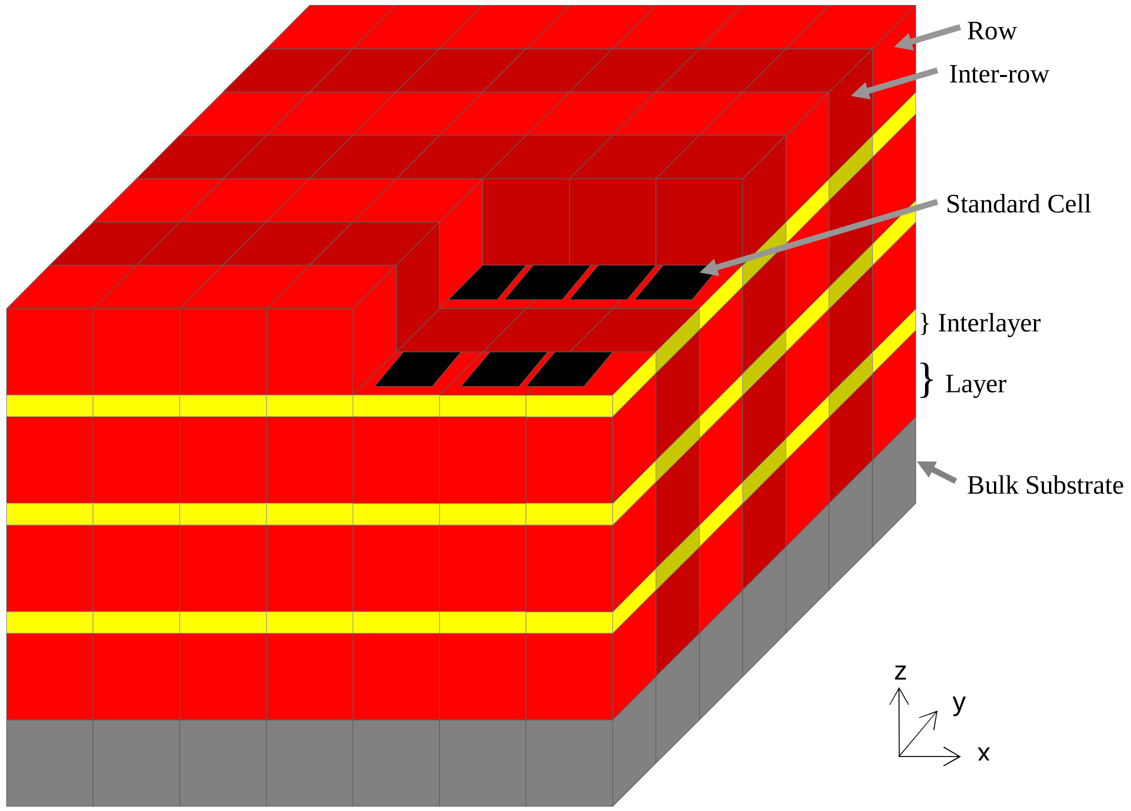

Figure 4. Mesh of a 3D IC.

In these experiments, 3D ICs are meshed into rectangular prism elements (Figure 3) as shown in Figure 4 and can consist of three different element types: bulk substrate, layer, and interlayer elements. Each type of element has different dimensions and thermal conductivities. The bulk substrate elements are located at the bottom of the chip attached to the heat sink. Above the bulk substrate elements are the layer and interlayer elements. In a 3D IC, the interlayers are composed of interlayer vias and bonding materials that hold the layers together, and the layers contain the device and metal levels. The circuits examined in this thesis are standard cell designs which means that they are composed of standard cells (logic blocks) arranged in rows. For more detailed thermal simulations, the bulk substrate, layer, and interlayer elements are further differentiated into rows and inter-rows types with appropriate dimensions. Figure 4 is cut away to show the standard cells located at the bottom of each layer in the rows. The heat generated by the standard cells is appropriately applied to the nodes at these locations.

3.6 Isothermic Boundary Conditions

Isothermic boundary conditions are applied to the global matrix using the following procedure [33], and this results in a reduced, nonsingular system of equations. Rows and columns that correspond to fixed temperature values within the global matrix are eliminated, as are the corresponding values in the power vector, and the remaining values in the right-hand side vector are modified using the fixed temperature values. For example, consider the following system:

$$\left[\begin{array}{c:c} A_{11} & A_{12} \\ \hdashline A_{21} & A_{22} \end{array}\right] \left\{\begin{array}{c} T_1 \\ \hdashline T_2 \end{array}\right\} = \left\{\begin{array}{c} P_1 \\ \hdashline P_2 \end{array}\right\} \tag{10}$$

Here, A_{11}, A_{12}, A_{21}, and A_{22} represent arrays of elements in the global stiffness matrix, T_1 is the vector of unknown temperatures, T_2 is the vector of fixed temperatures, P_1 is the vector of the known power values corresponding to the unknown values, T_1, and P_2 is the vector of the unknown power values corresponding to the known values, T_2. This system can be reduced as follows:

$$[A_{11}]\{T_1\} = \{P_1\} - [A_{12}]\{T_2\} \tag{11}$$

A_{11} is a nonsingular matrix, T_1 contain the unknowns, and the right-hand side is vector of constants so this linear system of equations can now be solved.

4 Placement Methods

4.1 Introduction

The objective of placement is to produce a non-overlapping arrangement of standard cells that minimize wirelength and possibly satisfy some other design constraints such as timing, congestion, or power reduction. Placement is usually divided into two steps: global placement and detailed placement. During global placement, wirelength minimization is the primary objective with cell overlap removal being a secondary objective. Most global placement methods produce placements with residual cell overlaps that require subsequent removal. During detailed placement, obtaining and maintaining an overlap-free placement is the primary objective with wirelength minimization being secondary in importance. Over the years, numerous placement methods have been developed using a variety of approaches. Placement methods can be categorized by the hierarchical and algorithmic approaches that they use. Large-scale placement methods may use top-down, multilevel, or flat hierarchical approaches when addressing scalability concerns [35]. In addition, placement methods can also be categorized into three types: analytical, partitioning, and simulated annealing based methods with some methods using a combination of these three types.

Analytical methods for global placement can be further subdivided into methods that use quadratic (force-directed) placement and nonlinear programming. More recent quadratic placement methods include *Kraftwerk* [36], *FDP* [37] [38], *mFAR* [39] [40] *FastPlace* [41], and *grid-warping* [42]. Quadratic placement methods minimize wirelengths using the same type of quadratic objective function, but primarily differ on their cell spreading mechanism. *Kraftwerk* uses forces derived from an application of Poisson's Equation to spread cells in a manner similar to electrostatic repulsion. *FDP* improved on this forced-directed framework by addressing its inherent instability issues. *mFAR* uses a multilevel approach with fixed points added to the unconstrained quadratic formulation. *FastPlac*e significantly reduced runtimes by using a much simpler cell shifting procedure for spreading, yet obtained comparable results to other leading placers. With *grid-warping*, a grid of the chip is deformed to move cells apart and retain the relative cell ordering. Other analytical methods such as *Aplace* [43] and *mPL* [44,45] uses nonlinear programming to more accurately model wirelength minimization and overlap removal by combining them together into the objective function and constraints. Whereas most quadratic placement methods use a flat approach with linear systems of equations, nonlinear methods need to use a multilevel approach to obtain scalability and only solve the nonlinear formulations directly at the coarsest level. With this approach, the netlist is successively clustered until a small problem size is reached, optimized using nonlinear techniques, and then successively declustered and refined until individual cells are reached.

Gordian [46] is an older placement method that interweaves quadratic placement and partitioning in its formulation. With this method, quadratic placement is used to guide partitioning, and partitioning provides constraints on quadratic placement. Currently, *Capo* [47] [48], *Feng Shui* [49] [50] [51], and *Dragon* [52] [53] are prominent partitioning-based methods that recursively apply fast min-cut hypergraph partitioning in a top-down fashion. *Capo* is a fixed-die placer using recursive bisection with a multilevel Fiduccia-Mattheyses (FM) heuristic applied to larger numbers of cells and simpler, optimal approaches used for smaller numbers of cells. *Feng Shui* is a variable-die placer that uses recursive min-cut partitioning with multi-way partitioning wirelength improvement. *Dragon* uses recursive quadrisection for larger blocks of cells and simulated annealing locally at the bottom level. Simulated annealing is first applied to small blocks of cells and then to small numbers of individual cells. *Timberwolf* [54] is a simulated annealing method for placement using a hierarchical approach. Purely simulated annealing methods are not scalable and have fallen to the wayside with recent advances in placement.

4.2 Force-Directed Placement Methods

In force-directed methods, an analogy to Hooke’s law is used by representing nets as springs, and placements are found by determining the system’s minimum energy state. From the springs, attractive forces are created between interconnected cells and are made proportional to the separation distance and interconnectivity. Cell overlap/congestion and other design criteria are used to derive the repulsive forces that counteract the attractive forces so that the system does not collapse into a single point. After repulsive forces are added, the system is solved for the minimum energy state, i.e., the equilibrium location. Ideally, this minimizes the wire lengths while at the same time satisfying the other design constraints.

Fundamentally, force-directed methodologies involve minimizing an objective function. Nets contribute certain costs to this objective function, and the cost of a connection between nodes *i* and *j* is defined as [36]:

$$c_{ij}\left(\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2\right) \tag{12}$$

where c_{ij} is the weight of the connection between the two nodes and nodes may be either cells or input/output (IO) pads. If the c_{ij} coefficients are combined into a global net stiffness matrix, [*C*], an objective function is obtained for the entire netlist:

$$\frac{1}{2}\{x\}^{\mathrm{T}}[C]\{x\}+\frac{1}{2}\{y\}^{\mathrm{T}}[C]\{y\}+\frac{1}{2}\{z\}^{\mathrm{T}}[C]\{z\} \tag{13}$$

where {*x*}, {*y*}, and {*z*} are the *x*, *y*, and *z* coordinates of all cells and pads. This objective function can be minimized by solving the following three systems of equations:

$$[C]\{x\}=\{f_x\},\ [C]\{y\}=\{f_y\},\ \text{and}\ [C]\{z\}=\{f_z\}\mathrm{x} \qquad \text{(14), (15), \& (16)}$$

The repulsive forces are represented in the force vectors, $\{f_x\}$, $\{f_y\}$, and $\{f_z\}$. In the absence of external repulsive forces, these force vectors would be zero. The net stiffness matrix, [*C*], describes the entire net connectivity and is analogous to a global stiffness matrix in FEA. These three systems of equations can be solved in the same way as in finite element analysis. They can be reduced and made solvable with the physical constraints created by fixed IO pads in a method analogous to that shown in Equation (11). In an iterative force-directed approach, the forces, $\{f_x\}$, $\{f_y\}$, and $\{f_z\}$, are updated at each iteration in the main loop [36]:

1. Forces are updated based on the previous placement.
2. Cell positions are calculated using Equations (14)-(16) with the new forces.
3. Steps 1 and 2 are repeated until convergence.

4.3 Partitioning Placement

Given a graph containing a set of vertices and edges, the objective of partitioning is to divide the vertices into equal partitions so that the number of edges that are cut by the partition boundary is minimized. This can be extended to hypergraphs, which can more accurately model netlists, with weighted vertices representing cells and weighted hyperedges (edges with more than two vertices) representing nets. With this formulation, the goal of partitioning becomes the minimization of total weighted cutsize of the hyperedges with the total vertex weight between partitions being equalized within a certain tolerance. Partitioning placement methods recursively apply hypergraph partitioning to the netlist in a top-down fashion so it is imperative that efficient hypergraph partitioning algorithms are used at each level. Fortunately, fast and scalable multilevel hypergraph partitioner such as *hMetis* [55] and *MLPart* [56] have already been developed and are currently employed in prominent placers such as *Feng Shui* and *Capo*. In multilevel hypergraph partitioning, the hypergraph is successively coarsened, partitioned at the smallest level, and successively uncoarsened with partition refinement. In partitioning placement, cells are assigned to regions that occupy a certain physical portion of the chip. When the region is subdivided by partitioning, cells are reassigned to the resulting subregions. For each region, nets that are connected to the cells inside the region are extracted, a hypergraph is created from these nets, and the external connections of these nets are represented with dummy vertices for terminal propagation [57]. The weights of the vertices are based on the area of the cells they represents, and the weights of the hyperedges correspond to the net weights.

With this basic framework, several advances have been made to better translate the min-cut objective of partitioning into improved wirelength minimization. These modifications address issues concerning cut sequence, cut direction, partition tolerance, boundary location, and terminal propagation. The order in which partitions are processed can have an effect on the final results. A naïve approach would simply process the recursive partitions in a depth-first manner. However, this would produce intermediate placements with some parts finely partitioned while leaving other areas unpartitioned, and consequently making terminal propagation less effective. A better approach is to partition the regions in a breadth-wise manner [48] [58]. Initially, the entire netlist is placed into a single region, and the region is placed into a queue. A main loop is entered with the region at the beginning of the queue being removed for processing. If the region contains more than one cell, it is partitioned into smaller regions which are added to the end of the queue. Subsequent iterations continue with the next region at the beginning of the queue being processed, and the loop is repeated until the queue is empty.

With recursive bisection methods, the cut sequence must more or less alternate the cut direction in order to produce an even distribution of cells in both directions. However, the exact cut sequence used by the method can have a big impact on the final results. Earlier methods simply alternated the cut direction as recursive bisection was applied [59]. Improved results could be obtained when the cut direction is determined for each recursive partition by comparing the region's height and weight, and partitioning perpendicular to the longest dimension [47]. In [49], cut sequence/direction determination was further improved using a dynamic programming approach based on Rent's Rule [60] to determine the optimal cut sequence. It was found that a simple aspect ratio method can be used instead to produces near optimal cut sequences. With this method, if the height to width ratio of the region is above a certain value, the region is partitioned horizontally otherwise it is partitioned vertically. They found that a value of about two for this aspect ratio worked best for determining cut direction on the circuits that they tested, rather than the expected value of one for the aspect ratio as used in [47].

Appropriate balance tolerances for partitioning are used so that subregions are not over-congested and include whitespace considerations [61]. In [47], the problem of *corking* was discussed where large cells are prevented from crossing the partition boundary because of balance constraints. To solve this problem, the balance tolerance is relaxed early in the refinement to allow larger cells to be moved, and is gradually tightened to the desired value as partitioning proceeds. After partitioning, the partition boundaries can be adjusted to better manage whitespace [62] and reduce wirelengths [50]. In [47], cut lines were placed on row boundaries to ease legalization, but this can over-constrain the area balance, produce problematic narrow regions of cells, and degrade wirelengths. The *fractional cut* method from [50] addressed these issues by not restricting horizontal cut boundaries to row boundaries and by placing cell in rows after partitioning placement using legalization. In [61] [62] [63], cut lines are shifted so that each subregion gets an equal percentage of whitespace.

Another issue in modeling global wirelength minimization with min-cut partitioning arises from the interdependence between different regions with terminal propagation and results in the problem of ambiguous terminal propagation. When partitioning a region, the exact positions of cells in other unprocessed regions are not well defined. Consequently, external connections to these cells may result in ambiguity in where to place the dummy vertices used by terminal propagation. A number of solutions have been proposed to address this problem such as *placement feedback* [64], *cycling* [65], and *iterative deletion* [66] [67]. Using *placement feedback*, partitions having ambiguous external terminal locations are repartitioned after these external terminal locations are determined. With *cycling*, regions at the same level are repeatedly repartitioned until terminal propagation stabilizes and no further improvements can be made. *Iterative deletion* begins by duplicating the vertices in question among the possible subregions and iteratively removing the worst duplicate vertex from subregions until only one of the duplicates remains.

4.4 Legalization and Detailed Placement

Detailed placement is the process of taking a legalized or nearly legalized placement and making improvement to it. Legalization completely removes residual overlaps that remain after global placement and may be an integral component of detailed placement or may be used on its own prior to any post-optimization operations performed by detailed placement. Besides overlap removal, legalization may also attempt to minimize perturbations from the original placement or may further improve on the design objectives. Sometimes, detailed placement is separated into two stages with a coarse legalizer (middle placer) such as Mongrel [68] making substantial improvements in overlap removal before a fine-grain legalizer is finally used. After legalization, post optimization methods are sometimes used to obtain additional improvements. The global placement results can be greatly degraded by the legalization process. This is well known in wirelength and timing-driven placement, but may also occur with respect to other design considerations such as repeater counts and thermal effects.

The amount of legalization required depends on the type of global placement algorithm used. Recall that there are three types of global placement algorithms: simulated annealing, partitioning placement, and analytical placement. Algorithms such as simulated annealing generally maintain a fully legalized placement throughout the process and require no legalization afterwards. Some partitioning placement methods [47] assign cells to rows making legalization necessary only on a localized level. More recent advances in partitioning placement makes legalization also necessary for placing cells into rows [50]. In analytical placement, the legalization process is much more involved. Cells, original placed within continuous space, need to be assigned to regions and rows, and the cell densities between different parts of the chip need to be balanced before legalization can make local refinements. Finally, localized movements are performed to remove overlap between individual cells within each region and row.

The classification of detailed placement methods is difficult because they may utilize a number of different algorithms within the same method, and as noted earlier, some amount of legalization may be inherent in the proceeding global placement. There tends to be four main techniques used by detailed placement methods although most use elements from more than one of these: network flow, min-cut, linear placement, and random methods. Many methods also use additional minor algorithms that are not easily classified, but tend to be greedy in nature. In network flow methods [69] [70] [71] [72] [73], a transportation problem is solved using a minimum cost flow or shortest path algorithm. The chip is divided into a grid, and density values are calculated for each region based on the cell area in it. The cost of moving a cell to another region takes into account wirelength or some other design objectives. Generally, cells are moved from the densest regions to the least dense regions following paths that result in the least degradation of placement quality. Min-cut methods [74] use partitioning algorithms to recursively separate a placement into roughly equal parts while minimizing the number of connection between them. Linear placement [75] methods optimize a linear arrangement of cells within a single row. A random method [76], such as one based on simulated annealing, randomly moves cells around locally to improve some objective function.

Generally, detailed placement and legalization has received little attention in the published literature. The main focus of placement has been on global placement, even though the detailed work of finding exact legal positions for cells tends to be much more difficult. More recent papers address the discrepancy between runtime and quality of detailed placement methods [77]. Older methods tend to be either to too slow such as network flow methods or produce poor results. Other papers have suggested that modern global placement methods have reached their limit in reducing wirelengths, but additional improvements can be made by focusing on detailed placement [78].

5 Placement and Legalization of 3D ICs

5.1 Introduction

A key characteristic of 3D ICs is the presence of interlayer vias that electrically connect vertically adjacent areas and allow routing to greatly reduce the wirelengths. However, they are difficult to fabricate, and their densities are limited. Recently, there has been a lot of work in the placement of 3D ICs with all major types of placement algorithms being implemented: nonlinear programming [79], quadratic/force-directed placement [27] [80] [81] [82], and partitioning placement [8] [83] [84] [85] methods. Generally, these methods do not allow the tradeoff between wirelength and interlayer via counts to be fully explored. A number of papers [8] [80] [83] have investigated the tradeoff at the extremes of the tradeoff curve, but not at any arbitrary point in the middle. Either the wirelength has been fully minimized without regard to interlayer via counts, or the interlayer via counts have been fully minimized without regard to the wirelength. Being able to adjust to the desired tradeoff point would be of great utility to a designer so that wirelength can be minimized for any required interlayer via density. The need to optimize both wirelength and interlayer via counts differentiates 3D placement from that of traditional 2D ICs.

Our previous work in the placement of 3D ICs used a force-directed methodology [27]. However, it did not consider the need to minimize the number of interlayer vias and could not trivially be modified to accommodate their consideration. There are a couple main reasons why the effectiveness of force-directed placement is limited in the context of three dimensional integration. First, limitation on the number of interlayer vias necessitate that the number of connections between layers be minimized or controlled. This can be easily accomplished using a min-cut algorithm, but the objective of force-directed methodologies is to minimize the total wirelength in all three directions simultaneously. Using a different weight in the *z*-direction is not very effective with force-directed placement because the system of equations for each direction, Equations (14)-(16), are solved separately and do not directly affect one another. Second, quadratic placements typically require an appropriate arrangement of IO pad connectivity in order to produce good results [41] [82]. This was cited in [41] as the reason why *FastPlace* and other quadratic placement methods can not generate good placements for the IBM-Place benchmarks [86]. For controllability, quadratic placements contain cells within the bounding box formed by IO pads (and other fixed nodes) [39]. Without fixed pads around the chip perimeter, the placement can collapse or become unstable. If IO pads are only present in a single plane perpendicular to the *z*-axis, as would be the case with most 3D ICs, quadratic placement would be unable to place cells beyond this plane into three dimensions without additional off-plane fixed connections or difficult-to-control forces that push cells off the plane. If all cells start in the same *z* position, the initial cell ordering in the vertical direction would make interlayer via minimization difficult.

Considering these issues, it was determined that a partitioning-based approach would be more effective for 3D ICs. Partitioning placement can efficiently reduce interlayer via counts with its intrinsic min-cut objective and can obtain good placement results even when IO pad connectivity information is missing. The two primary objectives of 3D placement are the minimization of wirelength and interlayer via counts. This can be represented with the following cost function to be minimized:

$$\sum_{\text{all nets}} \left(WL_i + \alpha_{ILV} \cdot ILV_i \right) \tag{17}$$

where WL_i is the bounding box wirelength and ILV_i is the number of interlayer vias for net *i* and α_{ILV} is the interlayer via coefficient. Our placement method for 3D ICs is composed of three different steps with each one minimizing the cost function: global placement, coarse legalization, and final legalization. Global placement uses a recursive bisection algorithm in which the cut direction is determined at each level with the cost function in mind. Coarse legalization combines cell shifting for spreading with local and global moves to improve the cost function value. The final legalization places cells in the nearest available slots based on the cost function.

5.2 Global Placement

Our global placement method uses a recursive bisection approach applied to the 3D context and is shown in Figure 5. Regions are defined as containing a number of cells and occupying a certain portion of the placement area. When a region is bisected, two new regions are created from the partitioned list of cells and the divided physical area. Regions are processed in a breadth-wise manner. Initially, all the cells in the netlist are placed into a single region occupying the entire placement area, and this region is placed into the queue. During each iteration of the main loop, the region at the front of the queue is removed for processing. If the region contains more than one cell, it is partitioned and the resulting regions are added to the end of the queue. The main loop continues until the queue is empty. Region partitioning is performed using the `3D_PARTITION` algorithm shown in Figure 6 and will be discussed later in this section.

```
3D_GLOBAL_PLACEMENT(NETLIST,CHIP,$\alpha_{ILV}$) {
   REGION_QUEUE=EMPTY
   NEXT_REGION.CELLS=CELLS OF NETLIST
   NEXT_REGION.BOUNDARIES=BOUNDARIES OF CHIP
   ENQUEUE NEXT_REGION INTO REGION_QUEUE
   WHILE(REGION_QUEUE NOT EMPTY) {
      DEQUEUE NEXT_REGION OFF REGION_QUEUE
      IF(NEXT_REGION HAS MORE THAN ONE CELL) {
         (BOTTOM_REGION,TOP_REGION)=3D_PARTITION(NEXT_REGION,$\alpha_{ILV}$)
         ENQUEUE BOTTOM_REGION INTO REGION_QUEUE
         ENQUEUE TOP_REGION INTO REGION_QUEUE
      }
   }
}
```

Figure 5. 3D global placement algorithm

```
3D_PARTITION(REGION,$\alpha_{ILV}$) {
   SET CUT DIRECTION PERPENDICULAR TO THE LARGEST OF
      REGION.WIDTH, REGION.HEIGHT, AND $\alpha_{ILV}/d_{layer}\bullet$ REGION.DEPTH
   EXTRACT HYPEREDGES CONNECTED TO REGION.CELLS
   CREATE A VERTEX FOR EACH REGION.CELLS BASED ON ITS AREA
   ADD TERMINAL PROPAGATION VERTICES TO HYPERGRAPH
      USING NET CENTERS OF REGION.CELLS
   DETERMINE PARTITION TOLERANCE FROM WHITESPACE OF REGION
   PARTITION HYPERGRAPH INTO BOTTOM_PARTITION AND TOP_PARTITION
   CREATE BOTTOM_REGION AND TOP_REGION FROM BOTTOM_PARTITION,
      TOP_PARTITION, AND REGION.BOUNDARIES
   ADJUST BOUNDARY BETWEEN BOTTOM_REGION AND TOP_REGION
      SO THAT WHITESPACE IS EVENLY DISTRIBUTED
   PUT BOTTOM_REGION.CELLS AT CENTER OF BOTTOM_REGION
   PUT TOP_REGION.CELLS AT CENTER OF TOP_REGION
   UPDATE NET CENTERS OF BOTTOM_REGION.CELLS AND TOP_REGION.CELLS
   RETURN (BOTTOM_REGION,TOP_REGION)

}
```

Figure 6. Region partitioning algorithm for 3D global placement.

Several key improvements are utilized in this partitioning placement method to minimize the cost function given in Equation (17) and allow any point on the tradeoff curve between wirelength and interlayer via density to be reached. The most important improvement in exploring this tradeoff is the cut direction determination method applied at each recursive bisection. Let the *weighted depth* of a region be defined as the depth (*z*-direction) of the region multiplied by α_{ILV}/d_{layer} where α_{ILV} is the interlayer via coefficient and d_{layer} is the layer depth or length of a single interlayer via. The direction orthogonal to the largest of the width, height or *weighted depth* of the region is selected as the cut direction. By doing this, the min-cut objective minimizes the number of connections in the costliest direction at the expense of allowing higher connectivity in the less costly orthogonal directions.

Terminal propagation is used so that connectivity to other areas outside the region being partitioned is taken into consideration. With terminal propagation, nets separated by a partition boundary are represented with dummy terminals to the external locations. In our method, a dummy vertex is created for each net that has external connections and is fixed in the partition closest to the net center. The net center is simply calculated as the average position of cells attached to the net. Partitioning tolerance is calculated to correspond to the amount of whitespace available in the region being partitioned. After partitioning, the cut line is positioned to ensure an even distribution of cell area. The difference in cell area between the two new regions is used to adjust the position of the boundary between them. This results in cut lines not being aligned to row and layer boundaries and makes legalization necessary after global placement.

The region partitioning algorithm used by the global placement method to partition individual regions is shown in Figure 6. It begins by determining the cut direction for the partition based on α_{ILV} and the dimensions of the region. Hyperedges are created from the nets attached to cells in the region, and vertices are created for each cells and weighted based on the cell area. For terminal propagation, dummy vertices of zero weight are created for nets that have externally connections. The partition tolerance is set to correspond to the amount of whitespace, and the hypergraph is partitioned using *hMetis* [55]. Two new regions are created from the partitioned list of cells and area. The boundary position between the regions is set to (x_b a_t + x_t a_b)/(a_b+a_t) where x_b is the bottom edge and x_t is the top edge of the original region, a_b is the total cell area in the bottom region, and a_t is the total cell area in the top region. Cells in the new regions are placed at their region's center, and the net centers of nets attached to these cells are updated. Finally, the algorithm returns the two new regions to the calling global placement algorithm.

5.3 Coarse Legalization

Coarse legalization is used to bridge the gap between global placement and final legalization. The differences in granularity between the two often necessitate the use of these pre-legalization methods as the global placement neglects overlap considerations in favor of simplicity and global wirelength reduction, whereas legalization works locally to remove overlap. Placements produced after coarse legalization still contains overlaps, but the cells are evenly distributed over the placement area so that the fine-gained legalization does not have to worry about the global distribution of cells and can focus on the localized effects of overlap removal. By ensuring an even global distribution of cells, the computationally intensive localized calculations used in final legalization are prevented from acting over an excessively wide area for each cell movement and seriously increasing runtimes. Our coarse legalization method utilizes a spreading mechanism called cell shifting to spreading cells globally, and mechanisms to reduce the cost function value by moving and swapping cells locally and globally. These local and global moves/swaps are necessary to counteract the degradation in the cost function caused by cell position perturbation during legalization. Both the local and global moves/swaps look within an isosurface of the cost function for an available spot that produces the largest reduction in the cost function value. The coarse legalization methods presented here can not only be used after global placement but also in conjunction with a legalizer to make iterative improvements to an existing placement during a post-optimization phase of detailed placement.

5.3.1 Cell Shifting

A cell shifting procedure was developed to overcome limitations discovered with a similar method used by *FastPlace* [41] and improved runtime and performance. The resulting method is extreme effective in rapidly producing an even distribution of cells from placements with highly uneven distributions, while at the same time, minimizing perturbations and degradation in quality. In cell shifting, a mesh of density bins is created, densities are calculated as the ratio of cell area in each bin to bin area, bin boundaries are shifted based on these densities, and cells are moved according to the new bin boundaries. The process is repeated until an even distribution of cells is produced. Cell spreading is achieved as the bin boundaries are shifted to reduce densities, and cells are moved apart. An appropriate application of controlling parameters as presented in [41] allows *FastPlace*'s cell shifting procedure to proceed without major problems, but it also limits its speed and flexibility. Two problems were discovered with *FastPlace*'s cell shifting method that prevents it from being applied more generally, and our cell shifting method addresses these issues to produce better results in significantly less time. The first deals with cross-over of bin boundaries that can change the relative cell ordering, and the second problem deals with preventing unnecessary spreading in order to prevent unnecessary net expansions. These problems will be discussed in detail in the proceeding paragraphs.

In the *FastPlace* method, bin boundaries are moved so that the method tries to average the densities of the bins that the boundary separates. These new bin boundaries depend only on the densities and old boundary positions of adjacent bins. The method does not consider how other bin boundaries are being moved, and a situation can occur in which new bin boundaries cross-over each other and become out of order. When this happens, as cells are being mapped to the new bin boundaries, their relative ordering will change and quality will degrade. *FastPlace* addresses this problem by introducing a parameter to prevent cross-over, but it slows down the procedure, and the problem still occurs if the parameter is not set correctly. Simple examples can be constructed in which this would be problematic, particularly when adjacent bins have vastly different densities such as during an initial placement where cells are clustered together. Our method addresses this first problem by calculating new bin boundary based on the densities of bins across an entire row of bins rather than individually from adjacent bins.

Second, the *FastPlace* method continues to spreads cells apart in areas that are already nearly legalized, even though this would not help reduce cell congestion in other over-congested areas. This method acts only locally because bin boundaries are moved by considering two adjacent bins at a time. The method moves bin boundaries to average the densities between adjacent bins even when both of the bins have densities less than 100%. This allows bins with densities less than 100% to be repeatedly expanded, causing cells to continue to spread well beyond what is necessary for legalization, and consequently wirelengths are increased unnecessarily. This could be particularly problematic when there is a significant amount of whitespace present as in the ISPD05 benchmark suite [87]. It might be argued that this problem can trivially be fixed by preventing or restricting the movement of bin boundaries when the bin densities are less than 100%, but this would inhibit necessary spreading, leaving the placement with an uneven distribution of cells. When there is an excess amount of cell area on one side of the bin in question and space available on the other side, the boundaries of the bin may need to shift great distances in unison to allow expansion on the over-congested side. Because of the local nature of *FastPlace*'s method, it would not be aware of this problem and can not trivially be modified to overcome this. In order to determine whether moving bin boundaries in nearly legalized bins will help elsewhere within the chip, the densities of other bins should be considered rather than just two adjacent bins. Our method acts more globally and addresses this issue by not allowing bins with densities less than 100% to expand, by expanding bin with densities greater than 100% at the expense of shrinking less dense bins, and by preventing bin densities to increase above 100% when they are shrunk.

Our method considers only one single row of bins at a time and expands the widths of over-congested bins at the expense of shrinking sparsely populated bins. This is analogous to a gas law problem in which there is a series of containers with moveable walls and different pressures in each container as shown in Figure 7. In this figure, P is the pressures of an ideal gas within each container and is analogous to bin densities. V is the volumes of each container and is analogous to bin widths. For the purposes of this discussion, let the term, *bin width*, refer to the adjustable dimension of bin and not specifically the size of the bin in the *x* dimension.

P=	0.5	2.0	1.0	0.3	1.5	0.1
V=	1.0	1.0	1.0	1.0	1.0	1.0

Figure 7. Series of containers at various pressures.

According to the gas laws, the shared walls of the containers would move so that the pressures equalize as shown in Figure 8 with P_{new} being the new pressures and P_{old} being the original pressures.

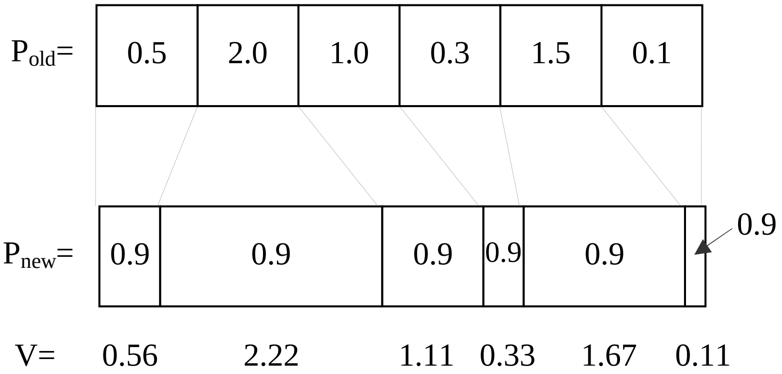

Figure 8. Series of containers after pressure equalization.

If the maximum allowable pressure is 1.0, the pressures are reduced and volumes are expanded by more than is necessary for some of these containers. In our cell shifting scheme, this would cause cells to spread more than is needed in order to produce nearly legal results. Therefore, we should not let the resulting densities (pressures) to be less than 1.0 if they were initially greater than 1.0. The desired result is shown in Figure 9 using our pressure container analogy. Containers with pressures above 1.0 are equalized to 1.0, and container with pressures below 1.0 are equalize to 0.6.

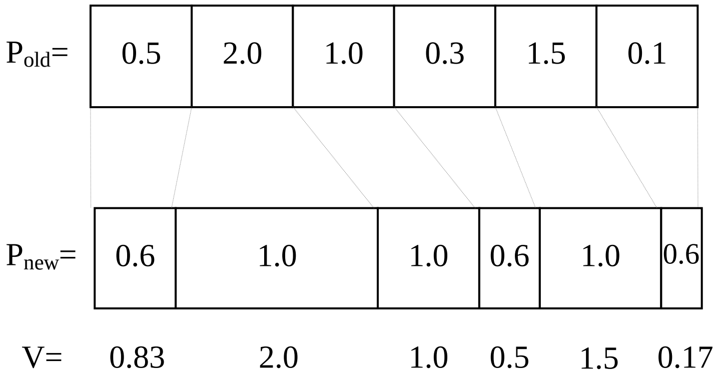

Figure 9. Pressure equalization with pressure reduction limit at 1.0.

With this cell shifting method, only a single row of bins is considered at a time so the relative cell ordering between adjacent rows might be affected if there are large changes in the bin widths during each iteration. In order to slow the process down and minimized the degradation in the relative cell ordering, some inertia should be applied to the bin expansions and contractions. During each iteration, bin densities need to slowly approach 1.0 as shown using our pressure container analogy in Figure 10.

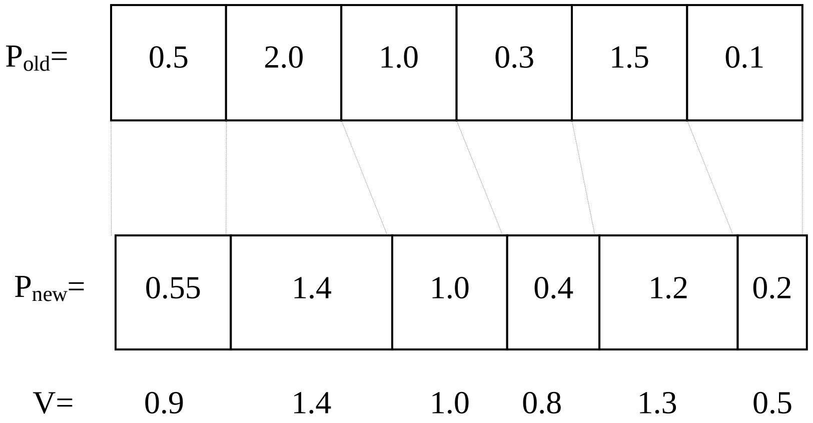

Figure 10. Pressure equalization toward 1.0 with damping.

Graphically, the relationship between bin width and density using damping is shown in Figure 11. In this graph, *W'/W* is the ratio of the new bin width to the old bin width, *d* is the original bin density, a^{lower} is the slope of the curve for densities less than one, and a^{upper} is the maximum slope of the curve for densities greater than one.

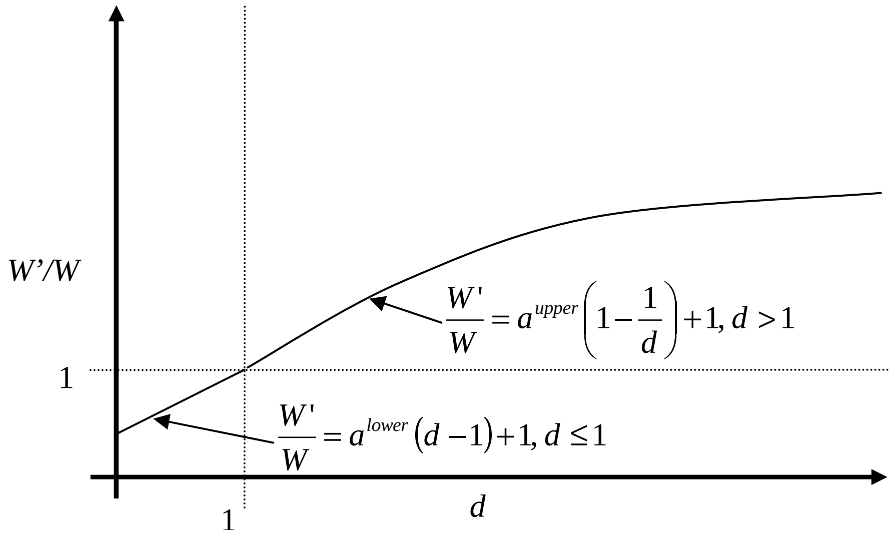

Figure 11. Cell shifting bin width versus density.

In certain circumstances however, all or most of the bins in a row may have densities greater than one. In this case, some bins with densities over 1.0 need to shrink in order to allow other bins with higher densities to expand and reduce in density. This can be accomplished by reducing the bin width by a factor of $b \leq 1$ when the bin density is equal to one as shown in Figure 12.

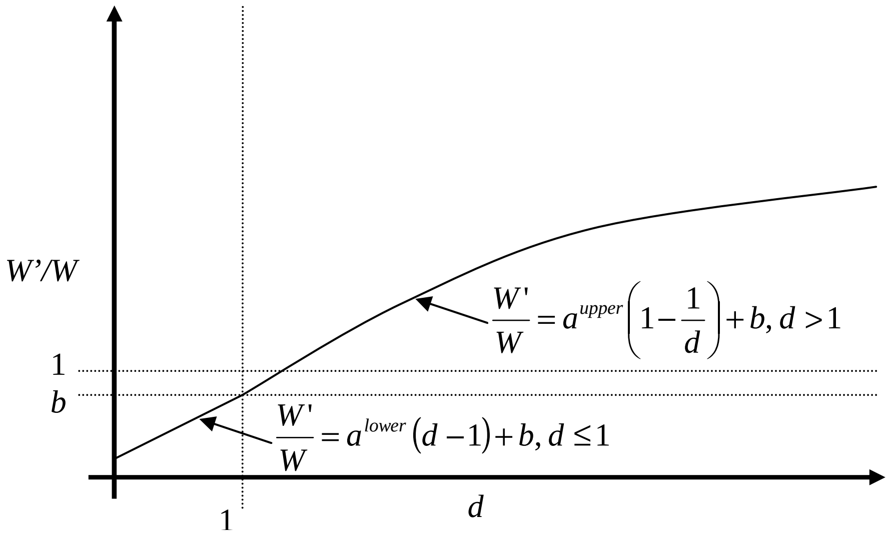

Figure 12. Bin width versus density for the improved cell shifting method.

The parameter b is determined by comparing the bin densities above 1.0 as represented by an upper density sum, s^{upper}, with the bin densities below 1.0 as represented by a lower density sum, s^{lower}, for the same row of bins. These sums are calculated using the following equations:

$$s^{upper} = c^{upper} \sum_{d>1} \left(1 - \frac{1}{d}\right) \tag{18}$$

$$s^{lower} = c^{lower} \sum_{d \leq 1} (1 - d) \tag{19}$$

where c^{upper} and c^{lower} are user defined parameters that determine the slope of their respective curves and are between zero and one. Using s^{upper} and s^{lower}, b, a^{lower}, and a^{upper} are calculated using the following equations:

$$b = \begin{cases} 1 & if\ s^{lower} > s^{upper} \\ \dfrac{1}{s^{upper} - s^{lower} + size} & if\ s^{lower} \leq s^{upper} \end{cases} \tag{20}$$

$$a^{lower} = \begin{cases} \dfrac{c^{lower} s^{upper}}{s^{lower}} & if\ s^{lower} > s^{upper} \\ \dfrac{c^{lower}}{s^{upper} - s^{lower} + size} & if\ s^{lower} \le s^{upper} \end{cases} \tag{21}$$

$$a^{upper} = \begin{cases} c^{upper} & if\ s^{lower} > s^{upper} \\ \dfrac{c^{upper}}{s^{upper} - s^{lower} + size} & if\ s^{lower} \le s^{upper} \end{cases} \tag{22}$$

where *size* is the number of bins in the row.

For the entire three-dimensional mesh of density bins, rows of bins are shifted one at a time for each direction. An example of a row of bins in the *x* direction is shown in Figure 13. In this figure, $d_{i,j,k}$ is the density of bin (i,j,k), $B^x_{i,j,k}$ is the boundary between bin $(i\text{-}1,j,k)$ and bin (i,j,k), and $B^x_{i+1,j,k}$ is the boundary between bin (i,j,k) and bin $(i+1,j,k)$.

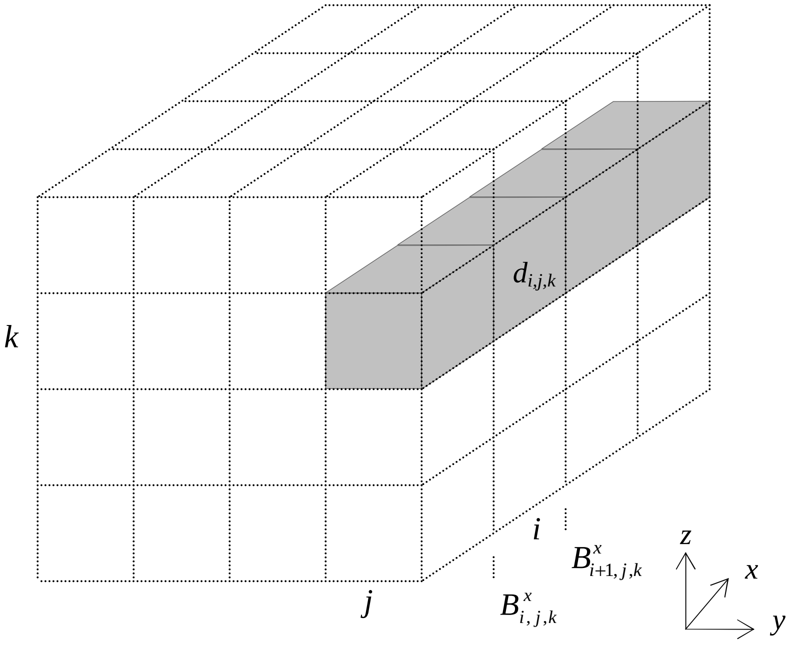

Figure 13. Row of cells in the x direction.

For each row of bins oriented in the *x* direction with a *y* position of *j* and a *z* position of *k*, new bin boundaries, $B'^x_{i,j,k}$, are calculated with the following equation that uses the density, $d_{i,j,k}$, of bins in the row, the specific a^{lower}, a^{upper}, and *b* values determined for this row of bins, and the width, W_x, of the bins.

$$B'^{x}_{i,j,k} = \begin{cases} W_x\left(a^{upper_x}_{j,k}\left(1-\dfrac{1}{d_{i-1,j,k}}\right)+b^{x}_{j,k}\right)+B'^{x}_{i-1,j,k} & if\ d_{i-1,j,k} > 1 \\ W_x\left(a^{lower_x}_{j,k}\left(1-d_{i-1,j,k}\right)+b^{x}_{j,k}\right)+B'^{x}_{i-1,j,k} & if\ d_{i-1,j,k} \le 1 \end{cases} \qquad (23)$$

It should be noted that $B'^{x}_{0,j,k} = B^{x}_{0,j,k}$ and $B'^{x}_{size_x,j,k} = B^{x}_{size_x,j,k}$ where $size_x$ is the number of bins in the x direction. The new bin boundaries in other directions are calculated similarly.

For mapping the x coordinate of cell p, x_p, in bin (i,j,k) to the new bin boundaries, the same formula is used as in *FastPlace* except we apply a different cell movement retention parameter, β_p^x, for each cell.

$$x_p^{\text{target}} = \frac{W_x'}{W_x}\left(x_p - B^{x}_{i,j,k}\right) + B'^{x}_{i,j,k} = \frac{B'^{x}_{i+1,j,k}\left(x_p - B^{x}_{i,j,k}\right) + B'^{x}_{i,j,k}\left(B^{x}_{i+1,j,k} - x_p\right)}{W_x} \qquad (24)$$

$$x'_p = \beta_p^x x_p^{\text{target}} + \left(1-\beta_p^x\right)x_p \qquad (25)$$

where W_x' is the new bin width, x_p^{target} is the target position, and x'_p is the new position of cell p after movement retention is applied. In Equation (24), the target position is in the same relative location between the new bin boundaries as the original position was with respect to the old bin boundaries. In Equation (25), β_p^x is used to slow down the move to these new positions and is between zero and one. The value of β_p^x is adjusted for every cell so that if the move to x_p^{target} causes a large degradation in the cost function, β_p^x is given a smaller value. A simple linear function is used to calculate β_p^x with the maximum cost function degradation for the row producing a β_p^x of β_{min} and the best cost function improvement producing a β_p^x of β_{max}. Typically values used were 0.5 for β_{min} and 1.0 for β_{max}.

5.3.2 Local Moves

In the local move procedure, the cost function from (17) is reduced by moving each cell to a position in its local vicinity that produces the largest reduction in the cost function. Besides simply moving a cell to a new position, swapping positions with other cells is also considered. The target region is defined as a region of bins within n bin widths, heights, and depths away from the original position, and moves and swaps to each bin in the target region are considered. The user defined parameter n is typically set to one so that only adjacent bins are considered and runtime is minimized. A cell move is considered possible if there is space available in the target region, otherwise the move is not considered at all. The cost of this move includes the change in the cost function value from shifting other cells aside in order to make room for the new neighbor. A swap between each of the cells in the target region is also considered. From these possible moves and swaps, the one producing the largest reduction in the cost function is executed.

5.3.2.1 Efficient Cost Reduction Calculation

It would be inefficient to recalculate the value of the entire cost function for every possible move so only the change in the cost function is calculated by making the following partial calculations. Each net's contribution, c_i^{net}, to the cost function in Equation (17) is calculated using Equation (26).

$$c_i^{net} = WL_i + \alpha_{ILV} ILV_i \tag{26}$$

where WL_i is the bounding box wirelength of net i, ILV_i is the number of interlayer vias for net i, and α_{ILV} is the interlayer via coefficient. A cost value, c_j^{cell}, is calculated for each cell j by adding the costs of its attached nets as shown in Equation (27).

$$c_j^{cell} = \sum_{\text{attached nets}} c_i^{net} \tag{27}$$

When a move for cell j is being considered, the change in the cost function, c_j^{move}, is calculated using Equation (29);

$$c'^{cell}_j = \sum_{\text{attached nets}} c'^{net}_i \tag{28}$$

$$c_j^{move} = c_j^{cell} - c'^{cell}_j \tag{29}$$

where c'^{net}_i is the net cost of attached net i if the move is executed and c'^{cell}_j is the cell cost for a moved cell j. After a move is executed, the net costs are updated, and lists of c_i^{net} and c_j^{cell} are maintained so that c_j^{move} can be quickly calculated for each potential move. The cost of a swap, c_{ij}^{swap}, between cells i and j can be calculated with Equation (30). The new cost value, c'^{cell}_i and c'^{cell}_j are calculated with the positions swapped.

$$c_{ij}^{swap} = c_i^{cell} - c_j^{cell} - c'^{cell}_i - c'^{cell}_j \tag{30}$$

5.3.2.2 Implementation

The procedure begins by finding the cost reduction of the best possible move for each cell without actually making any moves. A list of cells is sorted by the best cost reduction, and the algorithm processes cells based on this order, starting with the cell having the largest cost reduction. When processing a cell, previous moves made by the algorithm are taken into account, and the best possible move for a cell is recalculated and executed when it is actually processed. The move executed for each cell may differ from the initially calculated best move used in determining the processing order, but it is important to use the actual state of the placement at the time of processing so that previous moves are taken into account. If no move is found that reduces the cost function, then the cell is not moved. The entire algorithm is shown in Figure 14.

```
LOCAL_MOVES {
     FIND BEST COST REDUCTION FOR EACH CELL
     SORT BASED ON BEST COST REDUCTION
     FOR EACH CELL {
          FIND AND EXECUTED BEST MOVE OR SWAP
          UPDATE COST VALUES
     }
}
```

Figure 14. Local move procedure

5.3.3 Global Moves

The global swap procedure from [77] was modified to include three dimensional considerations. This global swap/move procedure moves cells globally to a target position in the optimal region. An optimal region for a cell is the area in which the cell should be placed in order to achieve the largest possible reduction in the cost function value assuming all other cells remain in their current positions. This is based on the optimal region idea from [77,88] and the cell cost function idea from [75]. Based on the positions of other cells in the attached nets, the optimal region may be a point, line segment, rectangle, or rectangular prism. Taking into account net weights and α_{ILV}, we are more interested in a larger target region around the optimal region that has cost function values less than a certain value. However, care must be taken to prevent the target region from encompassing too large of an area, particularly if there are large differences in the slope of the cost function in different direction. For each cell, the swap or move to the target region that produces the largest reduction in the cost function is performed.

5.3.3.1 Target Region

For each cell, a target region is created around the optimal region in which to search for an optimal move or swap. In [77], the optimal region is determined by finding the minimum and maximum cell positions, excluding the cell in question, of each net attached to the cell, and the boundaries of the optimal region are determined by the medians of these cell positions. However, this is only applicable in minimizing the unweighted bounding box wirelength. In 3D ICs, we must take weighted nets and the interlayer via coefficient into consideration and be able to explore the tradeoff between wirelength and interlayer via counts. In our method, a target region is created around the optimal region using the cell cost function that describes the change in the objective function with respect to cell position. This region is expanded in such a way as to allow wirelength and interlayer vias counts to be traded off.

The procedure for creating the target regions for each cell is shown in Figure 15. At the beginning of this procedure, piece-wise linear functions are created in each direction to represent the cell cost function and are stored as lists of position and cost pairs in C_x, C_y, and C_z. The cell cost function in the *z*-direction, C_z, is scaled appropriately using α_{ILV}/d_{layer}. Another list, V, is created to approximate the relationship between cost, w, in the cell cost function and region volume, v. For each direction *d*, the maximum cost value, w_d^{max}, of C_d is determined, and the volume of the region, v_d, within this cost value is determined from C_x, C_y, and C_z. The desired volume for the target region, v_{target}, is used by V to approximate the cost value, w_{target}, needed to achieve this target volume. The global move algorithm sets v_{target} to be a small percent of the total chip volume representing a fixed number of bins. The boundaries of the target region, p_x^{min}, p_x^{max}, p_y^{min}, p_y^{max}, p_z^{min}, and p_z^{max}, are determined by using w_{target} on C_x, C_y, and C_z.

```
TARGET_REGION(cell $i$,region_volume $v_{target}$) {
    FOR $d$ = $x$, $y$, and $z$ {
        $C_d$ = CELL_COST_FUNCTION($d$,$i$)
    }
    $C_z = \alpha_{ILV}/d_{layer} \cdot C_z$

    V = LIST of (w,v) pairs
    FOR $d$ = $x$, $y$, and $z$ {
        $w_d^{max}$ = max w of $C_d$
        FOR $t$ = $x$, $y$, and $z$ {
            ($p_t^{min}$, $p_t^{max}$)=REGION_SPAN($t$, $C_t$, $w_d^{max}$)
        }
        $v_d = (p_x^{max} - p_x^{min})*(p_y^{max} - p_y^{min})*(p_z^{max} - p_z^{min})$
        ADD ($w_d^{max}$, $v_d$) to V
    }

    SORT V by w
    $j$ = 0
    WHILE(($j$ < 3) and (V[$j$].v < $v_{target}$)) {
        $j = j + 1$
    }
    $m$ = (V[$j$].w-V[$j$-1].w)/($C_d$[$j$].v-$C_d$[$j$-1].v)
    $w_{target}$ = $m$ * ($v_{target}$ - $C_d$[$j$-1].v) + $C_d$[$j$-1].w

    FOR $d$ = $x$, $y$, and $z$ {
        ($p_d^{min}$, $p_d^{max}$)=REGION_SPAN($d$, $C_d$, $w_{target}$)
    }
    RETURN $p_x^{min}$, $p_x^{max}$, $p_y^{min}$, $p_y^{max}$, $p_z^{min}$, and $p_z^{max}$
}
```

Figure 15. Target region constructor

The cost function components, C_x, C_y, and C_z are created using the algorithm shown in Figure 16. The algorithm begins by finding the minimum and maximum cell positions, excluding the original cell, of all attached nets using the algorithm presented in Figure 17. These position values are added to the list C_d for each direction d along with the net weights. The boundaries of the chip are also added to the list to extend the cell cost function's range across the entire chip. The list, C_d, is sorted by position, and the weights stored in the w values of C_d are replaced with the slopes of the cost function at these positions. The slopes of the cost function are determined by subtracting the slope of the adjacent position on the left side from the position's weight. Next, the cost values are determined for each position, and the slope values are replaced with them in the process. The cost value for a position is calculated by subtracting the product of the slope and distance to the adjacent right-hand side position by the cost value of the adjacent right-hand side position. Finally, the cost function values are translated so that the minimum cost value in C_d is equal to zero.

```
CELL_COST_FUNCTION(direction $d$, cell $i$) {
     $C_d$ = LIST of (p,w) pairs
     $w_{total}$=0
     FOR EACH NET $j$ ATTACHED TO CELL $i$ {
          $w_{total}$ = $w_{total}$ + $w_j^{net}$
          ($P_d^{min}$, $P_d^{max}$) = PARTIAL_BOUNDING_BOX($d$, $i$, $j$)
          ADD ($P_d^{min}$, $w_j^{net}$) and ($P_d^{max}$, $w_j^{net}$) to $C_d$
     }
     ADD ($chip_d^{min}$, -$w_{total}$) and ($chip_d^{max}$, $w_{total}$) to $C_d$

     SORT $C_d$ by p
     $C_d$[0].w = -$w_{total}$
     size = length of $C_d$
     FOR $j$ = 1 to size–1 {
          $C_d$[j].w = $C_d$[j].w - $C_d$[j-1].w
     }

     $C_d$[size-l].w = 0
     FOR $j$ = size-2 to 0 {
          $C_d$[$j$].w=$C_d$[$j$+$1$].w–($C_d$[$j$+$1$].p-$C_d$[$j$].p)*$C_d$[$j$].w
     }

     $w_d^{min}$ = min w of $C_d$
     FOR $j$ = 0 to size–1 {
          $C_d$[j].w = $C_d$[j].w - $w_d^{min}$
     }
     RETURN $C_d$
}
```

Figure 16. Cell cost function constructor.

```
PARTIAL_BOUNDING_BOX(direction $d$, cell $i$, net $j$) {
     $min_d$ = position of net center $j$ in $d$ direction
     $max_d$ = position of net center $j$ in $d$ direction
     FOR EACH CELL $k$ in NET $j$ EXCLUDING CELL $i$ {
          $p_k^{cell}$ = position of cell $k$ in $d$ direction
          IF($min_d$ > $p_k^{cell}$)
               $min_d$ = $p_k^{cell}$
          ELSE IF($max_d$ < $p_k^{cell}$)
               $max_d$ = $p_k^{cell}$
     }
     RETURN ($min_d$, $max_d$)
}
```

Figure 17. Bounding box algorithm for net *j* excluding cell *i*.

The REGION_SPAN procedure used by Figure 15 is presented in Figure 18. This algorithm finds the region in which the values of the cost function C_d are less than some targeted cost value, w_{target}. The algorithm returns the minimum, p_d^{min} , and maximum, p_d^{max}, boundaries of the region. Basically, it looks in both directions to find a line segment can contains w_{target} and calculates p_d^{min} and p_d^{max} using an equation that describes the line.

```
REGION_SPAN(direction $d$, list $C_d$, w_value $w_{target}$){
     size = length of $C_d$
     $i$ = 0
     WHILE(($i$ < size) and ($C_d$[i].w > $w_{target}$)) {
          $i = i + 1$
     }
     $m$ = ($C_d$[i].p-$C_d$[i-1].p)/($C_d$[i].w-$C_d$[i-1].w)
     $p_d^{min}$ = $m$ * ($w_{target}$ - $C_d$[i-1].w) + $C_d$[i-1].p

     $i$ = size-1
     WHILE(($i$ >=$0$ ) and ($C_d$[i].w > $w_{target}$)) {
          $i = i - 1$
     }
     $m$ = ($C_d$[i+1].p-$C_d$[i].p)/($C_d$[i+1].w-$C_d$[i].w)
     $p_d^{min}$ = $m$ * ($w_{target}$ - $C_d$[i].w) + $C_d$[i].p

     RETURN ($p_d^{min}$, $p_d^{max}$)
}
```

Figure 18. Region span determination.

5.3.3.2 Implementation

The entire algorithm used for performing global moves and swaps is shown in Figure 19. The method utilizes efficient cost function calculation techniques discussed Section 5.3.2.1. The global move procedure begins by estimating the best possible cost reduction for each cell by calculated the cost difference between the cell's current position and its optimal region. The cell processing order is determined from this estimated cost reduction. For each cell, a target region is created to contain approximately n_{target} bins for move and swap consideration. Moves to every bin and swaps with every cell in the target region are considered, and the best one is executed. A cell move is only considered when there is enough space available in the target region, and the cost of the move includes the effects of moving other cells aside to make room. If no swap or move is found to reduce the cost function value, then the cell remains in it current position. After a move or swap is executed, the cost function is updated.

```
GLOBAL_MOVES {
     FIND BEST COST REDUCTION FOR EACH CELL
     SORT BASED ON BEST COST REDUCTION
     FOR EACH CELL $i$ {
          DETERMINE TARGET REGION FOR CELL $i$
          FIND AND EXECUTED BEST MOVE OR SWAP
          UPDATE COST VALUES
     }
}
```

Figure 19. Global move procedure

5.3.4 Implementation

Its possible to combine the local move, global move, and cell shifting procedures presented in the previous section any number of ways, but in any algorithmic combination, the density profile disruptions caused by local and global moves must be balanced with cell spreading provide by cell shifting. In addition, cell shifting must be performed iteratively to achieve a uniform distribution of cells. Since there is no way to ensure the density abnormalities caused by the local and global moves can be balanced with one iteration of cell shifting, cell shifting must be put into a loop of its own. The following algorithm was developed to perform coarse legalization given a desired maximum bin density, D_{max}, that is close to but not equal to one. The coarse density mesh used by this method has bin widths equal to the width of two cells, bin heights equal to two row heights, and bin depths equal to one layer.

```
COARSE_LEGALIZATION($D_{\mathrm{max}}$) {
     WHILE($d_{max}$>$D_{\mathrm{max}}$) {
          $new\_D_{\mathrm{max}}$= $d_{max}$
          GLOBAL_MOVES
          LOCAL_MOVES
          WHILE($d_{max}$>$new\_D_{\mathrm{max}}$) {
               CELL_SHIFTING
          }
          CELL_SHIFTING
     }
}
```

Figure 20. Coarse Legalization Algorithm

5.4 Final Legalization

With our placement methodology, legalization is broken up into two steps: coarse legalization and final legalization. Coarse legalization, as presented in Section 5.3, is performed to ensure a uniform density of cells across the chip so that final legalization can be performed with minimal global perturbations and expedited run times. Final legalization puts cells into the nearest available space that produces the least degradation in the cost function. Our legalization procedure assumes that the cell distribution has already been evened with coarse legalization and tries to move cells only locally. For each cell, the algorithm looks for an available position near its original position and the search area is gradually expanded until a spot is found. A much finer density mesh is created for the final legalization process than what was used with course legalization. This fine-grained density mesh is created with approximately the same number of bins as cells. The bin widths were set the average cell width, the bin heights corresponded to one row, and the bin depths corresponded to one layer. In coarse legalization, bin densities are calculated by dividing the cell volume that is inside the bin by the bin volume. In final legalization, bin densities are calculated in a more fine-grained fashion by dividing the precise amount of cell width in the bin by the bin width.

5.4.1 External Capacities

To ensure that densities are precisely balanced between different halves of the placement, the amount of space available or lack of space available is calculated for each side of the dividing planes formed by the bin boundaries. An external capacity value, *ec*, is calculated to determine the space available on either side of a bin boundary. If it is a negative number, it indicates how much cell area needs to be moved from one side of the bin boundary to the other. External capacity values are calculated for both the right and left sides of a bin boundary and for the bin boundaries in all three directions as shown in Figure 21.

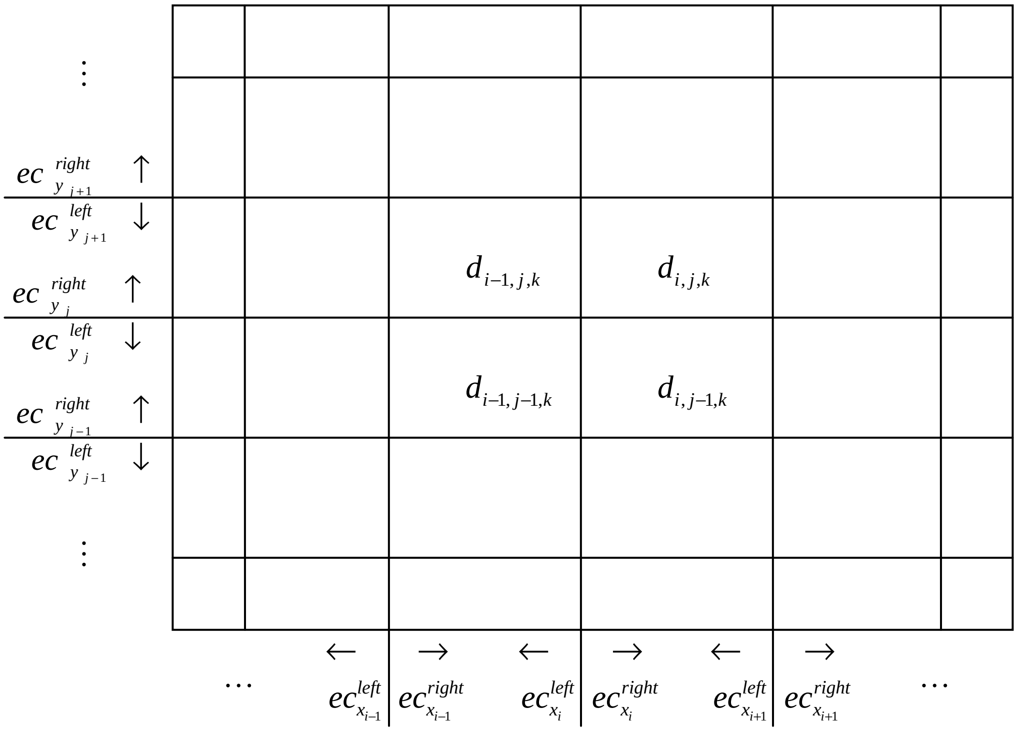

Figure 21. xy-cross-section of density mesh showing external capacities

External capacities are calculated for every common bin boundary in the density mesh using the following equations:

$$ec^{right}_{x_i} = ec^{right}_{x_{i-1}} + size_y size_z - \sum_{j=0}^{size_y-1} \sum_{k=0}^{size_z-1} d_{i-1,j,k} \qquad (31)$$

$$ec^{left}_{x_i} = ec^{left}_{x_{i+1}} + size_y size_z - \sum_{j=0}^{size_y-1} \sum_{k=0}^{size_z-1} d_{i,j,k} \qquad (32)$$

$$ec^{right}_{y_j} = ec^{right}_{y_{j-1}} + size_z size_x - \sum_{k=0}^{size_z-1} \sum_{i=0}^{size_x-1} d_{i,j-1,k} \qquad (33)$$

$$ec^{left}_{y_j} = ec^{left}_{y_{j+1}} + size_z size_x - \sum_{k=0}^{size_z-1} \sum_{i=0}^{size_x-1} d_{i,j,k} \qquad (34)$$

$$ec^{right}_{z_k} = ec^{right}_{z_{k-1}} + size_x size_y - \sum_{i=0}^{size_x-1} \sum_{j=0}^{size_y-1} d_{i,j,k-1} \qquad (35)$$

$$ec^{left}_{z_k} = ec^{left}_{z_{k+1}} + size_x size_y - \sum_{i=0}^{size_x-1} \sum_{j=0}^{size_y-1} d_{i,j,k} \qquad (36)$$

where $d_{i,j,k}$ is the density of bin (i,j,k), $size_x$ is the number of bins in the x direction, $size_y$ is the number of bins in the y direction, and $size_z$ is the number of bins in the z direction.

After the external capacities are determined, a directed acyclic graph (DAG) is constructed in which directed edges are created from bins having an excess amount of cell area to adjacent bins that can accept additional cell area, and this graph is saved as an adjacency list. There are two cases in which these directed edges are created. In both cases, at least one external capacity of the three directions on the target bin side must be greater than zero to signify that there is space available on the target bin side of the bin boundary. In the first case, a directed edge is created from a source bin that has a density greater than one to an adjacent bin that has a density less than one. In the second case, a directed edge is created when an external capacity on the source bin side is less than zero. This signifies that there are too many cells on the source bin side of the bin boundary. Figure 22 illustrate these two cases: edges *a* and *c* show the first case and edge *b* shows the second case.

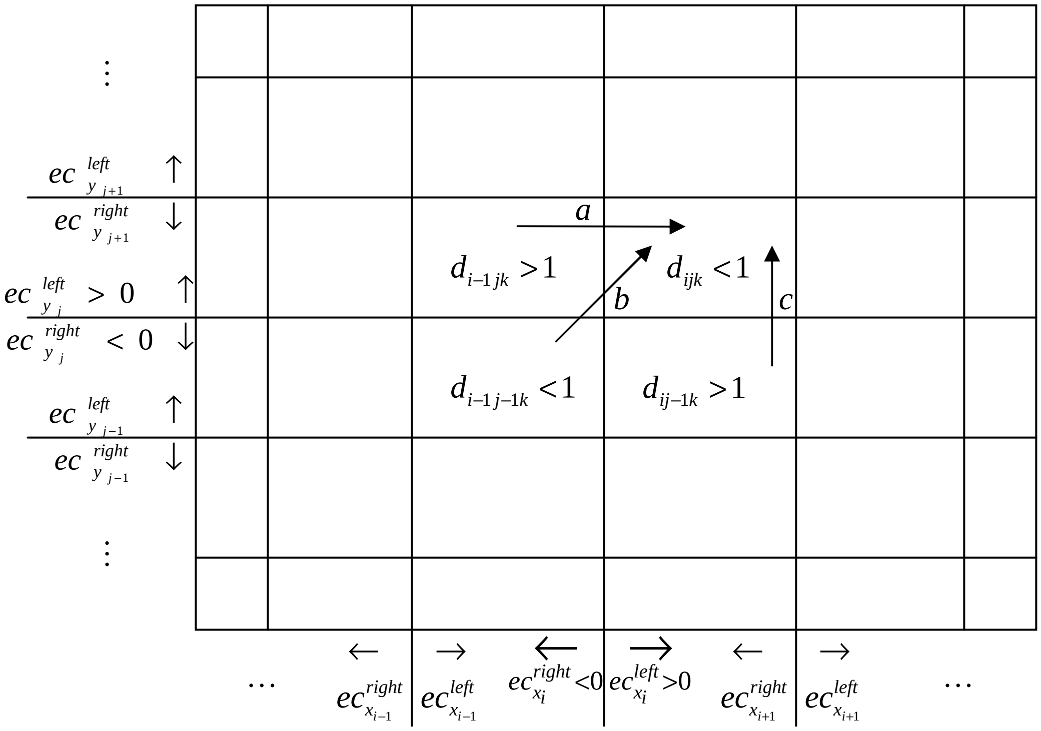

Figure 22. Density mesh showing directed edges

After the adjacency list is created, static timing analysis (STA) is performed to determine the order in which bins should be processes. Bins having the same criticality are grouped together and giving the same priority number. From the static timing analysis, the arrival times determined which bins need to be processed first. Bins with the latest arrival times are given a priority value of zero, and bins with the earliest arrival times are given the highest priority value.

5.4.2 Target Regions

For each cell *i*, a target region is created around the original cell position (x_i,y_i,z_i) in which to look for an available spot to place the cell. If space is not available, the target region is incrementally expanded. A simple approach to expanding the target region would be to increase its size by one bin width, height, and depth at a time. However, if the chip is divided into smaller bins in one direction or weighted differently in different directions, the target region expansion would unfairly favored one direction over another. Precautions must also be taken so that the number of bins being considered does not expand across the entire chip such as if the differences in weights or bin sizes in between directions is quite large. An algorithm was developed that selects a region of bins within a weighted distance away from the original positions such that each direction is given equal consideration with respect to the cost function. The *x* and *y* directions are equally weighted, but in the *z* direction, a weight of α_{ILV}/d_{layer} is used where α_{ILV} is the interlayer via coefficient and d_{layer} is the layer depth or length of an interlayer via. A problem can arise, for example when α_{ILV} is very large, in which the number of bins being considered would expands across an entire layer of the chip. Our target region expansion algorithm prevents this by expanding by at most one bin width, height, or depth per iteration.

The boundaries of the target region are defined by r_x^{top}, the upper *x* boundary, r_x^{bottom}, the lower *x* boundary, r_y^{top}, the upper *y* boundary, r_y^{bottom}, the lower *y* boundary, r_z^{top}, the upper *z* boundary, and r_z^{bottom}, the lower z boundary. Scaling factors are used to normalize the *x*, *y*, and *z* directions and are denoted with s_x, s_y, and s_z. The target region expansion algorithm initializes the target region boundaries and scaling factors as follows:

$$r_x^{top} = x_i^{cell} + \frac{w_x^{cell}}{2} \tag{37}$$

$$r_x^{bottom} = x_i^{cell} - \frac{w_x^{cell}}{2} \tag{38}$$

$$r_y^{top} = y_i^{cell} + \frac{w_y^{cell}}{2} \tag{39}$$

$$r_y^{bottom} = y_i^{cell} - \frac{w_y^{cell}}{2} \tag{40}$$

$$r_z^{top} = z_i^{cell} + \frac{w_z^{cell}}{2} \tag{41}$$

$$r_z^{bottom} = z_i^{cell} - \frac{w_z^{cell}}{2} \tag{42}$$

$$s_x = W_x^{bin} \tag{43}$$

$$s_y = W_y^{bin} \tag{44}$$

$$s_z = \frac{\alpha_{ILV} W_z^{bin}}{d_{layer}} \tag{45}$$

$$s_{\max} = \max(s_x, s_y, s_z) \tag{46}$$

$$s_{\min} = \min(s_x, s_y, s_z) \tag{47}$$

where (x_i^{cell}, y_i^{cell}, z_i^{cell}) is the original position and w_x^{cell}, w_y^{cell}, and w_z^{cell} are the width, height, and depth of cell *i*.

During each expansion, the target region is expanded by the region expanders, a_x, a_y, and a_z, on the tops and bottoms of the region. The region expanders depend on the minimum scaling factor, s_{min}, in order to prevent the region from expanding by more than W_d^{bin} on the top and bottom in any direction *d*.

$$a_x = \frac{s_{\min} W_x^{bin}}{s_x} \tag{48}$$

$$a_y = \frac{s_{\min} W_y^{bin}}{s_y} \tag{49}$$

$$a_z = \frac{s_{\min} W_z^{bin}}{s_z} \tag{50}$$

If both the top and bottom boundaries of the region exceed the boundaries of the chip in that direction, its scaling factor is set to s_{max} so that it's removed from consideration in determining s_{min}. The target region expansion algorithm proceeds as shown in Figure 23 with the boundary and scaling factors of region *R* assumed to be properly initialized as discuss earlier.

```
TARGET_REGION_EXPANSION(region $R$, cell $i$) {
     FOR $d = x, y, z$ {
          IF($r_d^{top}$>chip_top$_d$)and($r_d^{bottom}$<chip_bottom$_d$) {
               $s_d = s_{max}$
          }
     }
     $s_{min}$ = min($s_x$,$s_y$,$s_z$)
     FOR $d = x, y, z$ {
          $a_d = s_d W_d^{bin} / s_{min}$
          $r_d^{top} = r_d^{top} + a_d$
          $r_d^{bottom} = r_d^{bottom} - a_d$
     }
}
```

Figure 23. Target Region Expansion Algorithm

5.4.3 Implementation

The entire final legalization algorithm proceeds as shown in Figure 24. The algorithm is initialized by creating the density mesh and determining the density of each bin. In the process, a list of cells that cover each bin is created and maintained for efficiency purposes. Using the bin densities, the external capacities are calculated, and a directed acyclic graph is created in order to determine the bin processing order. A simple static timing analysis algorithm is used on this graph to determine bin priorities. Next, the cell processing order is determined for each cell by using the maximum priority value of the bins that it covers and a rough estimate of the worst improvement value that can happen within its target region. A worst improvement value is determined for each cell by finding the worst improvement of any move to a position within two target region expansions from the cell's current position. This assumes pessimistically that the only spot available to place the cell will have the largest degradation in the cost function value, and this gives an idea of the criticality of placing the cell in terms of cost function degradation. The later the cell is processed, the more likely it will be placed in the spot with the worst improvement/degradation. However, if the cell is processed earlier, this can be avoided by placing it close to its original position where no cost function change would occur. Cells that are less sensitive to this should be processed later. In order to determine the cell processing order, the list of cells is sorted first by the maximum bin priority and second by the worst improvement values. This is done so that the cell processing order is determined first by overlap removal criticality then by cost function criticality.

```
FINAL_LEGALIZATION {
     CREATE BIN DENSITY DAG
     DETERMINE BIN PRIORITES WITH STA OF DAG

     FOR EACH CELL {
          DETERMINE MAX PRIORITY FROM BINS IT COVERS
          FIND WORST IMPROVEMENT WITHIN TARGET REGION
     }
     SORT CELL LIST BY PRIORITY AND WORST IMPROVEMENT

     FOR EACH CELL {
          INITIALIZE TARGET REGION
          WHILE(AVAILABLE SPOT NOT FOUND) {
               PERFORM TARGET REGION EXPANSION
               FIND AVAILABLE SPOT WITH BEST COST
          }
          EXECUTE BEST MOVE
     }
}
```

Figure 24. Final Legalization Algorithm

Following the previously determined processing order, each cell is placed into its final position. For each cell, a target region is created and gradually expanded until a spot is found to place the cell into a legal position. The initial target region gives the cell's original position first preference in where to place the cell. After the target region is expanded, the algorithm finds all available spots within the target region and places the cell in the one that gives the best improvement in the cost function. If an available spot is not found, the target region is expanded, and the search for an available spot is repeated. In looking for an available spot, the target region is broken up into constituent row segments. For each row segment in the target region, the amount of available space is calculated by subtracting the widths of previously placed cells in the row segment from the width of the row segment. If the width of the cell is less than this value, then space can be made available at the target position by moving cells apart within the row segment in spot that the cell is placed. If space is available, the cost of moving the cell to each bin in the row segment is determined, and the best move is record. The cost of each move includes the change in the cost function value that results from moving already placed cells in the row segment aside to make room for the new cell. Of all the rows with available space, the row with the best move of all is chosen, and that move is executed.

5.5 Post-Optimization

The final legalization algorithm can produce some amount of degradation in the cost function value because its primary purpose is to produce a fully legalized placement. By construction, an already legal placement will not be changed by the final legalization algorithm, and no improvements can be made to an already legal placement by itself alone with respect to cost function reduction. In order to recover the cost function results obtained from global placement and even improve on them, iterative improvements to the legalized placement can be performed using a post-optimization method. A post-optimization procedure, as shown in Figure 25, was developed to combine the iterative improvements provided by coarse legalization with the complete legalization provided by the final legalization algorithm. In the algorithm, the main loop is repeated until the amount of improvement being made is below a certain level. In beginning of the main loop, cells are moved and swapped both globally and locally. Next, cell shifting is performed to even out the densities and prepare the placement for final legalization. At the end of the main loop, final legalization is performed so that legal positions are obtained before the next iteration.

```
POST_OPTIMIZATION {
     WHILE PLACEMENT IS IMPROVING {
          GLOBAL_MOVES
          LOCAL_MOVES
          WHILE($d_{max}$>$D_{max}$) {
               CELL_SHIFTING
          }
          FINAL_LEGALIZATION
     }
}
```

Figure 25. Post-Optimization Algorithm

5.6 Implementation

Our entire placement method for 3D ICs is shown in Figure 26 and combines the global and detailed placement algorithms presented earlier. The same cost function, Equation (17), and interlayer via penalty, α_{ILV}, were used by all algorithms. The placement method begins by extracting the netlist and creating the appropriate data structures. Global placement is performed with a partitioning-based placement of an initial placement. The initial placement is constructed with all the cells placed at the center of the chip with nets and IO pads positioned accordingly. Partitioning placement is performed with terminal propagation initially using these positions. As the placement is recursively partitioned, the positions are refined for terminal propagation. The method uses recursive partitioning until single cells are reached, with cut directions ensuring cost function minimization and cut boundary shifting ensuring even cell distributions. After global placement, detailed placement is performed using the coarse legalization, final legalization and post-optimization methods. After partitioning placement, the placement is already fairly uniform. However, to provide additional improvements to the cost function and guarantee an even density distribution for the final legalization, coarse legalization is performed. Next, final legalization produces a completely legal placement. Finally, post-optimization produces as series of improving legalized placement until it converges, giving the final placement.

```
3D_PLACEMENT($\alpha_{ILV}$){
     EXTRACT NETLIST
     INTIAL_PLACEMENT
     PARTITIONING_PLACEMENT
     COARSE_LEGALIZATION
     FINAL_LEGALIZATION
     POST_OPTIMIZATION
}
```

Figure 26. 3D Placement Method

5.7 Results

The 3D placement method was implemented in C++, run on a Linux workstation with a Pentium 4 3.2GHz CPU and 2GB memory, and incorporated *hMetis* [55] for partitioning. Benchmark circuits from the IBM-PLACE suite [86] were used to test the placement method and are shown in Table 1. The chip areas shown in Table 1 are the footprint areas for the circuits when four layers are used. Vertical dimensions of the 3D ICs were based on the design specifications for MIT Lincoln Labs' 0.18μm 3D FD-SOI technology [24] [89] [90]. However, we simulated the future progression of 3D IC technology by using four layers and face-to-back bonding [26]. From these technology specifications, layer thicknesses were set to 5.7μm, and interlayer thicknesses were set to 0.7μm. Inter-row spaces were set to a quarter of the row height in order to reserve space for interlayer vias, and an addition 5% whitespace was made available within the rows. The lateral (*x* and *y*) cell dimensions of the benchmarks were preserved so that the results can be compared to other work.

Table 1. Benchmark Circuits

name	cells	nets	iopads	chip area (m^2) with four layers
ibm01	12282	13056	246	6.03E-06
ibm02	19321	19291	259	8.60E-06
ibm03	22207	26104	283	9.01E-06
ibm04	26633	31328	287	1.22E-05
ibm05	29347	29647	1201	1.50E-05
ibm06	32185	34935	166	1.17E-05
ibm07	45135	46885	287	1.97E-05
ibm08	50977	49228	286	2.14E-05
ibm09	51746	59454	285	2.21E-05
ibm10	67692	72760	744	3.77E-05
ibm11	68525	78843	406	2.87E-05
ibm12	69663	75157	637	4.15E-05
ibm13	81508	97574	490	3.26E-05
ibm14	146009	150262	517	6.80E-05
ibm15	158244	183684	383	6.34E-05
ibm16	182137	188324	504	8.92E-05
ibm17	183102	186764	743	1.04E-04
ibm18	210323	201560	272	9.88E-05

Two variations of the 3D placement method from Figure 26 were used in these experiments. The course legalization step was skipped in these methods because partitioning placement produced fairly even density distribution in these experiments. To obtain rapid results, a fast 3D placement procedure was developed as shown in Figure 27 with *hMetis* using only one run and local swaps and moves not being removed. To produce more accurate results but at the cost of significantly longer runtimes, a slow 3D placement method was developed shown in Figure 28 with *hMetis* using the 10 runs, optimal regions expanded to 10 bins, and local swaps and moves being used. The tradeoff between these methods will be discussed in Section 5.7.4. In both of these methods, density bins used for cell shifting were given widths of two times the average cell width, heights of two times the sum of the row and inter-row heights, and depths of the layer plus interlayer thicknesses. Cell shifting iterations were terminated when the maximum bin density falls below 110%.

```
FAST 3D_PLACEMENT {
     PARTITIONING_PLACEMENT WITH ONE RANDOM START
     GLOBAL_SWAPS TO ONE OPTIMAL REGION BIN
     CELL_SHIFTING UNTIL UNIFORM DENSITY
     FINAL_LEGALIZATION
}
```

Figure 27. Fast 3D Placement Method

```
SLOW 3D_PLACEMENT {
     PARTITIONING_PLACEMENT WITH TEN RANDOM START
     FINAL_LEGALIZATION
     REPEAT 10 TIMES { /* post-optimization */
          GLOBAL_SWAPS/MOVES TO TEN OPTIMAL REGION BINS
          LOCAL SWAPS/MOVES
          CELL_SHIFTING UNTIL UNIFORM DENSITY
          FINAL_LEGALIZATION
     }
}
```

Figure 28. Slow 3D Placement Method

5.7.1 Tradeoff between Interlayer Via Count and Wirelength

Legalized placements were produced using the fast 3D placement method, Figure 27, over a wide range of interlayer via coefficients, α_{ILV}, for the benchmarks shown in Table 1. From these placements, tradeoff curves were created between interlayer via counts and wirelength as shown in Figure 29 and Figure 30. Figure 29 shows the tradeoff curves for the ibm01 to ibm09 benchmarks, and Figure 30 shows the tradeoff curves for the ibm10 to ibm18 benchmarks. In these plots, as the number of interlayer vias decreases, the wirelength increases for all benchmark circuits. Similar values of α_{ILV} produce similar effects on the wirelength and interlayer via counts across a wide range of circuit sizes. In Table 2, the wirelengths are a given over this range of interlayer via coefficients, and Table 3 gives the interlayer via counts. Figure 31 shows the percent increase in the wirelength as the interlayer via coefficient is increased, and Figure 32 shows the percent reduction in the interlayer via counts as the interlayer via coefficient is increased. As can be seen, large reductions in the interlayer via counts can be obtained with corresponding large increases in the wirelength. Conversely, a large wirelength reduction can be obtained with large increases in the interlayer counts. The percent change in both wirelength and interlayer via counts for the tradeoff are similar across all benchmarks.

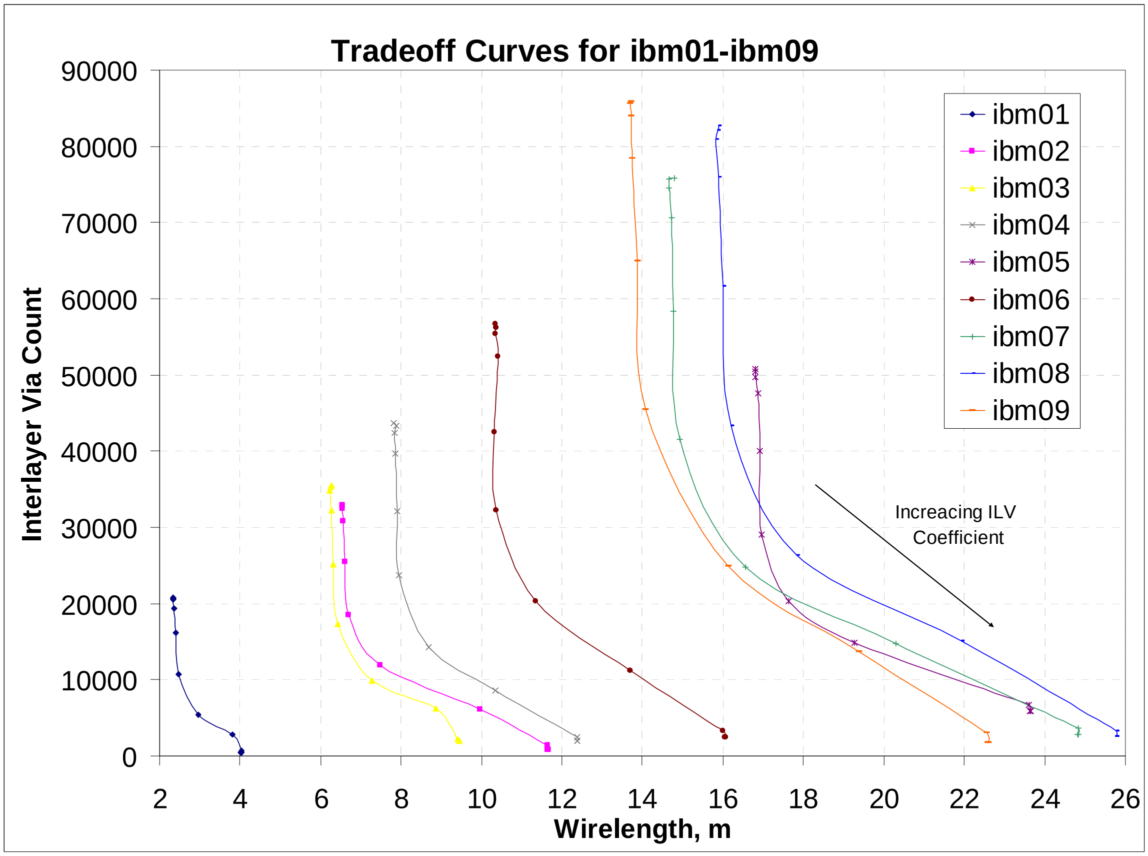

Figure 29. Tradeoff between wirelength and interlayer via count for ibm01 to ibm09.

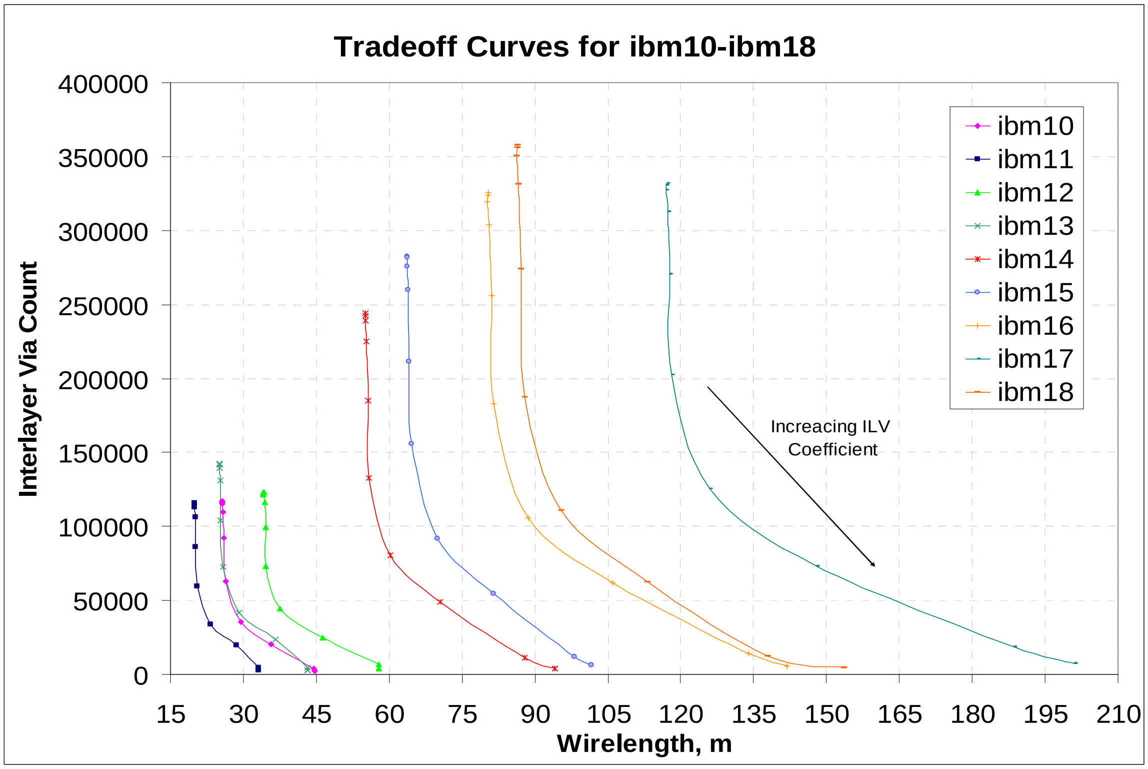

Figure 30. Tradeoff between wirelength and interlayer via count for ibm10 to ibm18.

Table 2. Wirelength for ibm01 to ibm18 as the interlayer via coefficient is varied.

		ILV Coefficient, α_{ILV}							
		8.0E-07	3.2E-06	1.3E-05	5.1E-05	2.0E-04	8.2E-04	3.3E-03	1.3E-02
Benchmark	ibm01	2.34	2.36	2.40	2.46	2.97	3.82	4.03	4.03
	ibm02	6.54	6.57	6.61	6.69	7.49	9.97	11.64	11.64
	ibm03	6.24	6.26	6.31	6.43	7.29	8.86	9.41	9.44
	ibm04	7.83	7.87	7.90	7.95	8.69	10.33	12.37	12.37
	ibm05	16.80	16.87	16.93	16.96	17.64	19.27	23.62	23.64
	ibm06	10.34	10.42	10.31	10.37	11.34	13.69	16.01	16.07
	ibm07	14.67	14.73	14.77	14.92	16.55	20.30	24.83	24.82
	ibm08	15.82	15.89	16.00	16.20	17.84	21.93	25.77	25.79
	ibm09	13.72	13.74	13.89	14.09	16.14	19.37	22.57	22.59
	ibm10	25.62	25.78	26.07	26.29	29.40	35.66	44.57	44.64
	ibm11	19.98	20.16	20.21	20.52	23.16	28.49	33.19	33.19
	ibm12	34.12	34.34	34.55	34.61	37.61	46.35	57.80	57.88
	ibm13	25.04	25.19	25.33	25.72	29.11	36.62	43.27	43.27
	ibm14	55.05	55.25	55.60	55.74	60.18	70.48	87.81	94.07
	ibm15	63.76	63.97	64.07	64.57	69.98	81.44	98.06	101.56
	ibm16	80.21	80.55	81.04	81.46	88.59	105.96	134.03	141.86
	ibm17	116.90	117.42	117.76	118.10	125.90	148.01	188.62	200.98
	ibm18	86.20	86.50	87.11	87.85	95.45	113.10	137.78	153.69

Table 3. Interlayer via counts for ibm01 to ibm18 as the interlayer via coefficient is varied.

		ILV Coefficient, α_{ILV}							
		8.0E-07	3.2E-06	1.3E-05	5.1E-05	2.0E-04	8.2E-04	3.3E-03	1.3E-02
Benchmark	ibm01	20500	19386	16180	10802	5388	2815	745	494
	ibm02	32440	30776	25458	18535	11981	6143	1395	860
	ibm03	34873	32289	25210	17387	9970	6269	2278	1949
	ibm04	42426	39742	32110	23781	14237	8575	2437	2047
	ibm05	49775	47628	39985	28999	20285	14936	6744	5923
	ibm06	55348	52391	42567	32288	20287	11211	3331	2472
	ibm07	74506	70620	58322	41517	24773	14797	3678	2849
	ibm08	80857	75979	61650	43325	26387	15128	3282	2590
	ibm09	83960	78436	65015	45477	24970	13743	3043	1789
	ibm10	115477	109912	92129	62877	35246	20136	4086	2390
	ibm11	112902	106159	86114	59382	33589	19550	4454	2693
	ibm12	121390	116399	99556	73041	44497	24584	6720	3972
	ibm13	139256	130812	103809	72355	41853	23805	4519	2627
	ibm14	238958	224999	184956	132781	80711	48879	11036	4157
	ibm15	275913	259795	211669	155660	91862	54503	11822	6404
	ibm16	319763	303929	256208	182603	105992	61863	13956	5582
	ibm17	327266	312753	270663	202605	125417	73209	18421	7587
	ibm18	350360	331115	273704	187327	110936	62191	12484	4584

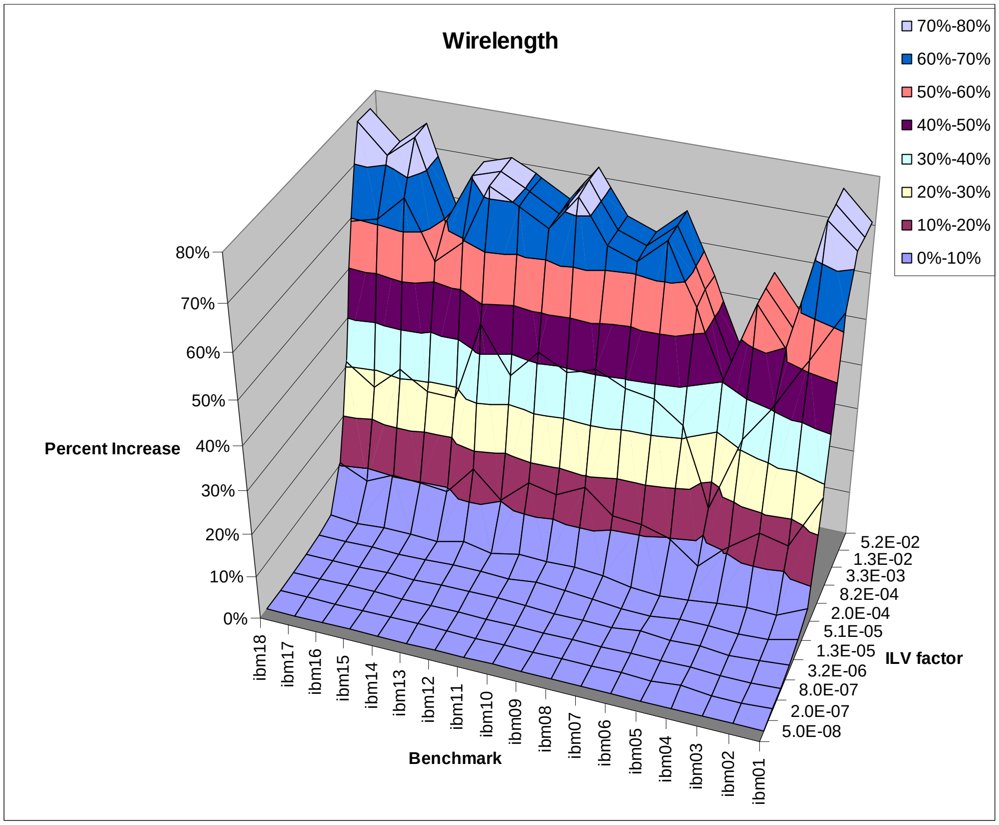

Figure 31. Percent increase in wirelength as α_{ILV} is increased for ibm01 to ibm18.

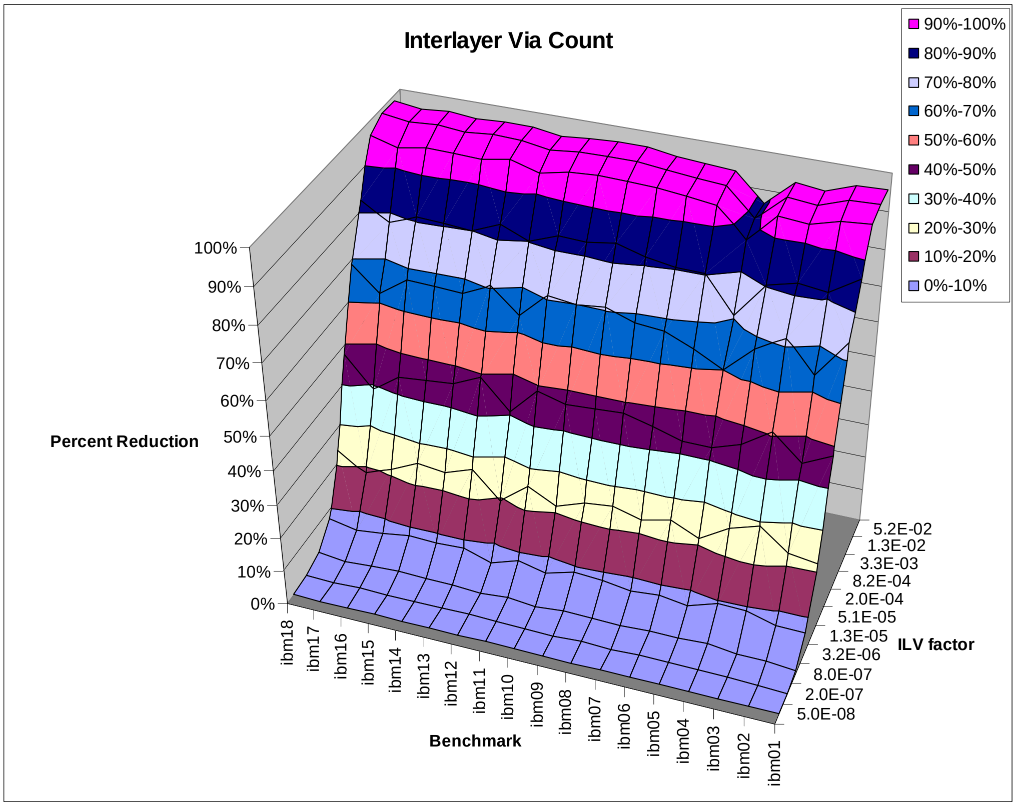

Figure 32. Percent reduction in interlayer via count as α_{ILV} is increased for ibm01 to ibm18.

The tradeoff between interlayer via count and wirelength was examined in more detail for the ibm01 benchmark as shown in Figure 33 and Table 4. To produce more accurate results, the slow 3D placement method from Figure 28 was used. In Figure 33, the interlayer via counts and wirelengths were plotted after both global placement and post-optimization as the interlayer via coefficient was increased from 8×10^{-7} to 1.3×10^{-2}. This range of interlayer via coefficients is centered around the average cell width and height, but the extent of it was empirically determined. The "After Global Placement" curve represents nearly legal placements obtained after partitioning placement and the "After Post-Optimization" curve represents fully legalized placements obtained after the placement is completed. The "Same ILV Coefficient" line segments connected the "After Global Placement" points with their resulting "After Post-Optimization" placements for each interlayer via coefficient value. Table 4 also shows the interlayer via count and wirelength values for each interlayer via coefficient after global placement and after post-optimization. The runtimes for "After Post-Optimization" in Table 4 represent the total runtime of both global placement and post-optimization.

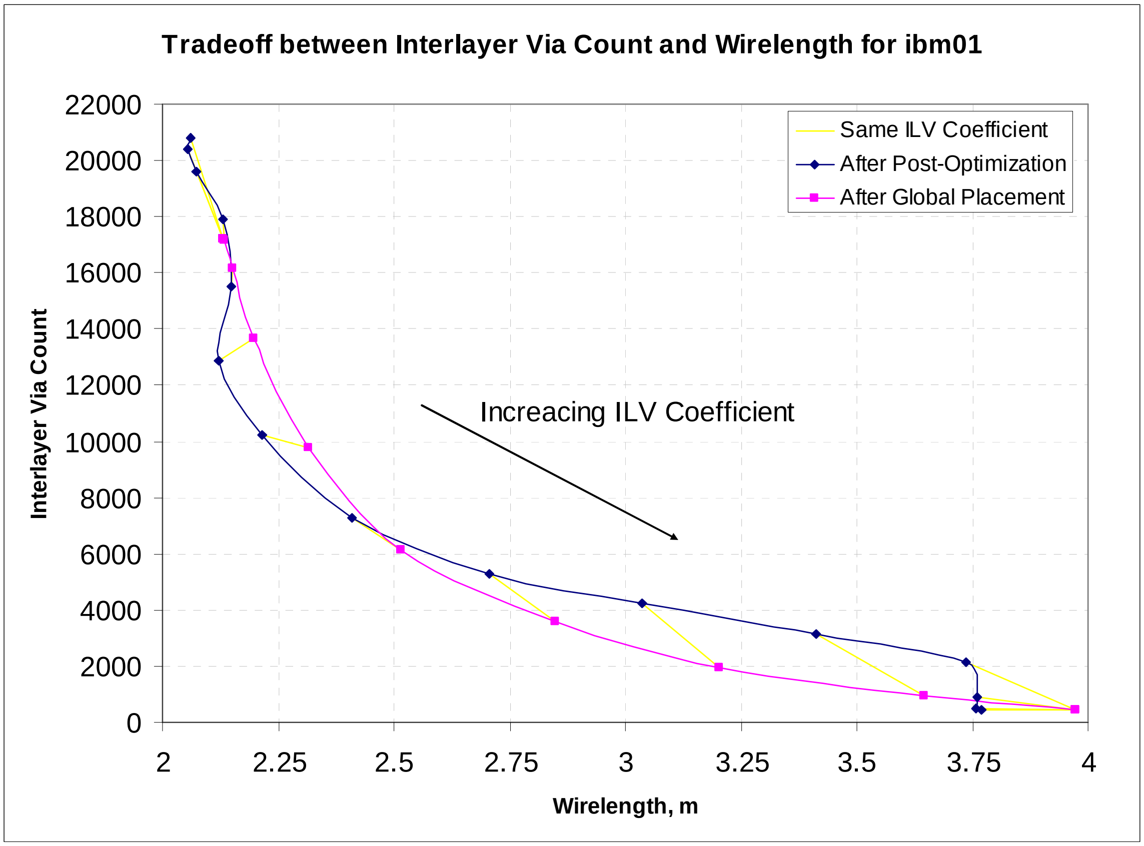

Figure 33. The tradeoff between interlayer via count and wirelength for ibm01.

Table 4. Tradeoff between wirelength and interlayer via count for ibm01.

Interlayer Via Coefficient	After Global Placement			After Post-Optimization		
	Wirelength, m	Interlayer Via Count	Runtime, sec	Wirelength, m	Interlayer Via Count	Runtime, sec
8.00E-07	2.13	17201	40.4	2.06	20782	203.7
1.60E-06	2.13	17201	40.8	2.05	20386	203.5
3.20E-06	2.13	17201	40.1	2.07	19623	208.0
6.40E-06	2.13	17161	40.4	2.13	17890	209.3
1.28E-05	2.15	16173	40.5	2.15	15516	225.1
2.56E-05	2.20	13650	40.8	2.12	12868	216.8
5.12E-05	2.32	9777	40.3	2.21	10207	217.2
1.02E-04	2.52	6117	40.5	2.41	7279	210.6
2.05E-04	2.85	3581	40.5	2.71	5307	202.1
4.10E-04	3.20	1968	41.3	3.04	4258	202.4
8.19E-04	3.65	927	40.2	3.41	3139	193.5
1.64E-03	3.97	464	40.9	3.74	2129	191.6
3.28E-03	3.97	464	40.8	3.76	890	193.8
6.55E-03	3.97	464	40.9	3.76	495	196.2
1.31E-02	3.97	464	40.2	3.77	464	191.5

The tradeoff curve after global placement is fairly smooth and has distinct end points when the interlayer via coefficient is either very high or very low. Legalization and post-optimization distorts the tradeoff curve, particularly when the endpoints are reached and partitioning placement can not produce any further improvements. At each point of the curve, post-optimization seems to prefer wirelength reduction over interlayer via count reduction. The wirelengths vary by almost two times and interlayer vias counts vary by over forty times across the tradeoff curves.

5.7.2 The Effect of Layer Count

The number of layers was increased from one to ten for the ibm01 benchmark, and the resulting tradeoff curves between wirelength and interlayer via count were plotting in Figure 34. Figure 35 shows the interlayer via counts as α_{ILV} and the number of layers are varied, and Figure 36 shows the wirelengths as α_{ILV} and the number of layers are varied. In Figure 34, the tradeoff curves are shifted to shorter wirelengths as the number of layers is increased. The "1 layer" curve is actual just a single point on the wirelength axis because it can not have any interlayer vias. As more layers are used, greater amounts of wirelength reduction can be achieved with the same number of interlayer vias. With more layers, more interlayer vias can be used to produce even greater wirelength reductions.

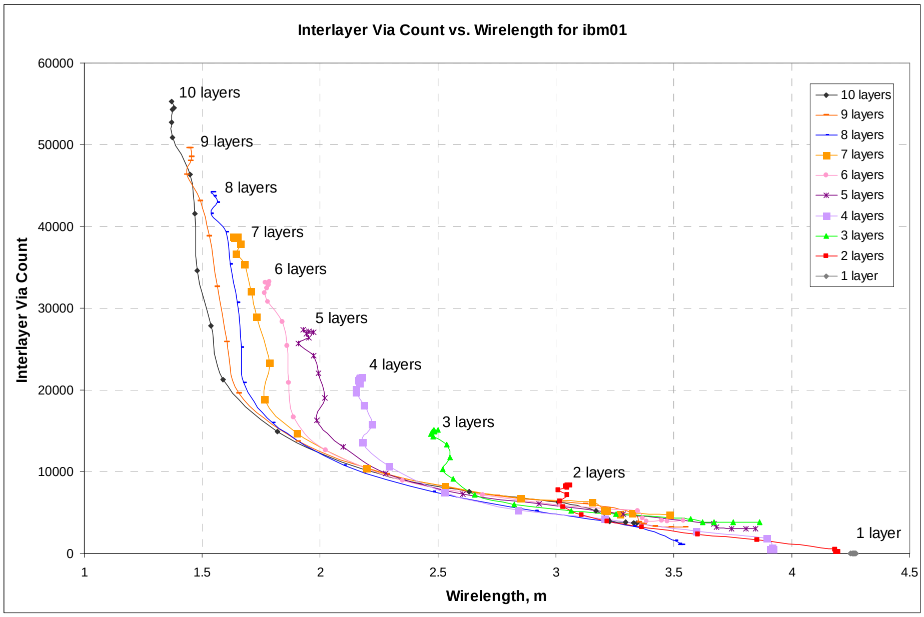

Figure 34. Tradeoff curves between wirelength and interlayer via counts as the number of layers is increased for ibm01.

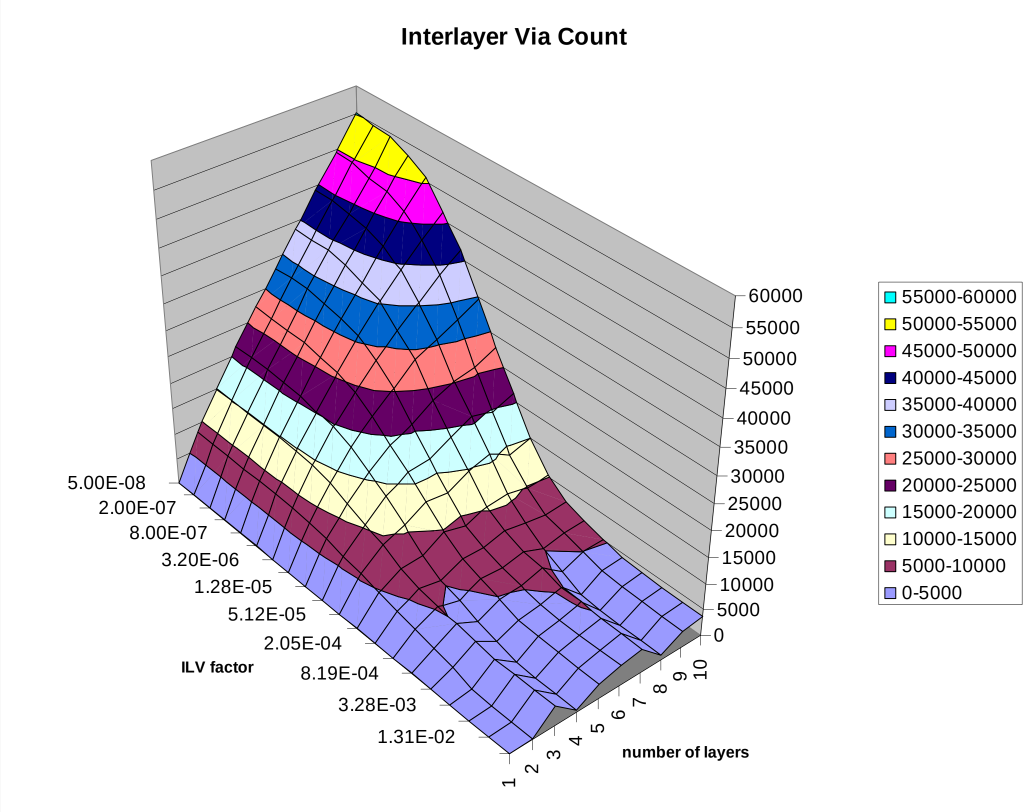

Figure 35. Interlayer via counts as α_{ILV} and the number of layers is varied for ibm01.

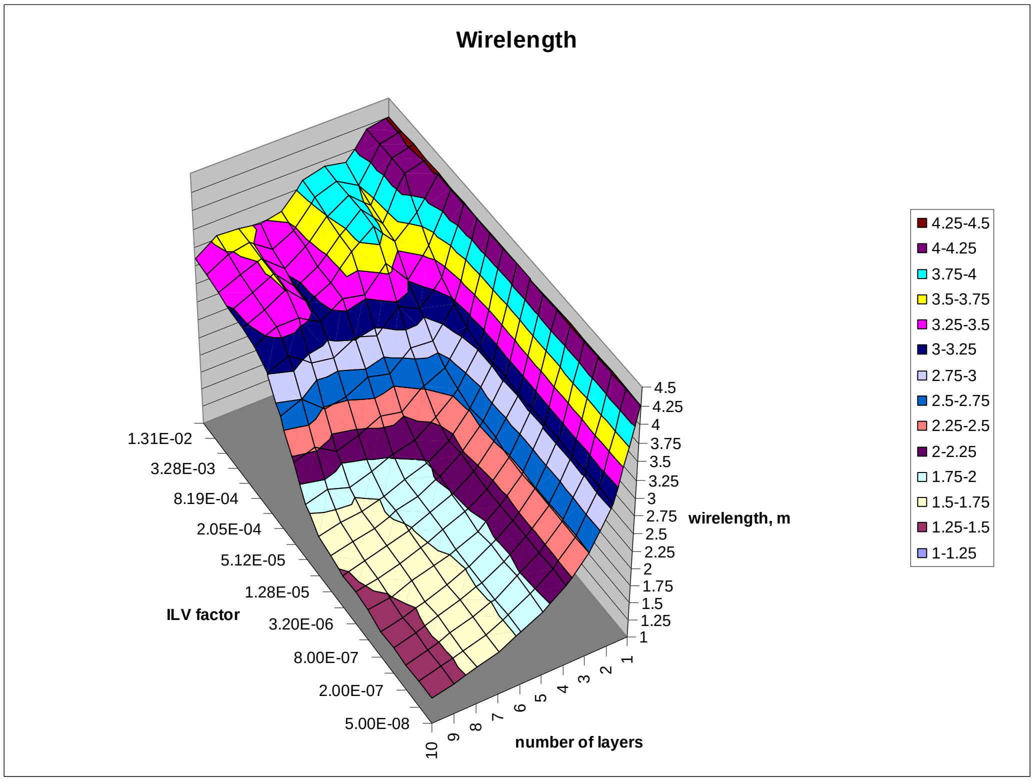

Figure 36. Wirelength as α_{ILV} and the number of layers is varied for ibm01.

5.7.3 The Effect of Post-Optimization

The effect of post-optimization on the ibm01 benchmark was examined using the slow 3D placement method from Figure 28 and an interlayer via coefficient of 1×10^{-4}. The progress during post-optimization is shown in Figure 37 with the maximum bin density plotted above the percent change in the objective function from the partitioning placement results. The results after each placement step is shown starting with partitioning placement (PP), and continuing through iterations of final legalization (FL), global swaps and moves (GSM), local swaps and moves (LSM), and cell shifting (CS) steps. The plot of the maximum density shows how far the placement is from being legal. A maximum bin density of 100% or less represents a legal placement. The solid line, "Legalized Objective Function Value," connecting the peaks of the "Objective Function" curve shows the values when the placement is legalized. Figure 38 shows these plots with respect to runtime.

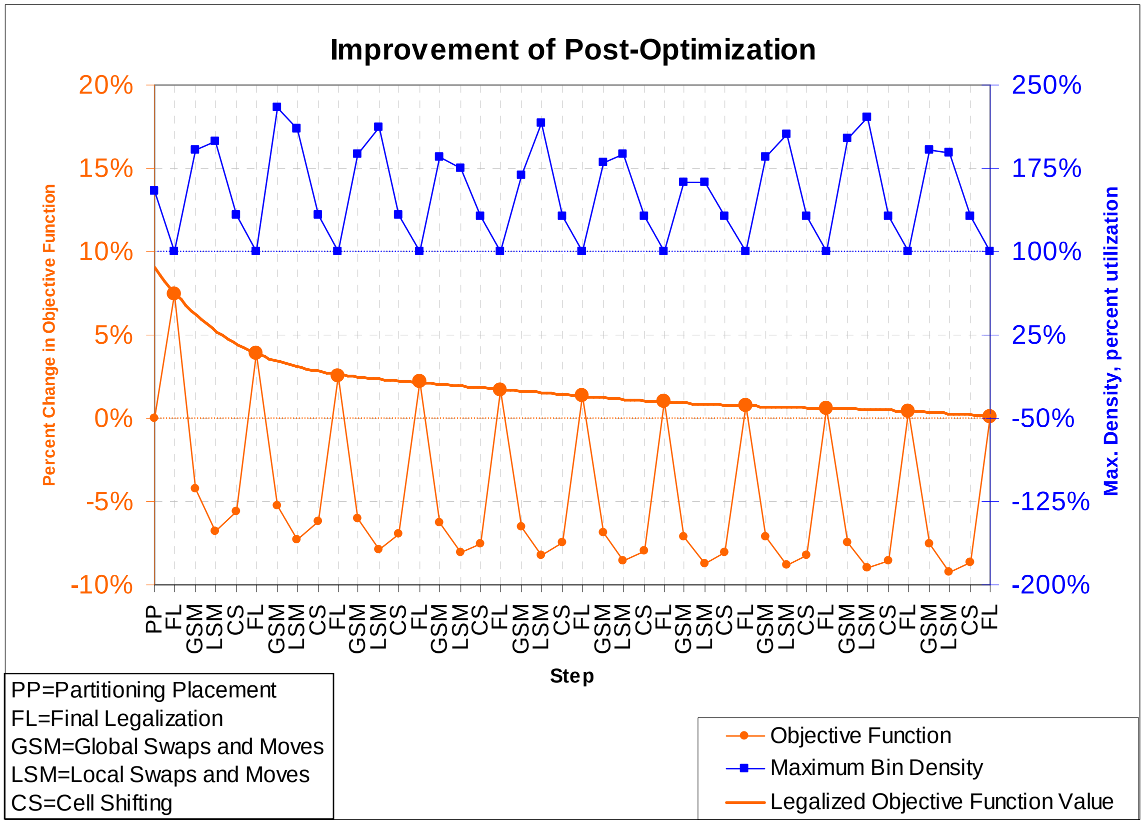

Figure 37. Improvement made by post-optimization on ibm01.

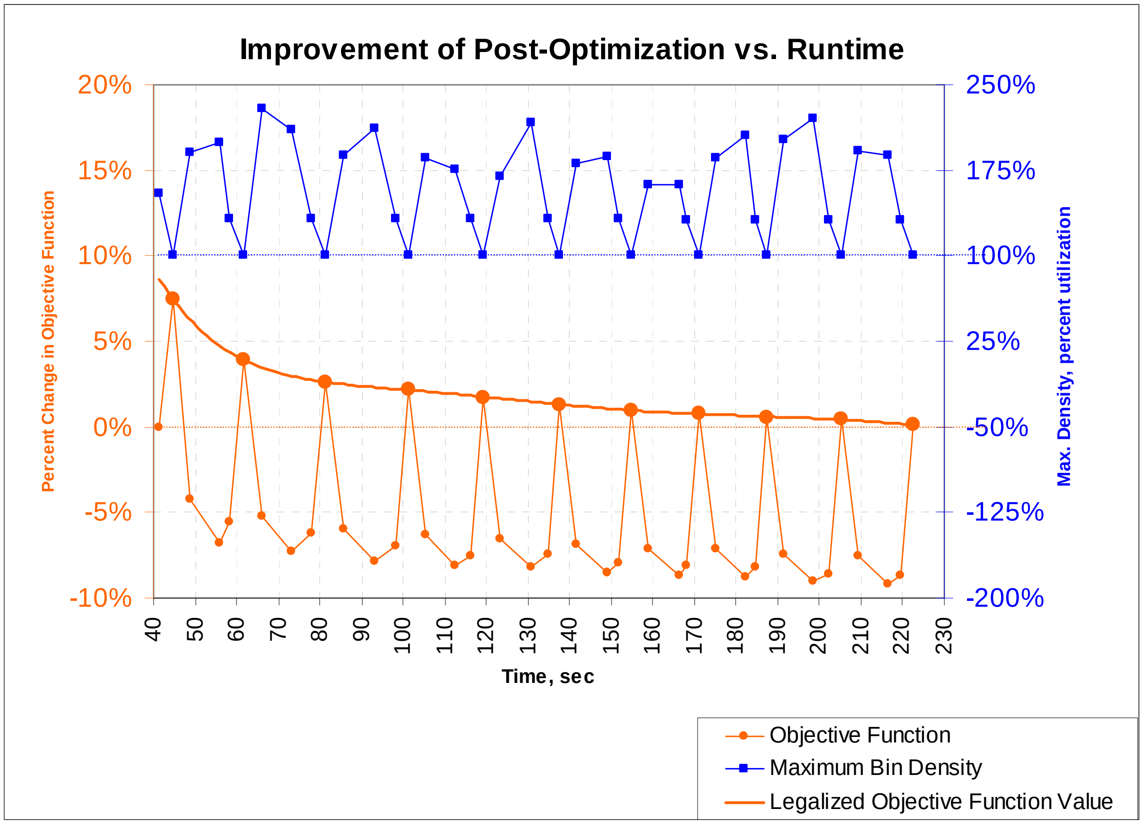

Figure 38. Runtime of post-optimization on ibm01.

After partitioning placement, the placement is legalized with an execution of the final legalization procedure. When global swaps and moves are performed, significant reductions are made in the objective function, but the maximum bin density increases rapidly. Local swaps and moves produce less improvement, and the maximum bin density also increases less rapidly. Cell shifting makes significant reductions in the maximum bin density with only minor degradations in the objective function. Final legalization brings the placement to a fully legalized condition, but also increases the objective function value rapidly. The largest improvements are made in the first few iterations. The amount of improvement per iteration decreases as post-optimization progresses. Partitioning placement is completed in little over 40 seconds, and each post-optimization iteration takes about 20 seconds. Global swaps and moves produce the largest reduction in the objective function per unit time, and cell shifting gives the largest reduction in the maximum bin density per unit time.

5.7.4 Run-Time Analysis

The runtime was examined using the fast 3D placement method, Figure 27, and the slow 3D placement method, Figure 28, on the benchmark circuits from Table 1 and with an interlayer via coefficient of 1×10^{-4}. In addition, the tradeoff between runtime and quality was examined between the fast and slow 3D placement methods. To examine the runtimes of each step, the slow 3D placement method was used with only one post-optimization iteration. The runtime after each step is plotted in Figure 39, and the percent runtime consumed by each step is shown in Table 5. In this figure, "After GP" means after global placement, "After FL" means after the first legalization, "After GSM" means after global swaps and moves, "After LSM" means after local swaps and moves, "After CS" means after cell shifting, and "Total" means after the final legalization. As can be seen, each step is performed in nearly linear times with global placement consuming a majority of the runtime.

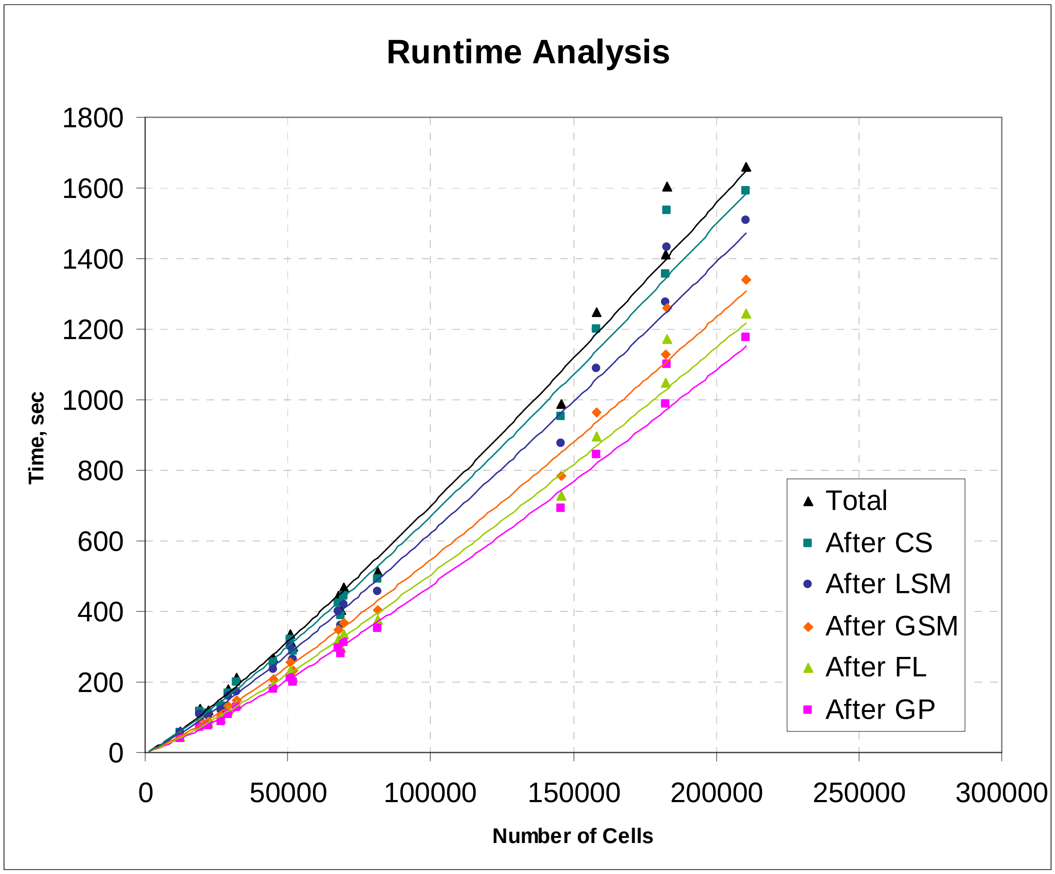

Figure 39. Runtime analysis of 3D placement.

Table 5. Runtime composition for benchmarks ibm01 to ibm18.

Bench-mark	Global Placement	First Legalization	Global Swaps and Moves	Local Swaps and Moves	Cell Shifting	Final Legalization	Total Runtime, sec
ibm01	67%	5%	7%	12%	4%	5%	60.6
ibm02	59%	7%	8%	14%	4%	7%	123.7
ibm03	64%	5%	6%	12%	8%	5%	118.5
ibm04	65%	5%	7%	12%	7%	5%	137.8
ibm05	61%	6%	8%	14%	6%	5%	178.6
ibm06	60%	5%	6%	11%	14%	5%	211.2
ibm07	67%	4%	6%	11%	8%	4%	267.5
ibm08	63%	6%	7%	13%	6%	5%	337.9
ibm09	66%	5%	6%	11%	7%	4%	301.1
ibm10	67%	5%	6%	11%	6%	5%	443.9
ibm11	69%	4%	6%	10%	7%	4%	405.4
ibm12	66%	5%	6%	12%	5%	5%	468.6
ibm13	68%	5%	6%	10%	7%	4%	513.6
ibm14	70%	4%	5%	10%	7%	4%	987.3
ibm15	68%	4%	6%	10%	9%	4%	1247.3
ibm16	70%	4%	6%	10%	6%	4%	1413.4
ibm17	69%	4%	6%	11%	7%	4%	1603.7
ibm18	71%	4%	6%	10%	5%	4%	1658.8
Average	66%	5%	6%	11%	7%	5%	

Table 6. Tradeoff between runtime and quality for benchmarks ibm01 to ibm18.

Bench-mark	Fast 3D Placement				Slow 3D Placement, one post-opt. iteration		Slow 3D Placement, ten post-opt. iteration	
	WL, m	ILV count	OFV	Runtime, sec	OFV % reduction	Slow Down	OFV % reduction	Slow Down
ibm01	2.69	7669	3.46	17	6.0%	3.6	9.4%	90.7
ibm02	6.99	15238	8.51	36	2.9%	3.5	6.8%	73.9
ibm03	6.76	12576	8.02	34	3.6%	3.5	7.3%	69.4
ibm04	8.22	18652	10.08	37	5.8%	3.7	10.1%	65.0
ibm05	17.21	23948	19.61	48	6.8%	3.7	10.7%	42.5
ibm06	10.72	25874	13.31	61	6.1%	3.4	11.1%	78.8
ibm07	15.53	31575	18.69	82	3.9%	3.3	7.8%	63.9
ibm08	16.88	33009	20.18	97	1.7%	3.5	5.6%	79.4
ibm09	14.92	33063	18.23	80	3.5%	3.8	7.3%	75.8
ibm10	27.47	46765	32.15	130	4.2%	3.4	7.7%	59.2
ibm11	21.68	44044	26.08	119	4.6%	3.4	8.7%	67.0
ibm12	35.42	59859	41.41	152	4.0%	3.1	7.8%	49.0
ibm13	27.18	53481	32.53	149	2.2%	3.4	6.3%	69.0
ibm14	57.55	101783	67.73	290	2.1%	3.4	6.1%	57.0
ibm15	66.60	119414	78.54	396	2.7%	3.1	6.2%	64.9
ibm16	84.28	137875	98.06	482	1.2%	2.9	4.9%	58.4
ibm17	120.98	157862	136.77	512	3.0%	3.1	6.6%	46.1
ibm18	90.79	141985	104.98	464	4.5%	3.6	8.1%	63.7
Average					3.8%	3.4	7.7%	65.2

The runtime and quality produced by the fast and slow 3D placement methods are shown in Table 6. For the slow 3D placement method, the results after the first and tenth post-optimization iterations are shown. In this table, WL is the wirelength, ILV is the interlayer via count, OFV is the cost function value, OFV % reduction is the percent reduction of the cost function value from the fast 3D placement results, and slow down represents the ratio of the method's runtime to fast 3D placement's runtime. After the first post-optimization iteration of the slow 3D placement method, a 3.8% improvement was made in the cost function with 3.4 times slower runtimes compared to the fast 3D placement method. After the tenth iteration, a 7.7% improvement was made but 65 times more runtime was consumed.

5.8 Conclusions

Vital to the usefulness of any CAD tool is its ability to satisfy constraints imposed by fabrication. With 3D ICs, this is no exception and limitations on interlayer vias counts make it an important consideration in the design of placement tools. However, it is important to mediate different concerns by not going to extremes. Previous work has either created placements with minimized wirelength neglecting interlayer via density limitations or created placement with minimized interlayer via counts and wirelengths unnecessarily lengthened. Our method fully exploits the tradeoff that exists between wirelength and interlayer via counts and shows that it's possible to achieve any configuration along this tradeoff curve. By providing this flexibility, a designer can pick an interlayer via density based on fabrication constraints and minimize the wirelength at this interlayer via density so that interconnect delays are minimized and performance is maximized. Efficiency is of utmost importance in CAD tool for next generation circuits, as the number of cells in a typical placement is rapidly increasing. Every component in our placement method is designed with efficiency in mind so that it does not become a bottleneck in the design process as the technology node progresses.

6 Thermal Placement and Legalization of 3D ICs

6.1 Introduction

In 3D ICs, thermal problems are particularly prominent because of high power densities and low thermal conductivities. Further technology scaling also exacerbates these high power densities. Previous work in thermal placement has been quite limited, particularly with 3D ICs. Chu and Wong presented a matrix synthesis method for the thermal placement of gate arrays by evenly distributing sources of heat [91]. In [36], Eisenmann and Johannes suggested that their force-directed method could potentially be used for distributing cells based on a heat map. Both of these approaches would lead to a uniform power distribution, but Tsai and Kang pointed out in their paper [28] that a uniform power distribution does not necessarily lead to a uniform temperature distribution. In 3D ICs, this is particularly true because the ideal temperature distribution would concentrate heat in the bottom layer next to the heat sink [12]. Techniques for power reduction using net weighting were presented in [92] and [93]. These methods use the switching activities of the net to determine the net weight, but neglected to represent the thermal environment of the driver cells, where power is being dissipated, in their net weight formulation

Tsai and Kang developed a thermal placement method for both standard cell and macro cell designs [28] using the finite difference method (FDM) for analysis and simulated annealing for optimization. The thermal distribution was improved without sacrificing chip area or wire length, but the runtime complexity for the calculation of thermal resistances was as high as O(n^3) where *n* was the number of nodes in the mesh. In [94], Chen and Sapatnekar presented a partitioning-based thermal placement method that improved upon the run time of the finite difference method presented in [28]. Despite their improvements, the method still appears to run in quadratic time with respect to the number of thermal nodes. With these methods, the calculation of thermal resistances is computationally expensive so simpler methods are needed for their application to thermal placement.

Our previous work [27] in the thermal placement of 3D ICs used FEA to accurately calculated temperatures efficiently and used a force-directed framework to push cells away from areas of high temperature. However, it was determined later that a partitioning-based approach is more appropriate for 3D ICs. Partitioning placement can more efficiently reduce interlayer via counts with its intrinsic min-cut objective and can obtain good placement results even when IO pad connectivity information is missing. In contrasts, the force-directed paradigm relies on an encompassing arrangement of IO pads, which 3D ICs may not have, to produce a well-spread initial placement in order to proceed efficiently and effectively in subsequent iterations. The force-directed placement method from [27] was effective in reducing maximum temperatures, but the average temperatures were only slightly reduced because net weighting for power reduction was not used. It will be shown later in this chapter that total power reduction results in average temperature reduction.

For any thermal placement method to be completely effective, it must also actively reduce power because power has a direct impact on temperatures. If power is disregarded in the thermal placement formulation, any wirelength degradation caused by thermal placement will in turn increase the power and subsequently the temperature. In addition, the cost of interlayer vias must be incorporated into the objective function for thermal placement. In this chapter, a thermal placement method is developed, extending the placement method present in the Chapter 5 to include temperature minimization. Net weights are added to reduce the power selectively during partitioning-based global placement, and additional nets are added to move cells to more favorable thermal conditions. Thermal-aware legalization maintains the improvements made from global placement by using thermal costs in determining cell movements.

6.2 Thermal Objective Function

In Chapter 5, an objective function, Equation (17), was used to take both wirelength and interlayer via counts into consideration during placement, and this allowed the tradeoff between the two to be explored. With thermal placement, the objective function needs to be modified to include thermal considerations. A simple approach would be to add a weighted sum of the cell temperatures to the objective function as shown below:

$$\sum_{each\,net\,i}\left[WL_i + \alpha_{ILV}\cdot ILV_i\right] + \alpha_{TEMP}\sum_{each\,cell\ j}\left[T_j\right] \tag{51}$$

where WL_i is the bounding box wirelength and ILV_i is the number of interlayer vias for net i, T_j is the temperature at the center of cell j, α_{ILV} is the interlayer via coefficient, and α_{TEMP} is the thermal coefficient. The cell temperatures are dependent on both the power dissipation of the cell and the thermal environment around the cell. The power dissipation depends on the capacitance of the net that it drives, and this capacitance depends on the length of the net, capacitance per length, and the fan-out. The thermal environment around the cell depends on the thermal resistance from that position to the heat sink and the temperature contributions from other cells. The thermal resistance in turn depends on the distance to the heat sink, the thermal conductivity of the materials on the way to the heat sink, and the boundary conditions.

However, in practice, temperatures cannot be used directly in the objective function because they are too expensive to recalculate for each individual cell movement, and therefore, simplifications need to be made. It should be noted that the temperature at each cell position is a sum of the temperature contributions from all power signatures in the chip. Since power generation at the cells dominates the total power, each temperature can be expressed as a sum of the temperature contributions from every cell:

$$T_j = \sum_{each\,cell\,\mathrm{k}}\left[\Delta T_j^k\right] \tag{52}$$

where ΔT_j^k is the change in temperature at cell j caused by the power from cell k. If cell k has a nonzero power dissipation, ΔT_j^k increases as the distance between cell j and cell k decreases. Consequently, the temperature contribution from the cell's own power, ΔT_j^j, is typically the dominant term in Equation (52), and this value can be approximated quickly, as will be discussed later in this section. By using ΔT_j^j instead of T_j, the objective function value can be efficiently calculated and used during placement. Therefore, the objective function becomes:

$$\sum_{each\,net\,i}\left[WL_i + \alpha_{ILV}\cdot ILV_i\right] + \alpha_{TEMP}\sum_{each\,cell\ j}\left[\Delta T_j^j\right] \tag{53}$$

ΔT_j^j can be calculated using the simple equation:

$$\Delta T_j^j = R_j^{cell} P_j^{cell} \tag{54}$$

where R_j^{cell} is the thermal resistance between the cell j and ambient, and P_j^{cell} is the power dissipation of cell j. Thus, our objection function becomes:

$$\sum_{each\ net\ i}\left[WL_i + \alpha_{ILV} \cdot ILV_i\right] + \alpha_{TEMP} \sum_{each\ cell\ j}\left[R_j^{cell} P_j^{cell}\right] \tag{55}$$

Thermal resistances were derived using basic equations for heat conduction shown in Equations (56) and (57). Assuming that heat flows in a direct path from the cell to the heat sink, the thermal resistance of the chip at that point, R_{chip}, can be roughly approximated with:

$$R_{chip} = \frac{d}{A_{cs}K} \tag{56}$$

where d is the distance from the cell to the heat sink, A_{cs} is the cross sectional area of the path (assumed to be the same size as the cell), and K is the thermal conductivity of the path. The thermal resistance of the heat sink, R_{hs}, can be roughly approximated with:

$$R_{hs} = \frac{1}{A_{cs}h} \tag{57}$$

where h is convective coefficient of the heat sink. More specifically, the thermal resistances from cell j to ambient in each direction are shown in Figure 40 and can be approximated using Equations (58) - (63). This formulation is similar to that of a single FDM element in [28] with the thermal resistances in each direction considering only heat conduction in that direction.

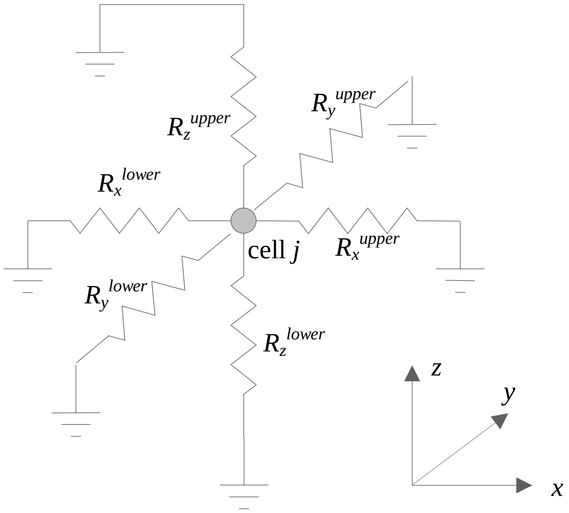

Figure 40. Thermal Resistance Approximation

$$R_x^{lower} = \frac{1}{hd}\left(\frac{x_j^{cell} - E_x^{lower}}{K_x} + \frac{1}{h_x^{lower}} \right) \tag{58}$$

$$R_x^{upper} = \frac{1}{hd}\left(\frac{E_x^{upper} - x_j^{cell}}{K_x} + \frac{1}{h_x^{upper}} \right) \tag{59}$$

$$R_y^{lower} = \frac{1}{wd}\left(\frac{y_j^{cell} - E_y^{lower}}{K_y} + \frac{1}{h_y^{lower}} \right) \tag{60}$$

$$R_y^{upper} = \frac{1}{wd}\left(\frac{E_y^{upper} - y_j^{cell}}{K_y} + \frac{1}{h_y^{upper}} \right) \tag{61}$$

$$R_z^{lower} = \frac{1}{wh}\left(\frac{z_j^{cell} - E_z^{lower}}{K_z} + \frac{1}{h_z^{lower}} \right) \tag{62}$$

$$R_z^{upper} = \frac{1}{wh}\left(\frac{E_z^{upper} - z_j^{cell}}{K_z} + \frac{1}{h_z^{upper}} \right) \tag{63}$$

where w, h, d, and (x_j^{cell},y_j^{cell},z_j^{cell}) are the width, height, depth, and position of the cell respectively. E_x^{lower}, E_x^{upper}, E_y^{lower}, E_y^{upper}, E_z^{lower}, and E_z^{upper} are the positions of the edges of the chip in the x, y, and z directions. h_x^{lower}, h_x^{upper}, h_y^{lower}, h_y^{upper}, h_z^{lower}, and h_z^{upper} are the convective coefficients for the sides of the chip. K_x, K_y, and K_z are the thermal conductivities of the chip. The total thermal resistance from cell j to ambient can be calculated as:

$$R_j^{cell} = \frac{1}{\frac{1}{R_x^{lower}} + \frac{1}{R_x^{upper}} + \frac{1}{R_y^{lower}} + \frac{1}{R_y^{upper}} + \frac{1}{R_z^{lower}} + \frac{1}{R_z^{upper}}} \tag{64}$$

In deep submicron technologies, dynamic power dominants the total power and is primarily dissipated in the cells because driver resistances are usually much larger than interconnect resistances. Therefore, the power dissipation of each cell is dominated by the dynamic power of the nets that it drives. Specifically, the dynamic power associated with net i, P_i^{net}, is divided among the $n_i^{output\ pins}$ cells attached to the net at cell output pins. The sum of these power contributions is the total power dissipation for the cell, P_j^{cell}, and is given by:

$$P_j^{cell} = \sum_{\substack{each\,driven \\ net\; i\; of\;\; cell\;\; j}} \left(\frac{P_i^{net}}{n_i^{output\;\; pins}} \right) \tag{65}$$

where $n_i^{output\ pins}$ is the number of cell output pins attached to net *i* and P_i^{net} is the dynamic power of net *i*. P_i^{net} is a function of the clock frequency, *f*, supply voltage, V_{DD}, switching activity, a_i , and total capacitance, C_i^{total}, of net *i*.

$$P_i^{net} = \tfrac{1}{2}\, f V_{DD}{}^2 a_i C_i^{total} \tag{66}$$

The total capacitance of net *i* can be calculated as;

$$C_i^{total} = C_{per\; wl} WL_i + C_{per\; ilv} ILV_i + C_{per\;\; pin} n_i^{input\;\; pins} \tag{67}$$

where $C_{per\ wl}$ is the capacitance per wirelength, $C_{per\ ilv}$ is the capacitance per interlayer via, $C_{per\ pin}$ is the input pin capacitance, and $n_i^{input\ pins}$ is the number of cell input pins attached to net *i*. Therefore, Equation (65) can be rewritten as

$$P_j^{cell} = \sum_{\substack{each\,driven \\ net\; i\; of\;\; cell\;\; j}} \left[\frac{\tfrac{1}{2}\, f V_{DD}{}^2 a_i}{n_i^{output\;\; pins}} \left(C_{per\; wl} WL_i + C_{per\; ilv} ILV_i + C_{per\;\; pin} n_i^{input\;\; pins} \right) \right] \tag{68}$$

Using Equations (64) and (68), ΔT_j^j can be calculated in constant time and used for efficient calculations of the objective function during placement.

6.3 Global Placement

Using the objective function in Equation (55), two new mechanisms were added to the partitioning placement method presented in Section 5.2 for global placement, namely, the use of net weights and additional nets. The net weighting scheme takes both the thermal environment of the driver cells and the potential power usage of the net into consideration. Different net weights are created for the lateral (*x* and *y*) and vertical (*z*) directions to take into account the interlayer via coefficient and differences in capacitance per unit length for different directions. This causes nets to shrink and power usage to be decreased when they have high power per length or high thermal resistances at their driver cells. Additional nets are created to move cells toward areas of lower thermal resistance based on their power dissipation. This provides an incentive for high powered cells to move closer to the heat sink in order to reduce temperatures.

6.3.1 Thermal-Aware Net Weighting

The thermal net weighting scheme takes into consideration the thermal resistance at the driver cells, the switching activity of the net, and the capacitances per length. Recall that in Equation (68), power is a function of wirelength, interlayer via count, and pin capacitance. All of the terms in Equation (68) depend on either lateral or vertical net length except $C_{per\ pin}\ n_i^{input\ pins}$ which will be removed from consideration at this time. A term must be dependent on net length in order to be useful for net weighting. By dropping the pin capacitance term and applying Equation (68) to Equations (55), we obtain the following objective function for deriving the net weights:

$$\sum_{each\ net\ i}\left[WL_i + \alpha_{ILV}\cdot ILV_i\right] + \alpha_{TEMP}\sum_{each\ cell\ j}\left[R_j^{cell}\sum_{\substack{each\ driven\\ net\ i\ of\ cell\ j}}\left(s_i^{wl}WL_i + s_i^{ilv}ILV_i\right)\right] \tag{69}$$

where $s_i^{wl} = \dfrac{\frac{1}{2}fV_{DD}{}^2 a_i C_{per\ wl}}{n_i^{output\ pins}}$ (70)

and $s_i^{ilv} = \dfrac{\frac{1}{2}fV_{DD}{}^2 a_i C_{per\ ilv}}{n_i^{output\ pins}}$. (71)

If we change the order of summation of the second term in Equation (69), we get

$$\sum_{each\ cell\ j}\left[R_j^{cell}\sum_{\substack{each\ driven\\ net\ i\ of\ cell\ j}}\left(s_i^{wl}WL_i + s_i^{ilv}ILV_i\right)\right] = \sum_{each\ net\ i}\left[\left(s_i^{wl}WL_i + s_i^{ilv}ILV_i\right)\sum_{\substack{each\ driver\\ cell\ j\ of\ net\ i}}\left(R_j^{cell}\right)\right] \tag{72}$$

If we applying Equation (72) to Equation (69), we obtain

$$\sum_{each\ net\ i}\left[WL_i + \alpha_{ILV}\cdot ILV_i\right] + \alpha_{TEMP}\sum_{each\ net\ i}\left[\left(s_i^{wl}WL_i + s_i^{ilv}ILV_i\right)\sum_{\substack{each\ driver\\ cell\ j\ of\ net\ i}}\left(R_j^{cell}\right)\right] \tag{73}$$

and finally,

$$\sum_{each\ net\ i}\left[\left(1+\alpha_{TEMP}R_i^{net}s_i^{wl}\right)WL_i + \alpha_{ILV}\left(1+\frac{\alpha_{TEMP}R_i^{net}s_i^{ilv}}{\alpha_{ILV}}\right)ILV_i\right] \tag{74}$$

where $R_i^{net} = \sum_{\substack{each\ driver\\ cell\ j\ of\ net\ i}}\left(R_j^{cell}\right)$.

From this we obtain the following net weights for net j:

$$nw_i^{lateral} = 1 + \alpha_{TEMP} R_i^{net} s_i^{wl} \tag{75}$$

$$nw_i^{vertical} = 1 + \frac{\alpha_{TEMP} R_i^{net} s_i^{ilv}}{\alpha_{ILV}} \tag{76}$$

where $nw_i^{lateral}$ is the net weight used in the x and y directions, and $nw_i^{vertical}$ is the net weight used in the z direction.

6.3.2 Thermal Resistance Reduction Nets

Better thermal results can be obtained when higher powered cells are placed in areas with lower thermal resistances to the ambient. During placement, this can be encouraged by adding nets that pull each cell toward the heat sink and weighted based on the power dissipated at the cell. With our method, these nets are called thermal resistance reduction nets and are weighted according to the power usage of the cell and the slope of the thermal resistance profile of the chip. As the thermal resistance slope increases, high powered cells are more strongly attracted to the heat sink where temperatures and thermal resistances would be lower. Recall that the thermal component of our objective function, Equations (55), is

$$\alpha_{TEMP} \sum_{each\,cell\ j} \left[\Delta T_j\right] = \sum_{each\,cell\ j} \left[\alpha_{TEMP} P_j^{cell} R_j^{cell}\right] \tag{77}$$

Because the distances are much shorter and heat sinking is primarily in the z direction, the thermal resistance increases principally with vertical distance away the heat sink. As such, the thermal resistance from cell j to the ambient, R_j^{cell}, can be approximated with $R_j^{cell} \approx R_0^z + R_{slope}^z d_j^z$ where R_0^z is the thermal resistance at the bottom of the chip, R^z_{slope} is the slope the thermal resistance in the z direction, and d_j^z is the distance of the cell from the bottom of the chip. Because R_0^z is constant with respect to d_j^z, it can be dropped from the objective function, and the thermal component, Equation (77), of the objective function becomes

$$\sum_{each\,cell\ j} \left[\alpha_{TEMP} P_j^{cell} R_{slope}^z d_j^z\right] \tag{78}$$

where $P_j^{cell} = \sum_{\substack{each\,driven \\ net\ i\ of\ cell\ j}} \left[s_i^{wl} WL_i + s_i^{ilv} ILV_i + s_i^{input\ pins}\right]$ (79)

and $s_i^{input\ pins} = \dfrac{\frac{1}{2} f V_{DD}{}^2 a_i C_{per\ pin} n_i^{input\ pins}}{n_i^{output\ pins}}$. (80)

The additional nets that are added to the netlist for thermal resistance reduction have net weights of

$$nw_j^{cell} = \alpha_{TEMP} P_j^{cell} R_{slope}^z \tag{81}$$

where nw_j^{cell} is the net weight of cell j to the heat sink.

In Equation (79), P_j^{cell} depends on wirelength and interlayer via counts of its driven nets so it is updated after cells in these nets are moved. However at the beginning of global placement, cells are placed at the center of the chip and consequently the wirelengths and interlayer via counts are zero. This would cause the power contribution from the wirelength and interlayer via counts to be neglected in determining nw_j^{cell}. Some minimum value for the wirelength and interlayer via counts should be used instead, and these values can be determined by minimizing the objective function for each net in question. The derivation of these minimum values is similar to the PEKO (Placement Example with Known Optimal wirelength) formulation presented in [95], but is extended to 3D ICs. PEKO benchmarks were created to have known optimal wirelengths for 2D ICs. For a single net, the objective function, neglecting thermal considerations, can be written as

$$F_i = WL_i + \alpha_{ILV} ILV = WL_i^x + WL_i^y + \alpha_{ILV} ILV_i \tag{82}$$

where WL_i^x and WL_i^y are the wirelengths in the x and y directions respectively. When the net is optimal, disregarding all other nets, its cells are closely abutted within a rectangular prism as shown in Figure 41, and the following approximation can be made.

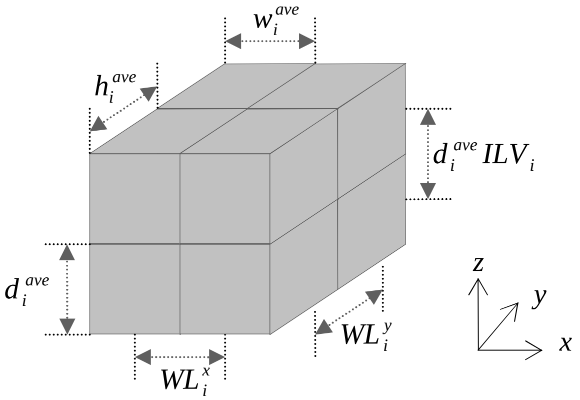

Figure 41. Optimal arrangement of cells for net *i*.

$$n_i^{total\ pins} \approx \left(\frac{WL_i^x}{w_i^{ave}}+1\right)\left(\frac{WL_i^y}{h_i^{ave}}+1\right)\left(ILV_i+1\right) \qquad (83)$$

where $n_i^{total\ pins}$ is the total number of cells in net i, w_i^{ave} is the average width of the cells in net i, and h_i^{ave} is the average height of the cells in net i. Combining Equations (82) and (83) yields the following equation when F_i is minimized.

$$F_i^{\min} \approx WL_i^x + WL_i^y + \alpha_{ILV}\left(\frac{n_i^{total\ pins}}{\left(\frac{WL_i^x}{w_i^{ave}}+1\right)\left(\frac{WL_i^y}{h_i^{ave}}+1\right)}-1\right) \qquad (84)$$

The values of WL_i^x and WL_i^y at the minimum of F_i, F_i^{min}, can be found by setting the partial derivatives of F_i with respect to WL_i^x and WL_i^y to zero:

$$1-\frac{\alpha_{ILV}\,n_i^{total\ pins}}{w_i^{ave}\left(\frac{WL_i^x}{w_i^{ave}}+1\right)^2\left(\frac{WL_i^y}{h_i^{ave}}+1\right)}=0 \qquad (85)$$

$$1-\frac{\alpha_{ILV}\,n_i^{total\ pins}}{h_i^{ave}\left(\frac{WL_i^x}{w_i^{ave}}+1\right)\left(\frac{WL_i^y}{h_i^{ave}}+1\right)^2}=0 \qquad (86)$$

Solving for WL_i^x, WL_i^y, and ILV_i using Equations (83), (85), and (86) yields

$$WL_i^{x\,opt} = \sqrt[3]{\alpha_{ILV}\,w_i^{ave}h_i^{ave}n_i^{total\ pins}} - w_i^{ave} \qquad (87)$$

$$WL_i^{y\,opt} = \sqrt[3]{\alpha_{ILV}\,w_i^{ave}h_i^{ave}n_i^{total\ pins}} - h_i^{ave} \qquad (88)$$

$$ILV_i^{opt} = \frac{\sqrt[3]{\alpha_{ILV}\,w_i^{ave}h_i^{ave}n_i^{total\ pins}}}{\alpha_{ILV}} - 1 \qquad (89)$$

where $WL_i^{x\ opt}$, $WL_i^{y\ opt}$, and ILV_i^{opt} are the values obtained when F_i is minimize. In calculating P_j^{cell} for nw_j^{cell}, if WL_i^x, WL_i^y, or ILV_i fall below their optimal values, $WL_i^{x\ opt}$, $WL_i^{y\ opt}$, or ILV_i^{opt}, their optimal value is used instead. P_j^{cell} and nw_j^{cell} are updated after the cells that affect WL_i^x, WL_i^y, and ILV_i are moved.

6.3.3 Implementation

The global placement algorithm used for thermal placement is based on the partitioning placement method presented in Section 5.2 and proceeds as shown in Figure 42. The algorithm uses a queue to recursively partition the placement in a breadth first fashion, and the net weights are updated after all regions (and cells) at each depth have been processed. The algorithm begins by adding thermal resistance reduction nets for each cell to the netlist and adding the entire netlist to the empty queue. The main loop continues until the queue is empty and begins by taking the first region off the front of the queue. If the depth of the recursive bisection for the region is greater than that of the last partition on the queue, then the net weights are updated. If the region has more than one cell, it is partitioned and the two halves are added to the end of the queue.

```
THERMAL_PARTITIONING_PLACEMENT(NETLIST,$\alpha_{ILV}$, $\alpha_{TEMP}$) {
     REGION_QUEUE=EMPTY
     LAST_LEVEL=0
     FOR EACH CELL IN NETLIST {
          ADD THERMAL RESISTANCE REDUCTION NET TO NETLIST
     }
     NEXT_REGION=NETLIST
     NEXT_REGION.LEVEL=1
     ENQUEUE NEXT_REGION INTO REGION_QUEUE
     WHILE(REGION_QUEUE NOT EMPTY) {
          DEQUEUE NEXT_REGION OFF REGION_QUEUE
          if(NEXT_REGION.LEVEL > LAST_LEVEL) {
               UPDATE $nw_i^{lateral}$'s, $nw_i^{vertical}$'s, and $nw_j^{cell}$'s
          }
          LAST_LEVEL=NEXT_REGION.LEVEL
          if(NEXT_REGION HAS MORE THAN ONE CELL) {
               PARTITION NEXT_REGION INTO
                    BOTTOM_REGION AND TOP_REGION
               BOTTOM_REGION.LEVEL=LAST_LEVEL+1
               TOP_REGION.LEVEL=LAST_LEVEL+1
               ENQUEUE BOTTOM_REGION INTO REGION_QUEUE
               ENQUEUE TOP_REGION INTO REGION_QUEUE
          }
     }
}
```

Figure 42. Thermal Partitioning Placement

6.4 Legalization

Thermal legalization utilizes the same coarse and final legalization methods presented in Sections 5.3 and 5.4, but in determining the cost of cell movements, the objective function from Equation (55) was used instead of Equation (17). The pseudocode of the legalization procedure is shown in Figure 43. The procedure begins by finding the optimal region for each cell as discuss in Section 5.3.3.1 and performing the best move or swap to that region that results in a reduction in the cost function. Next, moves and swaps to adjacent bins are considered for each cell. Cell shifting is performed until the current maximum bin density, d_{max}, is less than some specified maximum bin density, D_{max}, that is close to 100%. Final legalization finishes up by completely removing any residual overlaps.

```
THERMAL_LEGALIZATION (PLACEMENT,$\alpha_{ILV}$, $\alpha_{TEMP}$) {
     OPTIMAL REGION SWAPS AND MOVES USING THERMAL OBJECTIVE
     LOCAL SWAPS AND MOVES USING THERMAL OBJECTIVE
     WHILE($d_{max}$>$D_{max}$) {
          CELL_SHIFTING
     }
     FINAL LEGALIZATION USING THERMAL OBJECTIVE
}
```

Figure 43. Thermal-Aware Legalization

6.5 Implementation

The entire thermal placement method is shown in Figure 44 and begins by extracting the netlist and placing the cells at the center of the chip. Partitioning-based placement with thermal considerations is performed, followed by thermal legalization. Post-optimization procedures from Section 5.5 were not used here, because they were found to produce no improvements with thermal placement.

```
3D_THERMAL_PLACEMENT(NETLIST,$\alpha_{ILV}$, $\alpha_{TEMP}$){
     EXTRACT NETLIST
     INTIAL_PLACEMENT
     THERMAL_PARTITIONING_PLACEMENT
     THERMAL_LEGALIZATION
}
```

Figure 44. 3D Thermal Placement Method

6.6 Results

The thermal placement method was implemented in C++ and run on a Linux workstation with a Pentium 4 3.2GHz CPU and 2GB memory. The benchmark circuits from the IBM-PLACE suite [86] were used in these experiments as shown in Table 7. The circuits were scaled to 100nm, given 5% white space, and placed into four layers with the inter-row distances set to a quarter of the row heights. The chip areas in Table 7 are the footprint areas of the circuits under these conditions. The vertical dimensions for the chips were based on the design specifications of MIT Lincoln Labs' 0.18μm 3D FD-SOI technology [24] [89] [90] with the bulk substrate given a thickness of 500μm, layers given thicknesses of 5.7μm, and interlayers given thicknesses of 0.7μm.

Table 7. Benchmark Circuits

name	cells	nets	iopads	chip area (m^2)
ibm01	12282	13056	246	6.03E-08
ibm02	19321	19291	259	8.60E-08
ibm03	22207	26104	283	9.01E-08
ibm04	26633	31328	287	1.22E-07
ibm05	29347	29647	1201	1.50E-07
ibm06	32185	34935	166	1.17E-07
ibm07	45135	46885	287	1.97E-07
ibm08	50977	49228	286	2.14E-07
ibm09	51746	59454	285	2.21E-07
ibm10	67692	72760	744	3.77E-07
ibm11	68525	78843	406	2.87E-07
ibm12	69663	75157	637	4.15E-07
ibm13	81508	97574	490	3.26E-07
ibm14	146009	150262	517	6.80E-07
ibm15	158244	183684	383	6.34E-07
ibm16	182137	188324	504	8.92E-07
ibm17	183102	186764	743	1.04E-06
ibm18	210323	201560	272	9.88E-07

From these design specifications, effective thermal conductivities of the chip were determined using the thicknesses and thermal conductivities of its constituent integration materials. The thermal conductivities of the vias and interconnects are shown in Table 8 with the vertical arrangement of their composing materials. The total effective thermal conductivity of each was determined by considering the materials as being in series. For example, the total thermal conductivity of the via plugs is (700+75+30)/(700/170+75/20 +30/22) = 87.2.

The thickness and composition of each integration level is shown in Table 9. Only the integration levels of the bottom layer up to the first "interlayer" are shown. Levels "SOI active" through "overglass" are repeated for each additional layer with an "interlayer" level between layers. The effective thermal conductivity of each level was calculated by using the weighted average of the thermal conductivities for that layer. For example, the effective thermal conductivity of the interlayer is a weighted average of 5% "via plug" and 95% "adhesive": 0.5×87.2+0.95×1.1=5.405. Integration levels are arranged verticals so the effective thermal conductivity of the chip was calculated by considering the integration level as being in series as with Table 8. From this, the effective thermal conductivity was found to be 10.2 W/mK for the entire chip. This value is used in following experiments for calculating temperatures and the thermal resistances.

Table 8. Thermal Conductivity of Vias and Interconnects

Via Plugs		
Thickness, nm	Material	Thermal conductivity, W/mK
700	W	170
75	TiN	20
30	Ti	22
Total		87.2

Metal levels		
Thickness, nm	Material	Thermal conductivity, W/mK
50	TiN	20
40	Ti	22
500	AlSi	200
40	Ti	22
Total		72.95

Table 9. Thermal Conductivity of Individual Levels

Level	Thickness, nm	Material Type	Thermal Conductivity, W/mK	Material Type Percent	Effective Thermal Conductivity, W/mK
interlayer	700	via plug	87.2	5%	5.405
		adhesive	1.1	95%	
overglass	750	via plug	87.2	5%	5.405
		SiO2	1.1	95%	
metal 3	630	via plug	87.2	10%	38.449
		metal wire	72.9	40%	
		SiO2	1.1	50%	
ILD	1000	via plug	87.2	10%	9.710
		SiO2	1.1	90%	
metal 2	630	via plug	87.2	10%	38.449
		metal wire	72.9	40%	
		SiO2	1.1	50%	
ILD	1000	via plug	87.2	10%	9.710
		SiO2	1.1	90%	
metal 1	630	via plug	87.2	10%	38.449
		metal wire	72.9	40%	
		SiO2	1.1	50%	
ILD	600	via plug	87.2	10%	9.710
		SiO2	1.1	90%	
polysilicon	200	via plug	87.2	10%	11.795
		PolySi	15.0	15%	
		SiO2	1.1	75%	
SOI active	40	via plug	87.2	5%	146.860
		Si	150.0	95%	
BOX	190	via plug	87.2	5%	5.405
		SiO2	1.1	95%	
Entire Chip	24780				10.237

Temperatures are calculated with FEA (Chapter 2) using meshes that have element widths and heights approximately equal to the average cell width, and element depths approximately equal to the layer thickness. FEA elements are given this cell-level granularity because the internal structures of the cells are not provided and the specific interconnect structures between cells on the chip cannot be obtained until after routing.

Recall from Section 6.2 that the dynamic power calculation used by this method depends on the capacitance of the net, which consists of three components: wirelength capacitance, interlayer via capacitance, and input pin capacitance. Capacitance values were derived from [96] for the 100nm technology. Interconnect capacitances in the *x* and *y* directions are assumed to be 73.8pF/m, interlayer via capacitances are assumed to be 1480pF/m, and input pin capacitances are assumed to be 0.350fF. In these experiments, the bottom of the chip is connected to the heat sink and given convective boundary conditions with a coefficient of 10^6 W/m^{2}K. The other sides of the chip are modeled as insulated to simulate the low heat sinking properties of the packaging on these surfaces. If desired, a more sophisticated thermal model for the heat sink and packaging could be used instead. The ambient temperature was set to 0^oC for convenience, but this is only a reference value. The temperatures can trivially be translated by the amount of any other ambient as desired.

6.6.1 Interlayer Via Coefficient Versus Thermal Coefficient

The interlayer via coefficient, α_{ILV}, and thermal coefficient, α_{TEMP}, are the two main controlling parameters in our thermal placement method for 3D ICs. These parameters were presented in Section 6.2 and are used by the objective function, Equation (55), to control the tradeoff between temperature, wirelength, and interlayer via counts. As shown in Chapter 5, the interlayer via coefficient reduces the interlayer via counts as it is increased. As will be shown, the thermal coefficient reduces the temperatures as it is increased. In this section, the effect of these parameters on temperature, power, wirelength, and interlayer via counts will be explored for the ibm01 benchmark as the interlayer via coefficient is varied from 5×10^{-8} to 1.6×10^{-3} and the thermal coefficient is varied from 1×10^{-8} to 1.3×10^{-3}. These ranges of values were experimentally determined to span the entire range of interlayer via and temperature reduction and are centered around the value of average cell width and height ($\sim10^{-5}$). The parameters are incremented within their range by multiplying the previous value by two to get the next value in the list.

The effect of the interlayer via and thermal coefficients on the average temperature, total power, maximum temperature, wirelength, and interlayer via counts are respectively tabulated in Tables 10 through 14 and graphically shown in Figures 45 through 49. As the interlayer via coefficient and thermal coefficient are reduced to their minimum values, the interlayer via counts, temperatures, and power increases to their maximum values, whereas wirelengths are minimized. In general, as the interlayer via coefficient is decreased, the temperatures and power increase because of the interlayer via counts. With the capacitance per interlayer via being relatively large, interlayer vias produce a dominate portion of the total power, even more so when interlayer via counts are high. It should be noted that as layer thicknesses and interlayer via sizes decrease relative to other features, this is not the case. When the interlayer via coefficient is low, interlayer via counts are at their maximum and produce a large portion of the total power so as the thermal coefficient is increased in this case, interlayer via counts are actually decreased in order to reduce power and temperature. When the interlayer via coefficients are higher, increasing thermal coefficients reduce the temperature and power at the expense of both wirelength and interlayer via counts. For any interlayer via coefficient, increasing the thermal coefficient causes the wirelength to increase. Average temperature and power closely follow each other and decrease steadily with increasing thermal coefficients, but maximum temperature decreases less smoothly because the objective function does not directly act upon it.

Table 15 shows the percent change in the temperature, power, wirelength, and interlayer via counts as the thermal coefficient is increased from 1×10^{-8} to 4.1×10^{-5}. On average, the temperatures are reduced by about 30%, power reduced by 21%, and wirelength increased by 22%. The percent change in the number of interlayer vias depend on the interlayer via coefficient and vary from a reduction of 14% when the interlayer via coefficient is low to an increase of about 30% when the interlayer via coefficient is higher. In Figure 50, the grid of interlayer via coefficients and thermal coefficients is shown on a plot of interlayer-via counts versus wirelength. As the thermal coefficient is increased and temperatures are reduced, the tradeoff curves are degraded and moved to the right toward high wirelengths and interlayer via counts.

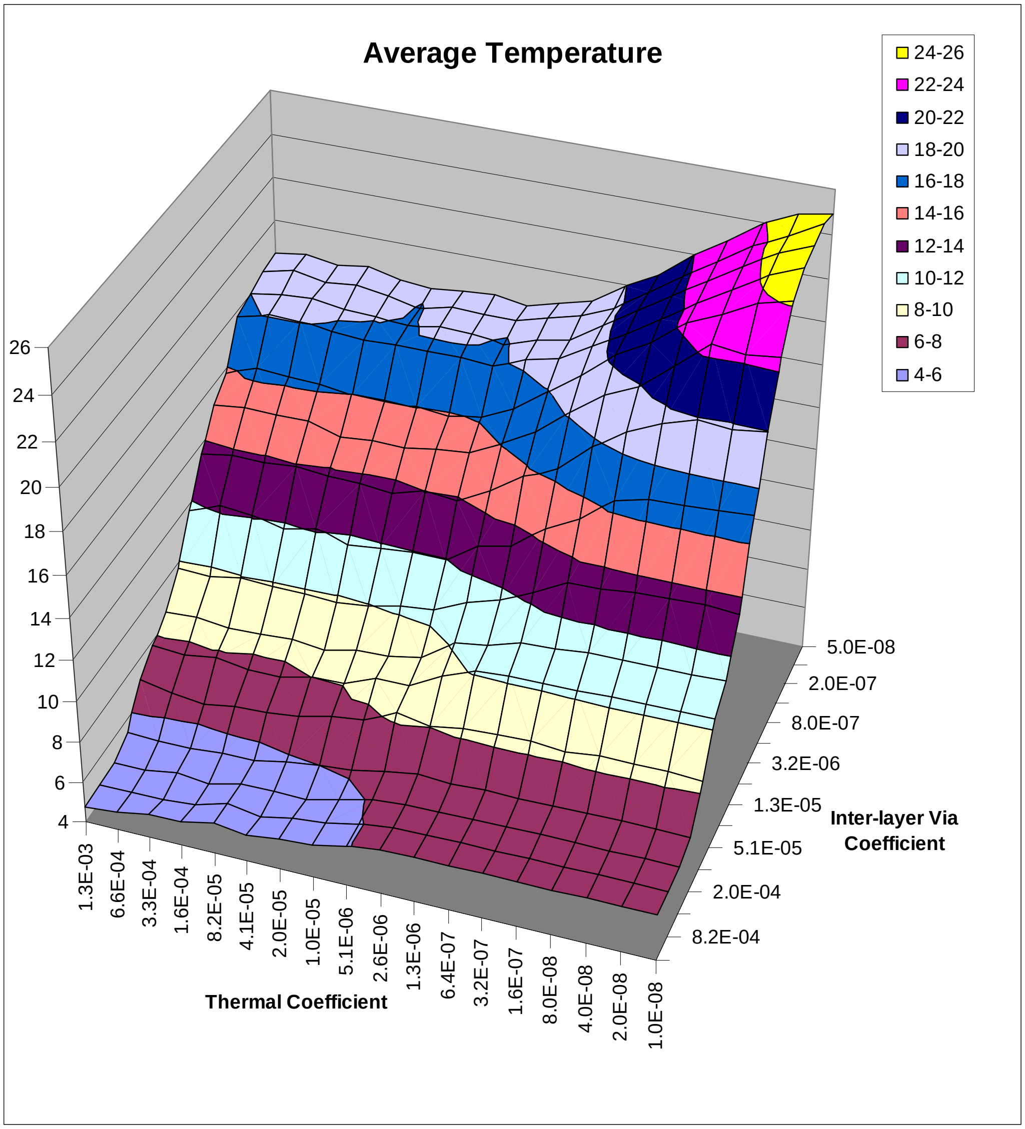

Figure 45. The effect of thermal placement on the average temperature of ibm01 as the thermal and interlayer via coefficients are varied.

Table 10. The effect of thermal placement on the average temperature of ibm01.

		Interlayer Via Coefficient						
		1.0E-07	4.0E-07	1.6E-06	6.4E-06	2.6E-05	1.0E-04	4.1E-04
Thermal Coefficient	1.0E-08	25.2	24.8	22.5	17.1	11.2	8.4	6.3
	4.0E-08	23.7	23.3	22.5	16.9	11.1	8.5	6.3
	1.6E-07	21.8	21.4	19.9	16.4	11.1	8.3	6.4
	6.4E-07	19.7	19.6	18.1	14.5	10.8	8.4	6.3
	2.6E-06	18.7	18.3	16.1	12.9	10.1	8.1	6.3
	1.0E-05	18.3	17.8	15.1	12.0	8.8	7.6	5.7
	4.1E-05	18.0	17.8	14.9	11.7	8.4	7.0	5.4
	1.6E-04	18.5	17.9	15.3	11.8	8.4	6.8	5.0
	6.6E-04	18.6	18.1	15.3	12.3	8.6	6.7	4.9

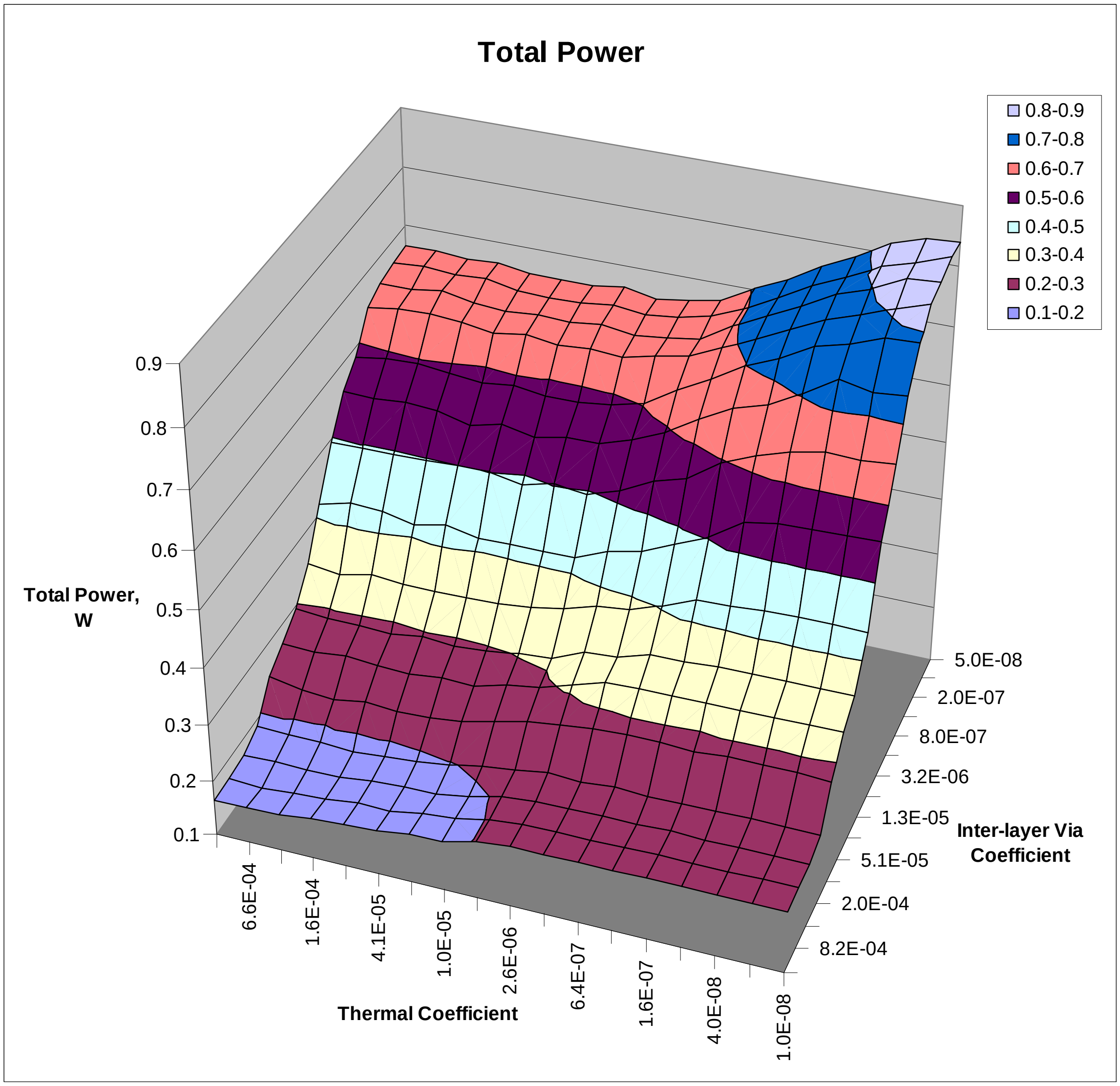

Figure 46. The effect of thermal placement on the total power of ibm01 as the thermal and interlayer via coefficients are varied.

Table 11. The effect of thermal placement on the total power of ibm01.

		Interlayer Via Coefficient						
		1.0E-07	4.0E-07	1.6E-06	6.4E-06	2.6E-05	1.0E-04	4.1E-04
Thermal Coefficient	1.0E-08	0.845	0.823	0.733	0.555	0.359	0.281	0.213
	4.0E-08	0.808	0.786	0.739	0.550	0.358	0.284	0.213
	1.6E-07	0.755	0.741	0.675	0.540	0.358	0.279	0.214
	6.4E-07	0.695	0.688	0.629	0.478	0.352	0.283	0.212
	2.6E-06	0.661	0.652	0.569	0.442	0.331	0.273	0.211
	1.0E-05	0.660	0.644	0.551	0.417	0.302	0.252	0.192
	4.1E-05	0.651	0.648	0.551	0.417	0.290	0.233	0.181
	1.6E-04	0.664	0.647	0.553	0.416	0.287	0.235	0.170
	6.6E-04	0.667	0.650	0.552	0.431	0.290	0.231	0.172

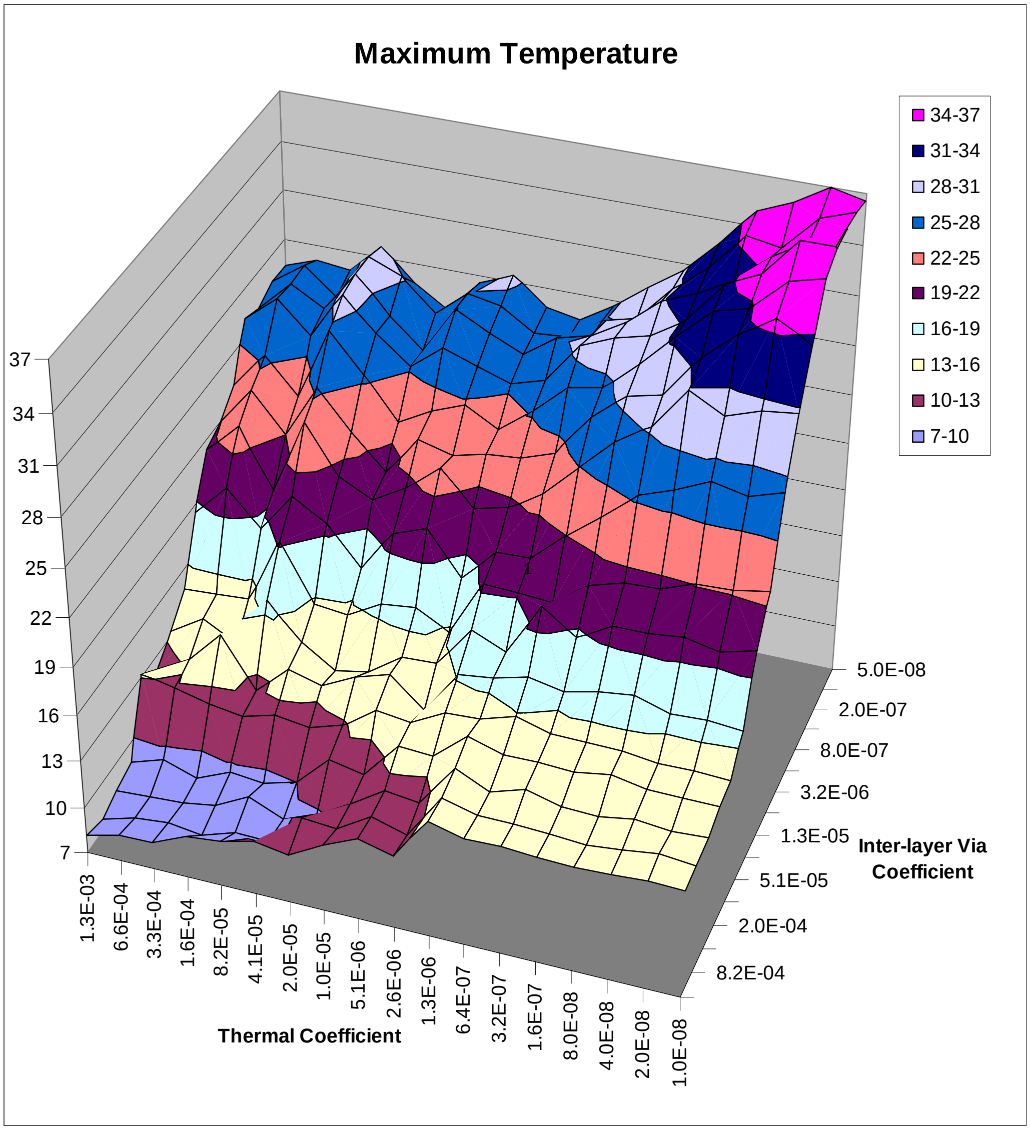

Figure 47. The effect of thermal placement on the maximum temperature of ibm01 as the thermal and interlayer via coefficients are varied.

Table 12. The effect of thermal placement on the maximum temperature of ibm01.

		Interlayer Via Coefficient						
		1.0E-07	4.0E-07	1.6E-06	6.4E-06	2.6E-05	1.0E-04	4.1E-04
Thermal Coefficient	1.0E-08	37.0	37.1	33.6	27.3	20.2	14.9	14.2
	4.0E-08	34.2	34.8	33.4	27.0	20.1	15.0	14.3
	1.6E-07	32.2	31.3	29.0	25.3	19.9	15.2	14.1
	6.4E-07	28.4	29.4	26.8	22.0	20.1	15.9	13.9
	2.6E-06	26.5	26.0	23.5	20.1	17.7	15.1	12.1
	1.0E-05	27.6	26.0	22.8	20.6	15.0	14.2	11.0
	4.1E-05	26.3	25.2	20.9	17.2	13.2	12.2	9.0
	1.6E-04	29.1	28.8	25.3	20.9	16.7	11.8	8.8
	6.6E-04	27.8	27.9	24.3	20.5	14.6	12.7	8.4

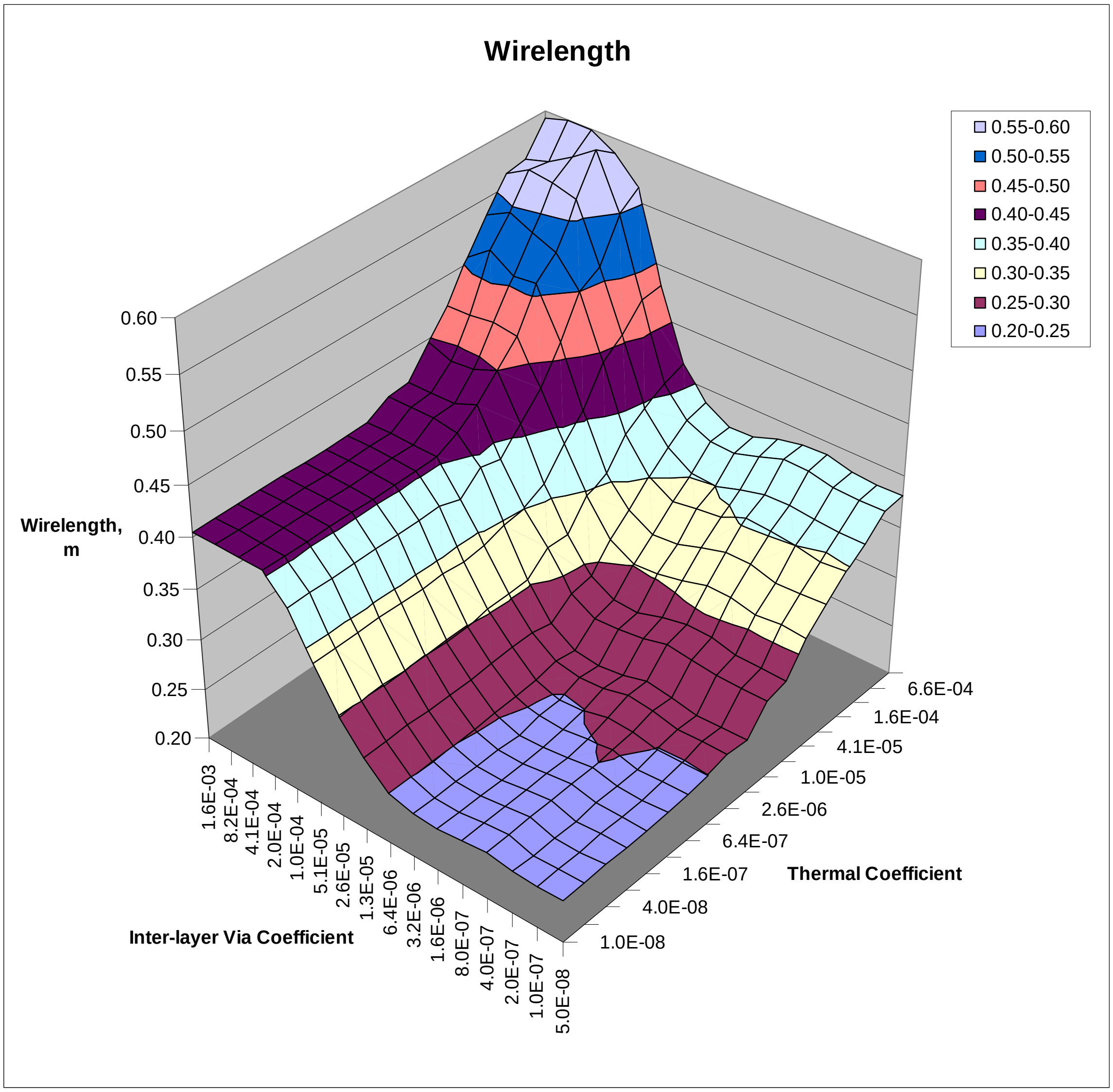

Figure 48. The effect of thermal placement on the wirelength of ibm01 as the thermal and interlayer via coefficients are varied.

Table 13. The effect of thermal placement on the wirelength of ibm01.

		Interlayer Via Coefficient						
		1.0E-07	4.0E-07	1.6E-06	6.4E-06	2.6E-05	1.0E-04	4.1E-04
Thermal Coefficient	1.0E-08	0.241	0.247	0.242	0.250	0.301	0.380	0.404
	4.0E-08	0.241	0.249	0.242	0.250	0.301	0.380	0.405
	1.6E-07	0.243	0.249	0.243	0.251	0.300	0.380	0.405
	6.4E-07	0.246	0.252	0.246	0.254	0.299	0.381	0.405
	2.6E-06	0.255	0.259	0.257	0.260	0.303	0.374	0.406
	1.0E-05	0.278	0.282	0.278	0.274	0.322	0.387	0.420
	4.1E-05	0.311	0.314	0.308	0.301	0.346	0.431	0.475
	1.6E-04	0.346	0.353	0.344	0.337	0.381	0.500	0.536
	6.6E-04	0.378	0.385	0.377	0.371	0.407	0.549	0.585

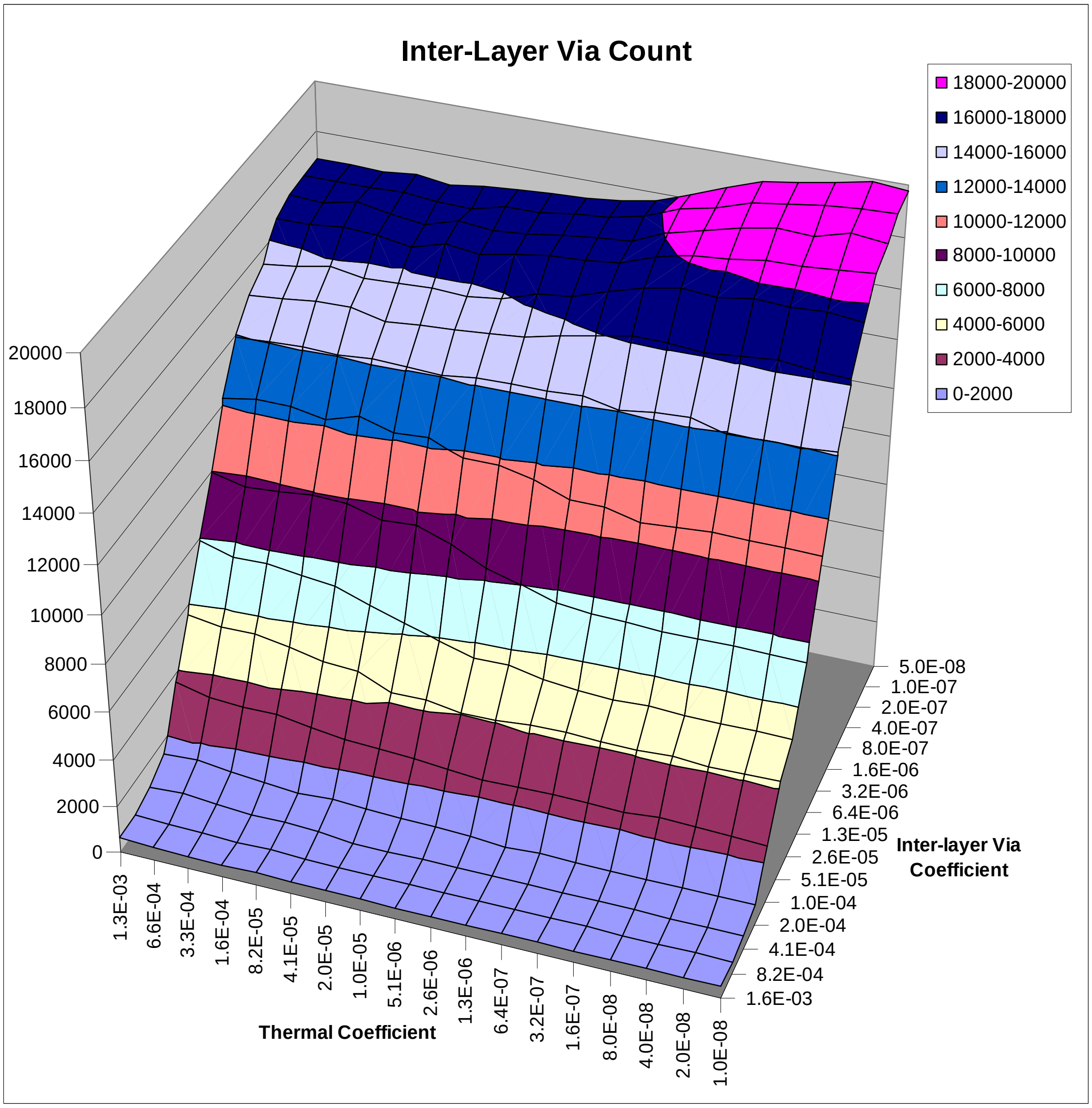

Figure 49. The effect of thermal placement on the interlayer via count of ibm01 as the thermal and interlayer via coefficients are varied.

Table 14. The effect of thermal placement on the interlayer via count of ibm01.

		Interlayer Via Coefficient						
		1.0E-07	4.0E-07	1.6E-06	6.4E-06	2.6E-05	1.0E-04	4.1E-04
Thermal Coefficient	1.0E-08	19594	18692	16145	10801	5036	2447	588
	4.0E-08	19399	18476	16370	10800	5032	2527	582
	1.6E-07	19003	18229	16100	10724	5114	2423	590
	6.4E-07	18268	17815	16184	10960	5131	2521	582
	2.6E-06	17449	17163	15646	11510	5526	2457	575
	1.0E-05	17150	16878	15218	11761	5906	2680	569
	4.1E-05	16795	16793	15019	12212	6765	2841	704
	1.6E-04	16961	16644	15189	12192	7389	3274	736
	6.6E-04	16909	16578	15016	12467	7528	3317	1009

Table 15. The percent change in the average temperature, maximum temperature, total power, wirelength, and interlayer via counts for ibm01 from a thermal coefficient of 1×10^{-8} to a value of 4.1×10^{-5} for ibm01.

		Average Temperature	Maximum Temperature	Total Power, W	Wirelength, m	Interlayer Via Count
Interlayer Via Coefficient	5.0E-08	-27%	-30%	-22%	29%	-14%
	1.0E-07	-29%	-29%	-23%	29%	-14%
	2.0E-07	-28%	-33%	-21%	28%	-11%
	4.0E-07	-28%	-32%	-21%	28%	-10%
	8.0E-07	-34%	-37%	-26%	29%	-10%
	1.6E-06	-33%	-38%	-25%	27%	-7%
	3.2E-06	-31%	-34%	-23%	25%	0%
	6.4E-06	-32%	-37%	-25%	21%	13%
	1.3E-05	-27%	-31%	-20%	15%	29%
	2.6E-05	-25%	-35%	-19%	15%	34%
	5.1E-05	-23%	-19%	-19%	14%	15%
	1.0E-04	-16%	-18%	-17%	13%	16%
	2.0E-04	-14%	-32%	-16%	17%	30%
	4.1E-04	-15%	-36%	-15%	17%	20%
	Average	-26%	-31%	-21%	22%	6%

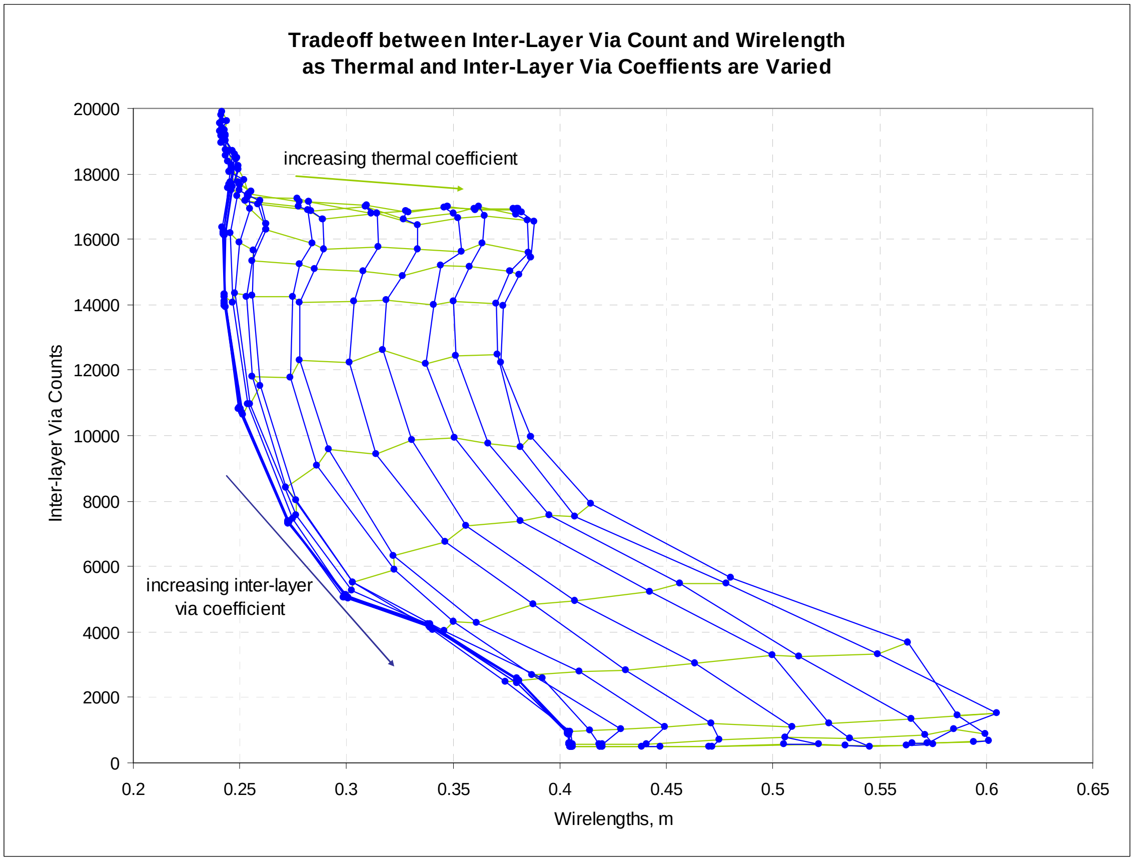

Figure 50. Tradeoff curves between interlayer via counts and wirelength as the thermal and interlayer via coefficients are varied.

6.6.2 The Effect of Number of Layers on Thermal Placement

In this set of experiments, the effect of the thermal coefficient on temperature, power, interlayer via count, and wirelength was examined as the number of layers increases. It is also shown that the thermal placement method is effective not only with 3D ICs but also with 2D ICs. In these experiments, the ibm01 benchmark was used with the number of layers increased from one to eight and the interlayer via coefficient set to 1×10^{-5}. In Figure 51, the change in the interlayer via counts and wirelength is shown as the thermal coefficient is increased for 3D ICs with 1, 2, 4, 6, and 8 layers. The percent increase in wirelength and interlayer via counts is shown in Figure 52 and Figure 53, and the reduction in average temperature and total power is shown Figure 54 and Figure 55 as the thermal coefficient increases with different numbers of layers.

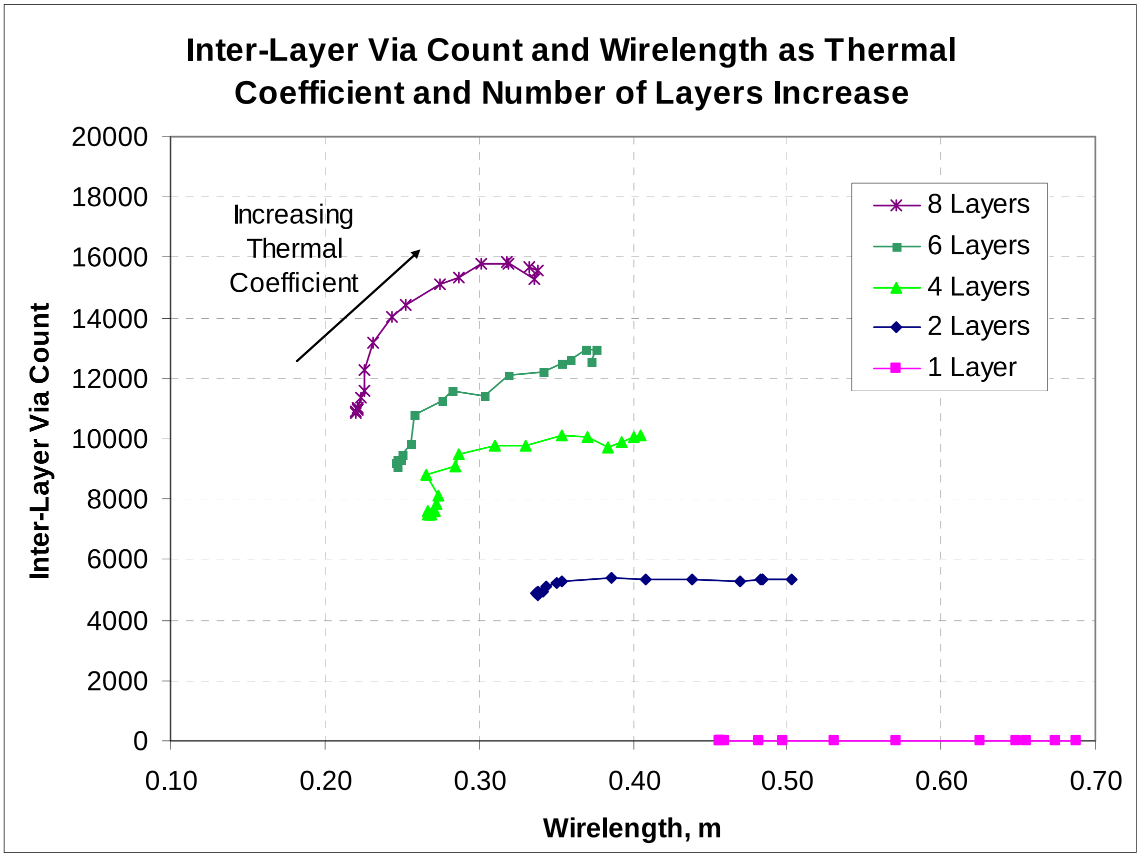

Figure 51. The effect of thermal placement on interlayer via count and wirelength as the number of layers is increased for ibm01.

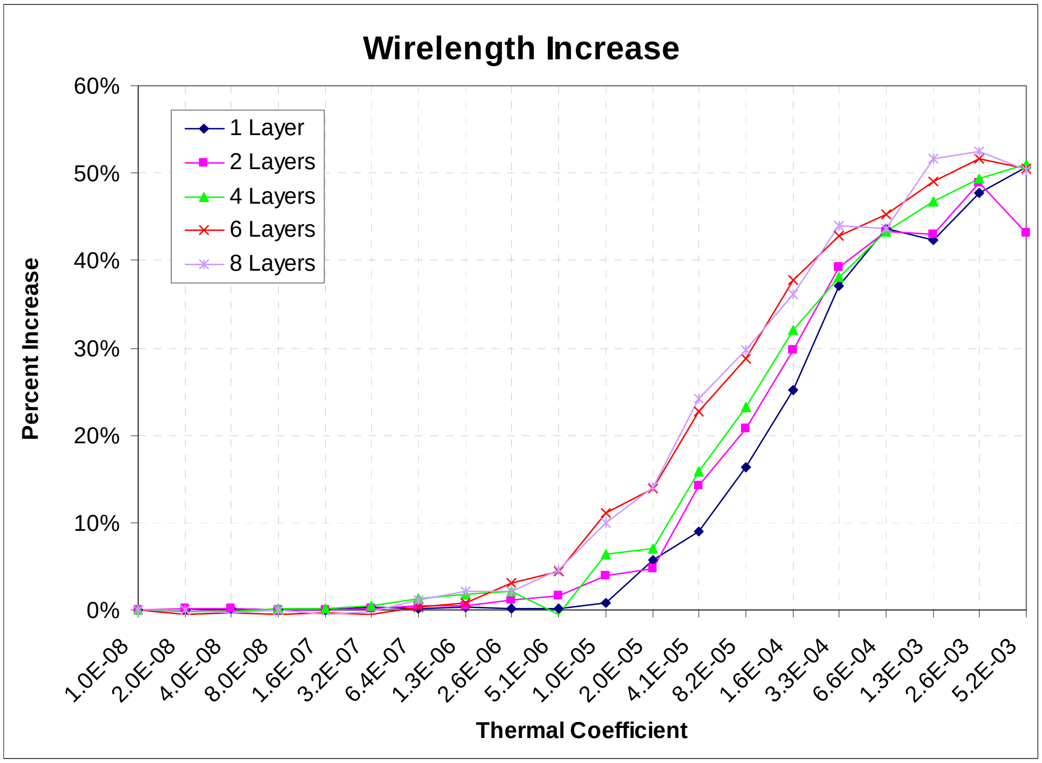

Figure 52. The percent increase in the wirelength of ibm01 as the thermal coefficient is increased with different numbers of layers.

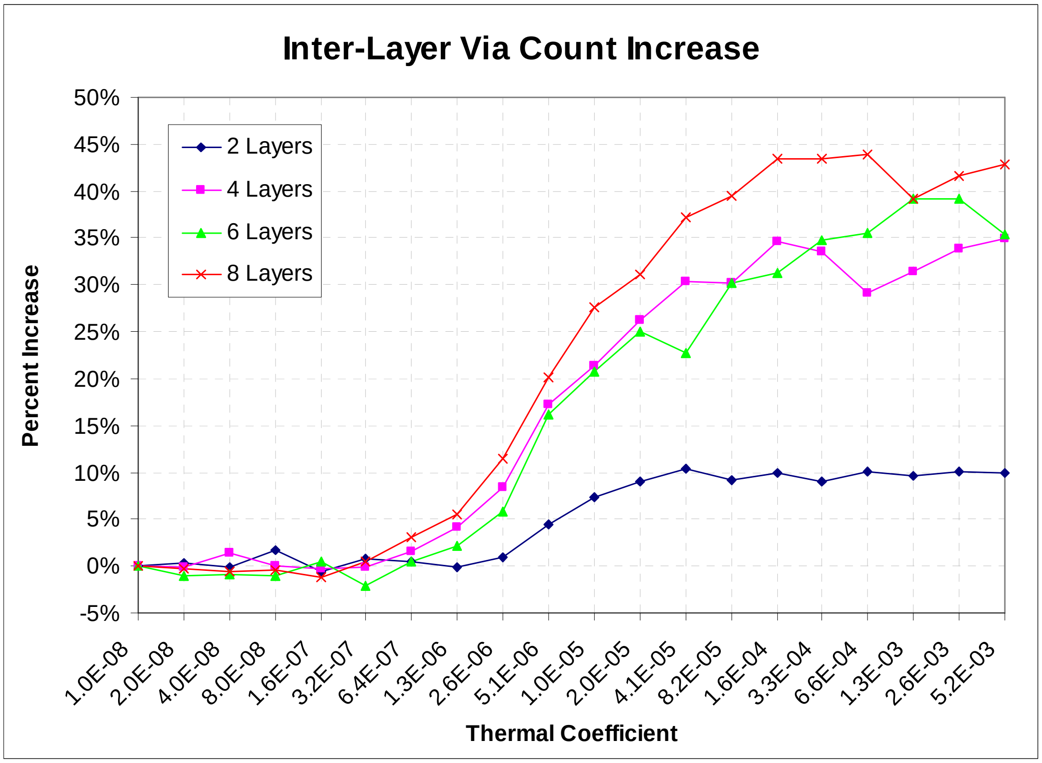

Figure 53. The percent increase in the interlayer via counts of ibm01 as the thermal coefficient is increased with different numbers of layers.

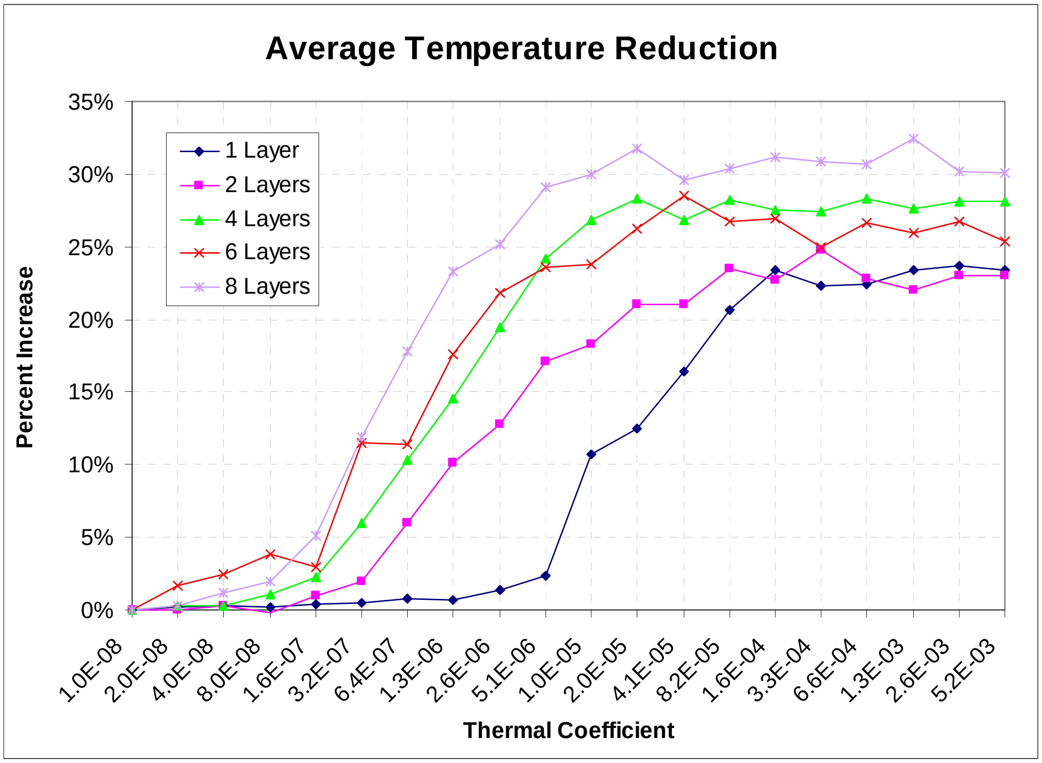

Figure 54. The percent reduction in the average temperature of ibm01 as the thermal coefficient is increased with different numbers of layers.

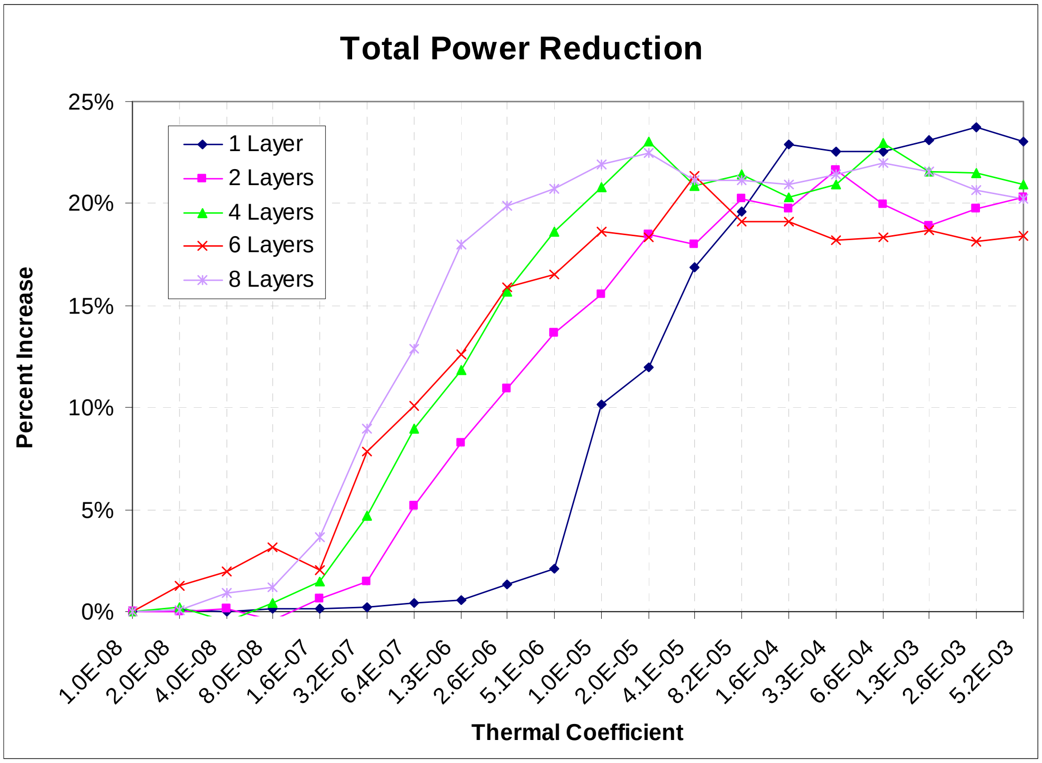

Figure 55. The percent reduction in the total power of ibm01 as the thermal coefficient is increased with different numbers of layers.

For any number of layers, as the thermal coefficient increases, the temperatures and power are reduced, but wirelengths and interlayer via counts increase. Wirelength increases similarly for any number of layers as the thermal coefficient is increased. The percent change in the interlayer via counts and temperature reduction becomes larger as the number of layer increases. The maximum achievable power reduction in each case is similar when the thermal coefficient is high. In each of these figures, the one layer case represents a 2D IC and shows that similar improvements can be made with our thermal placement method on 2D ICs as with multi-layer 3D ICs.

6.6.3 Thermal Placement of Various Benchmarks

Thermal placements were created for the benchmark circuits from Table 7 using an interlayer via coefficient of 1×10^{-5} and by varying the thermal coefficient from 0 to 4.1×10^{-5}. The average temperatures, maximum temperatures, total power, wirelength, and interlayer via counts are shown in Table 16 through Table 20 respectively. The percent reduction in average temperature, maximum temperature, and total power are shown in Figure 56 through Figure 58. Figure 59 and Figure 60 show the percent increase in the wirelength and interlayer via counts for each of the benchmark circuits. Figure 61 summarizes these results by averaging the percent change across all benchmark circuits as the thermal coefficient increases. On average, maximum temperatures can be reduced by 33%, average temperatures by 29%, and power by 23% as the thermal coefficient is increased. Wirelength degrades by 18% and interlayer via counts by 27% at this level of thermal optimization. Overall, thermal placement produced fairly similar improvements across a wide range of circuit sizes. In Figure 61, it is observed that wirelength does not start to increase that much until thermal coefficients are greater than 2.56×10^{-6}. At that point, the wirelengths are only increased by 1%, but the average temperatures are reduced by 19%. This thermal coefficient value of 2.56×10^{-6} produces a larger thermal improvement with only minor degradation in wirelengths and interlayer via counts.

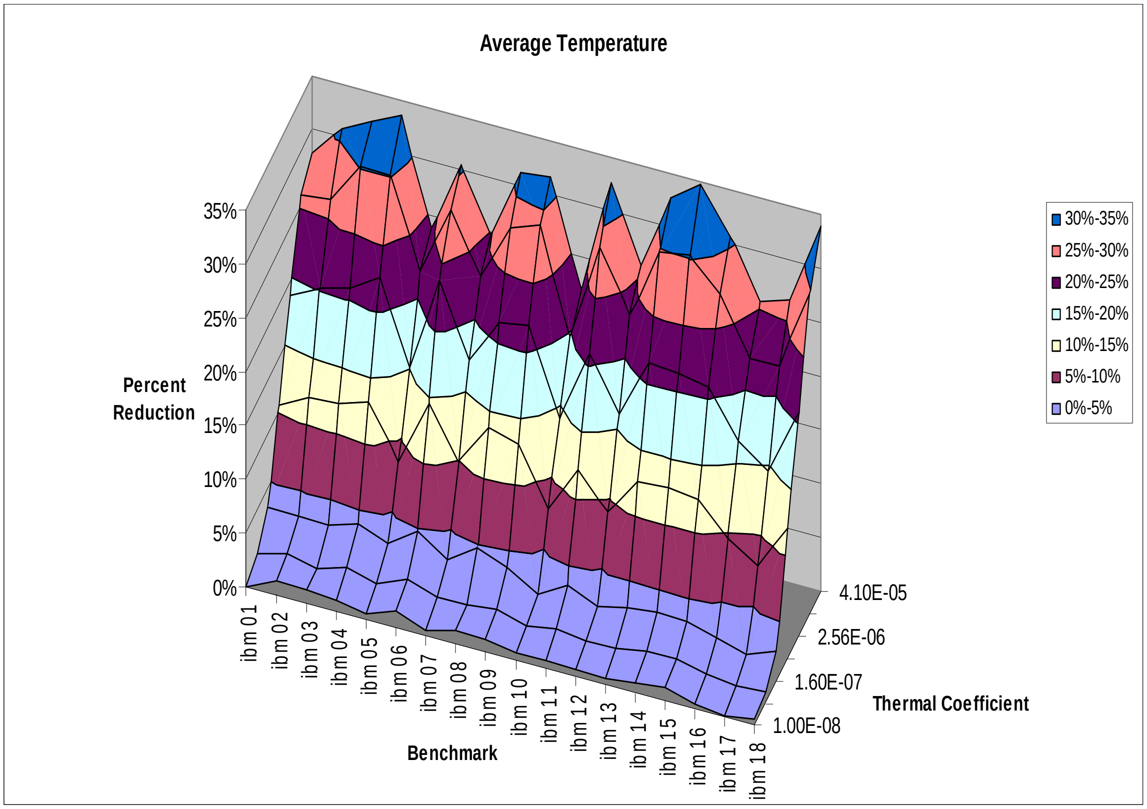

Figure 56. Percent reduction of the average temperature for benchmarks ibm01 to ibm18 as the thermal coefficient is increased.

Table 16. Average temperature for thermal placements of the benchmark circuits.

		Thermal Coefficient							
		0	1.0E-08	4.0E-08	1.6E-07	6.4E-07	2.6E-06	1.0E-05	4.1E-05
Benchmark	ibm01	13.7	13.7	13.6	13.3	12.2	11.1	10.1	9.9
	ibm02	18.0	17.7	17.7	17.4	15.8	14.3	13.2	12.4
	ibm03	15.7	15.5	15.5	15.2	13.7	12.4	10.9	10.6
	ibm04	15.9	15.8	15.6	15.3	13.8	12.3	11.1	10.6
	ibm05	24.1	23.9	23.8	23.4	22.0	20.4	18.8	18.3
	ibm06	21.5	21.1	21.0	20.4	18.3	16.3	15.4	14.9
	ibm07	19.9	19.8	19.6	19.3	17.9	16.4	15.3	14.9
	ibm08	17.0	16.8	16.7	16.2	14.7	13.4	12.2	11.7
	ibm09	16.1	15.9	15.8	15.5	14.0	12.5	11.4	11.0
	ibm10	16.4	16.3	16.2	16.1	15.2	14.1	13.2	12.8
	ibm11	17.4	17.2	17.1	16.8	15.3	13.7	12.4	11.7
	ibm12	17.7	17.6	17.5	17.2	16.1	14.8	13.6	13.1
	ibm13	17.3	17.2	17.1	16.8	15.1	13.6	12.2	11.6
	ibm14	16.2	16.0	15.9	15.6	14.1	12.7	11.3	10.6
	ibm15	20.0	19.7	19.6	19.3	17.5	15.8	14.5	13.8
	ibm16	19.9	19.8	19.7	19.4	18.0	16.6	15.5	14.8
	ibm17	20.3	20.3	20.1	20.0	18.7	17.3	15.8	15.0
	ibm18	15.6	15.5	15.4	15.1	13.7	12.3	10.9	10.3

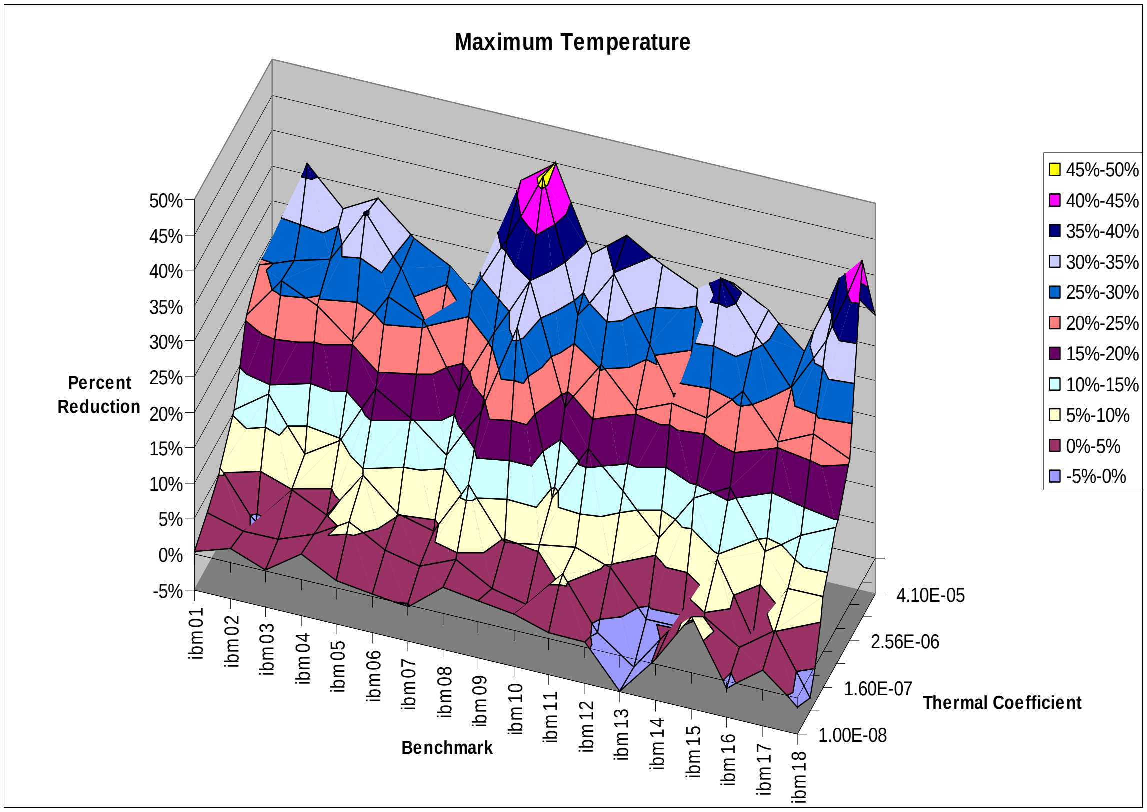

Figure 57. Percent reduction of the maximum temperature for benchmarks ibm01 to ibm18 as the thermal coefficient is increased.

Table 17. Maximum temperature for thermal placements of the benchmark circuits.

		Thermal Coefficient							
		0	1.0E-08	4.0E-08	1.6E-07	6.4E-07	2.6E-06	1.0E-05	4.1E-05
Benchmark	ibm01	21.7	21.6	21.1	20.5	19.4	17.2	16.4	16.6
	ibm02	30.4	29.8	30.1	30.7	26.2	22.3	22.5	19.3
	ibm03	26.3	26.3	26.0	25.0	24.6	19.9	19.1	18.1
	ibm04	26.8	25.8	25.9	26.1	24.2	21.4	17.3	17.7
	ibm05	49.0	48.5	46.1	44.5	40.0	35.3	35.0	34.6
	ibm06	36.1	36.0	34.9	34.2	30.6	26.8	29.4	26.4
	ibm07	32.6	32.6	31.9	31.3	28.9	27.0	23.5	25.3
	ibm08	33.9	32.6	32.4	29.9	27.5	24.9	21.5	19.9
	ibm09	36.2	35.1	35.7	34.5	24.7	23.5	19.4	19.9
	ibm10	30.7	30.0	30.6	29.0	28.0	25.1	22.1	20.5
	ibm11	33.4	33.1	31.4	31.3	27.7	24.9	21.7	21.0
	ibm12	31.8	31.5	32.1	30.3	27.9	25.3	23.4	20.9
	ibm13	32.6	34.6	34.2	33.1	30.4	25.3	27.1	22.2
	ibm14	31.0	30.7	30.5	31.4	26.7	24.3	19.2	20.4
	ibm15	37.9	34.8	36.1	34.3	31.7	27.8	26.0	26.0
	ibm16	35.1	35.5	34.6	35.2	32.1	27.9	26.4	25.7
	ibm17	41.4	40.3	39.3	38.3	36.5	32.0	27.8	25.6
	ibm18	39.1	39.6	40.3	36.0	33.5	31.0	21.4	25.7

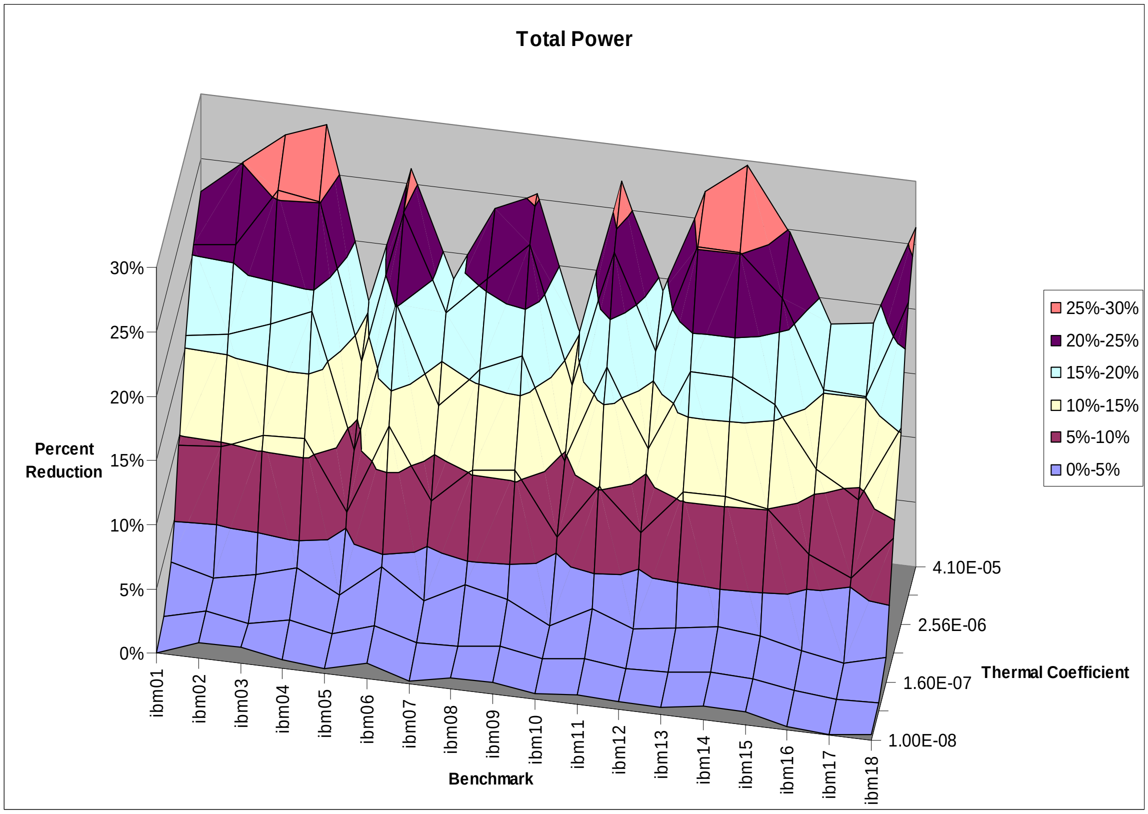

Figure 58. Percent reduction of the total power for benchmarks ibm01 to ibm18 as the thermal coefficient is increased.

Table 18. Total power for thermal placements of the benchmark circuits.

		Thermal Coefficient							
		0	1.0E-08	4.0E-08	1.6E-07	6.4E-07	2.6E-06	1.0E-05	4.1E-05
Benchmark	ibm01	0.44	0.45	0.44	0.43	0.40	0.37	0.35	0.34
	ibm02	0.85	0.84	0.84	0.84	0.77	0.71	0.67	0.64
	ibm03	0.78	0.77	0.77	0.76	0.70	0.65	0.58	0.57
	ibm04	1.08	1.07	1.06	1.04	0.96	0.88	0.81	0.77
	ibm05	2.00	1.99	1.98	1.96	1.88	1.83	1.74	1.69
	ibm06	1.36	1.34	1.33	1.30	1.18	1.08	1.02	1.00
	ibm07	2.16	2.16	2.14	2.12	2.00	1.89	1.78	1.77
	ibm08	2.01	1.99	1.99	1.94	1.80	1.69	1.60	1.53
	ibm09	1.94	1.92	1.91	1.89	1.73	1.61	1.48	1.45
	ibm10	3.42	3.40	3.39	3.37	3.22	3.09	2.97	2.90
	ibm11	2.73	2.71	2.69	2.65	2.45	2.26	2.08	1.99
	ibm12	4.06	4.03	4.02	3.99	3.77	3.60	3.38	3.28
	ibm13	3.10	3.08	3.07	3.03	2.77	2.55	2.32	2.26
	ibm14	6.02	5.95	5.93	5.85	5.38	4.96	4.51	4.23
	ibm15	7.00	6.93	6.91	6.83	6.30	5.89	5.46	5.27
	ibm16	9.80	9.77	9.72	9.64	9.12	8.69	8.31	8.03
	ibm17	11.69	11.68	11.63	11.57	11.05	10.61	9.93	9.52
	ibm18	8.45	8.41	8.39	8.29	7.69	7.16	6.53	6.22

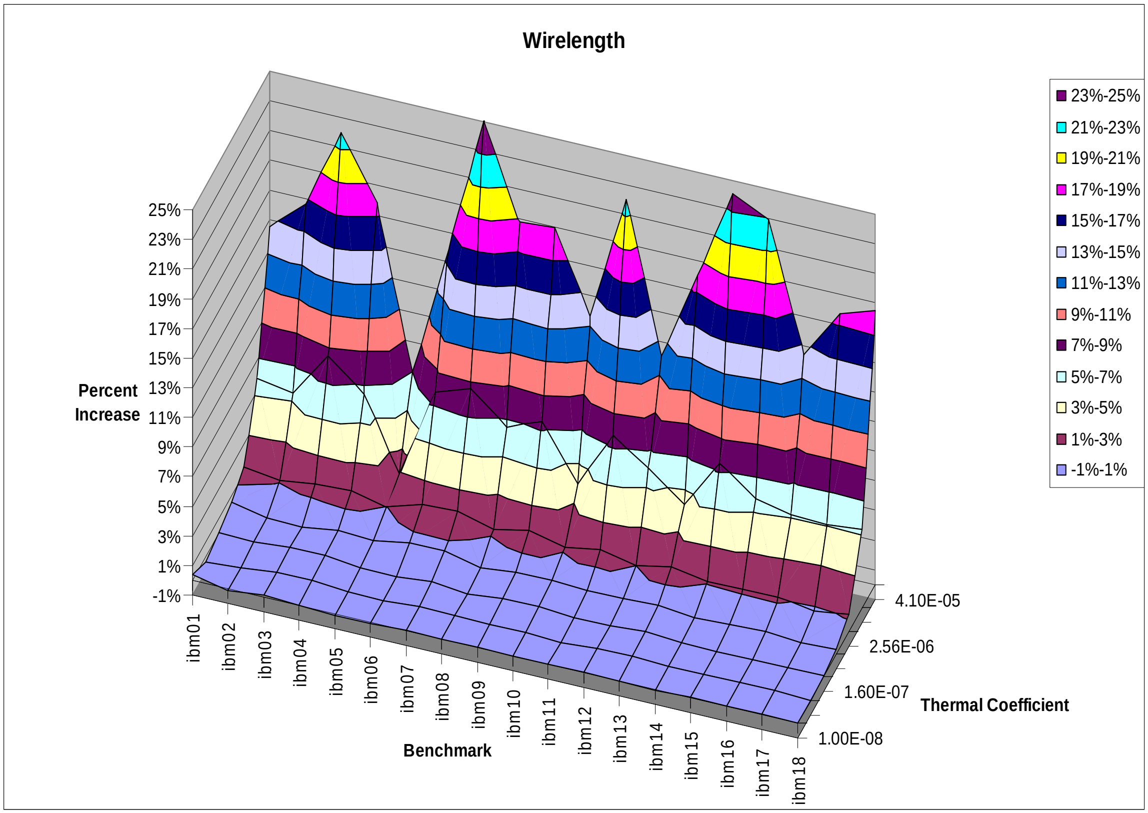

Figure 59. Percent increase in the wirelength for benchmarks ibm01 to ibm18 as the thermal coefficient is increased.

Table 19. Wirelength for thermal placements of the benchmark circuits.

		Thermal Coefficient							
		0	1.0E-08	4.0E-08	1.6E-07	6.4E-07	2.6E-06	1.0E-05	4.1E-05
Benchmark	ibm01	0.27	0.27	0.27	0.27	0.27	0.28	0.29	0.31
	ibm02	0.71	0.71	0.71	0.71	0.71	0.72	0.75	0.83
	ibm03	0.68	0.68	0.68	0.68	0.68	0.69	0.74	0.83
	ibm04	0.83	0.84	0.84	0.84	0.84	0.85	0.89	0.98
	ibm05	1.74	1.73	1.74	1.74	1.74	1.75	1.77	1.86
	ibm06	1.08	1.08	1.07	1.08	1.08	1.09	1.16	1.24
	ibm07	1.57	1.57	1.57	1.57	1.58	1.60	1.70	1.96
	ibm08	1.70	1.70	1.71	1.70	1.71	1.72	1.82	2.03
	ibm09	1.50	1.50	1.50	1.51	1.51	1.53	1.62	1.79
	ibm10	2.77	2.77	2.77	2.78	2.78	2.80	2.88	3.15
	ibm11	2.18	2.18	2.18	2.18	2.18	2.21	2.34	2.66
	ibm12	3.58	3.58	3.58	3.59	3.60	3.62	3.80	4.01
	ibm13	2.79	2.79	2.79	2.79	2.80	2.83	2.90	3.29
	ibm14	5.81	5.81	5.81	5.81	5.82	5.87	6.24	7.21
	ibm15	6.70	6.71	6.71	6.70	6.73	6.78	7.09	8.24
	ibm16	8.52	8.52	8.53	8.52	8.55	8.62	8.96	9.75
	ibm17	12.19	12.20	12.20	12.20	12.22	12.29	12.81	14.36
	ibm18	9.13	9.12	9.13	9.14	9.16	9.23	9.61	10.81

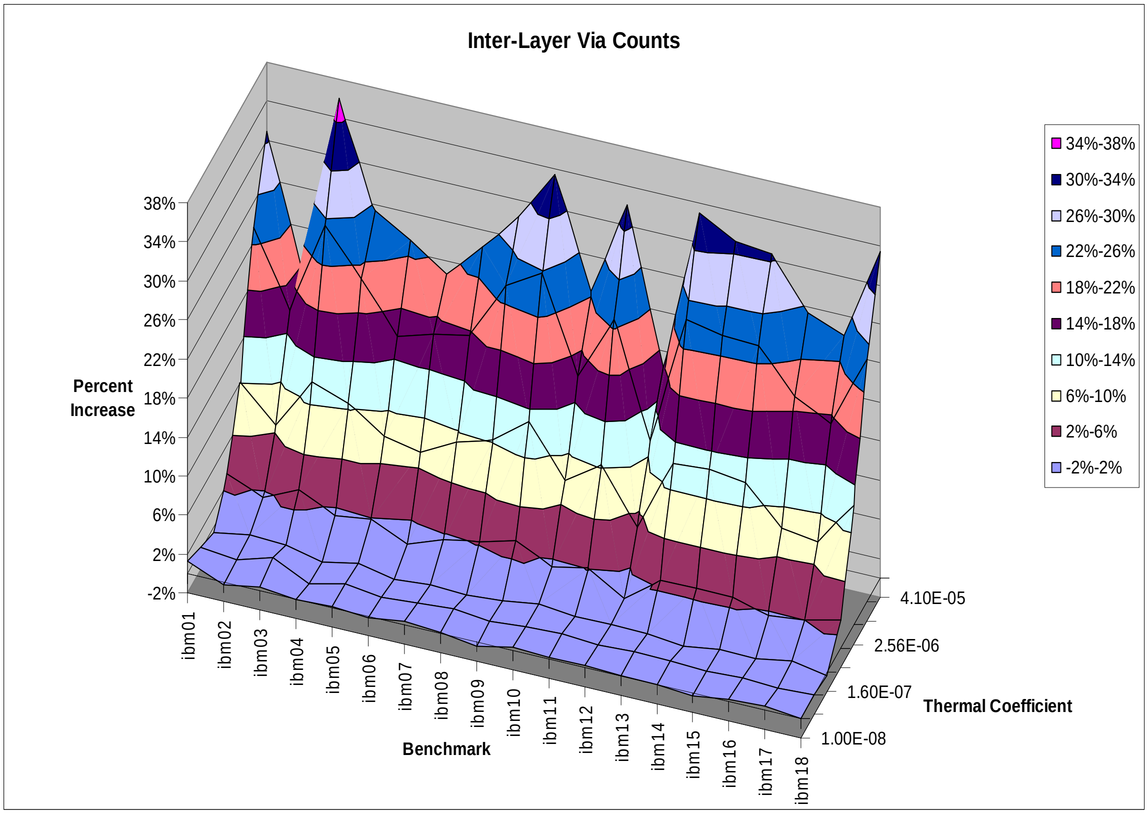

Figure 60. Percent increase in the interlayer via count for benchmarks ibm01 to ibm18 as the thermal coefficient is increased.

Table 20. Interlayer via counts for thermal placements of the benchmark circuits.

		Thermal Coefficient							
		0	1.0E-08	4.0E-08	1.6E-07	6.4E-07	2.6E-06	1.0E-05	4.1E-05
Benchmark	ibm01	7445	7543	7469	7405	7672	8191	9195	9750
	ibm02	15061	15019	15053	15063	15285	16047	17454	17703
	ibm03	12453	12499	12582	12445	12854	13930	15624	16937
	ibm04	18689	18685	18547	18500	18948	20644	22584	23436
	ibm05	24254	24283	24271	24196	24710	26181	28078	29907
	ibm06	25676	25640	25553	25411	25718	27517	30001	30968
	ibm07	31417	31525	31374	31214	31910	34279	36970	39042
	ibm08	32738	32778	32787	32443	33466	36082	40460	42009
	ibm09	32335	32210	32238	32130	33359	36513	40681	43167
	ibm10	45522	45665	45524	45499	46174	49061	52835	55693
	ibm11	42790	42871	42846	42613	43680	47177	52544	56527
	ibm12	60247	60379	60159	59993	60171	63146	66974	70081
	ibm13	52455	52440	52471	52234	53816	58830	65296	69768
	ibm14	101512	101528	101522	100650	104219	114124	125669	132956
	ibm15	119647	119281	119501	119205	122756	133922	147876	156210
	ibm16	136661	136880	136621	136111	138919	147766	162881	171469
	ibm17	155447	155969	155275	155789	158564	167258	182438	192702
	ibm18	139807	139814	139787	139186	143926	156905	174446	186553

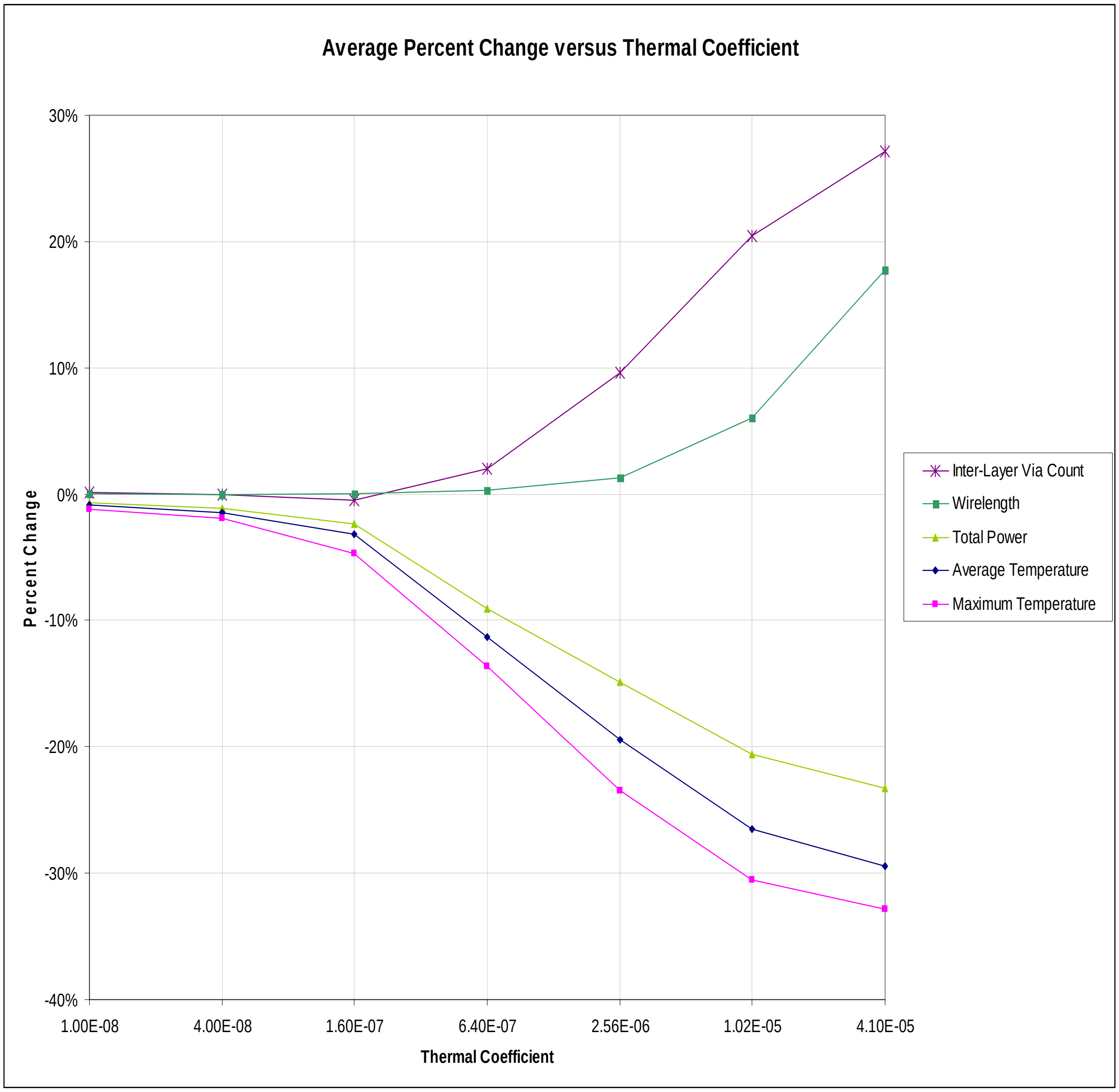

Figure 61. Average percent change in the interlayer via counts, wirelengths, total power, average temperatures, and maximum temperatures for benchmarks ibm01 to ibm18 as the thermal coefficients are varied.

In Figure 62, the run time efficiency of thermal placement is shown versus the number of cells in the benchmark circuits. The run time of the placement method without thermal considerations is also plotted and shows that thermal placement has a run time overhead of about 12%. In this figure, the run time of the thermal placement method appears to scales well with increasing circuit sizes and is close to linear in efficiency.

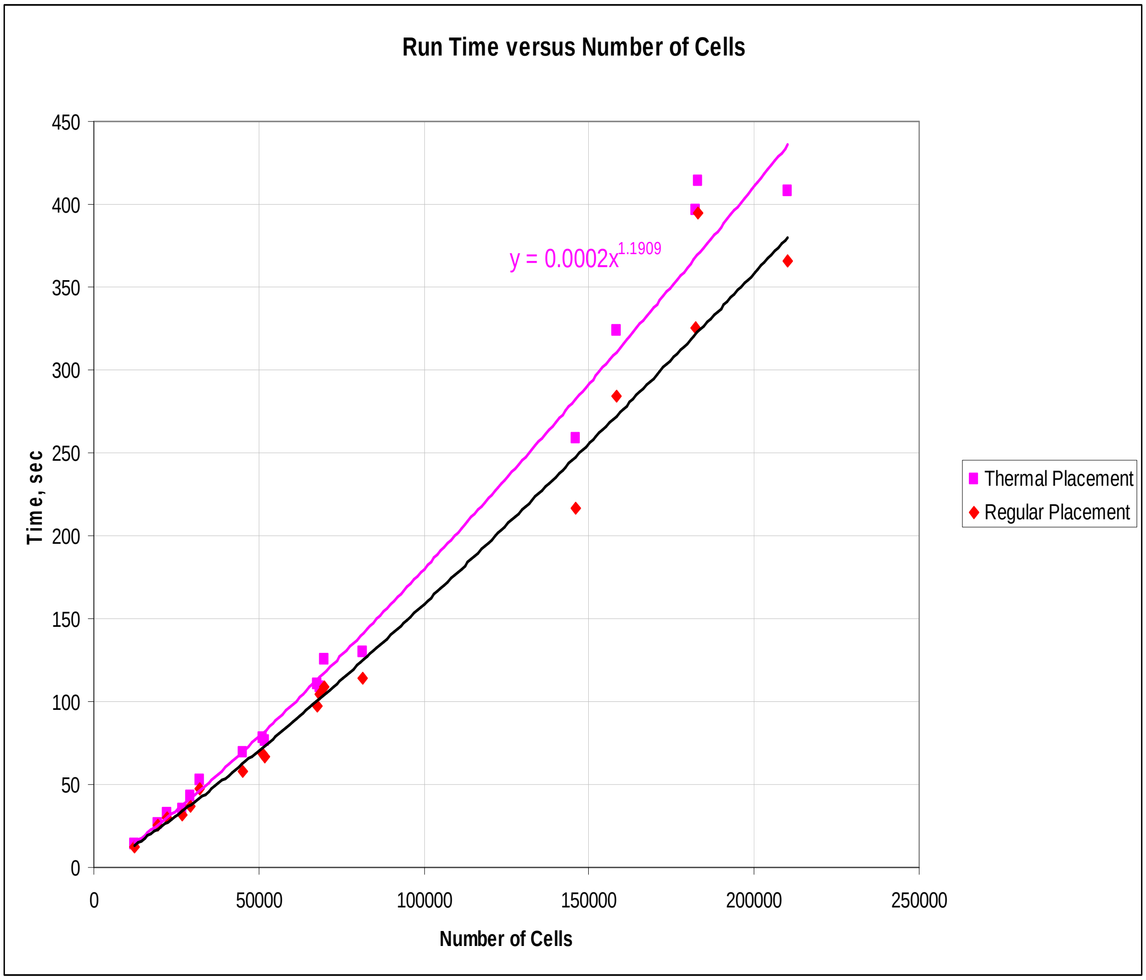

Figure 62. Runtime of thermal placement method.

6.7 Conclusions

An efficient and effective thermal placement method was developed for 3D ICs that allows the tradeoff between wirelength, interlayer via count, and temperature to be explored. The method considers not only the thermal environment, but also power usage. As a result, both temperatures and power are minimized in the process. In global placement, net weighting is used to reduce the length of high-powered nets and nets with driver cells having high thermal resistances. Additional nets are added to move cells toward lower thermal resistances. In legalization, the cost of each cell movement incorporates the thermal objective so that objective function degradation does not occur. Generally, as the thermal coefficient is increased, temperatures and power are reduced, and the wirelength and interlayer via counts are increased. However, wirelength and interlayer via counts are sometimes reduced when the thermal coefficient is increased if it is necessary for temperature reduction. Depending on the technology, interlayer via capacitances may be the dominant contributor to the total power so thermal optimization may involve adjusting the interlayer vias coefficient to reduce the power caused by interlayer via counts as well as adjusting the thermal coefficient. Our thermal placement method was shown to be effective for both 2D ICs and 3D ICs with increasing numbers of layers, and thermal placement produced similar improvements across all the benchmark circuits tested. Finally, the run time efficiency was shown to be nearly linear with circuit sizes.

7 Thermal Via Placement in 3D ICs

7.1 Introduction

As thermal problems become more evident, new physical design paradigms and tools are needed to alleviate them. With increasing power densities and thermal resistances of VLSI chips, a greater amount of heat sinking could be used to alleviate thermal problems. However, improved heat sinking may not be practical or affordable. Using existing technology, thermal vias can be fabricated as dummy vias for improving heat conduction and reducing thermal problems. Thermal vias with their high thermal conductivities greatly reduces the thermal resistance along the heat dissipation paths internally, similar to what heat sinks do externally. However, thermal vias take up valuable routing space, and therefore, algorithms are needed to minimize their usage while placing them in areas where they would make the greatest impact. Thermal problems are expected to be more prominent in 3D ICs than in 2D ICs, and in order to take advantage of the benefits of 3D integration, these thermal problems need to be overcome. Fortunately, thermal vias can be used to alleviate these thermal problems and would be more effective in 3D ICs.

With our thermal via placement method [97] [98], thermal vias are assigned to specific areas of a 3D IC and used to adjust their effective thermal conductivities. Temperatures are quickly calculated during each iteration using FEA and used to make iterative adjustments to the thermal conductivities in order to achieve a desired thermal objective. Various thermal objectives can be used with this method such as achieving a desired maximum operating temperature. With this method, 49% fewer thermal vias are needed to obtain a 47% reduction in the maximum temperatures, and 57% fewer thermal vias are needed to obtain a 68% reduction in the maximum thermal gradients than would be needed using a uniform distribution of thermal vias to obtain these same thermal improvements. Similar results were seen for other thermal objectives, and the method efficiently achieves its thermal objective while minimizing the thermal via utilization.

7.2 Thermal Vias

The idea of using thermal vias to alleviate thermal problems was first utilized in the design of packaging and printed circuit boards (PCBs). Lee *et al.* studied arrangements of thermal vias in the packaging of multichip modules (MCMs) and found that as the size of thermal via islands increased, more heat removal was achieved but less space was available for routing [99]. Li studied the relationships between design parameters and the thermal resistance of thermal via clusters in PCBs and packaging [100]. These relationships were determined by simplifying via clusters into parallel networks using the observation that heat transfer is much more efficient vertically through the thickness than laterally from heat spreading. Pinjala *et al*. performed further thermal characterizations of thermal vias in packaging [101], and Yamaji *et. al*. examined the effectiveness of thermal vias in 3D MCM [102]. Although these papers have limited application for the placement of thermal vias inside chips, the basic use and properties of thermal via are demonstrated. It is important to realize that there is a tradeoff between routing space and heat removal, indicating that thermal vias should be used sparingly. Simplified thermal calculations can be used for thermal vias, and the direction of heat conduction is primarily in the orientation of the thermal vias.

Chiang *et al*. first suggested that “dummy thermal vias” can be added to the chip substrate as additional electrically isolated vias to reduce effective thermal resistances and potential thermal problems [103]. A number of papers have addressed the potential of integrating thermal vias directly inside chips to reduce thermal problems internally [12] [14] [29] [103] [104]. Because of the insulating effects from numerous dielectric layers, thermal problems are greater in 3D ICs and thermal vias can have a larger impact in 3D ICs than in 2D ICs. In addition, interconnect structures can create efficient thermal conduits and greatly reduce chip temperatures [105].

It has become of particular interest to design efficient heat conduction paths right into a chip to eliminate localized hot spots directly. Despite all the work that has been done in evaluating thermal vias, thermal via placement algorithms are lacking for both 3D and 2D ICs. As circuits and temperature profiles increase in complexity, efficient algorithms are needed to determine the most effective location and number of thermal vias to use. The thermal via placement method [97] presented in this thesis uses designated thermal via regions to place thermal vias and efficiently adjusts the density of thermal vias in each of these regions with minimal perturbations on routing. In the design process, thermal via placement can be applied after placement and before routing.

As discussed in Section 2.2, there are a number of different fabrication technologies being developed for 3D ICs, and it is not entirely certain which of these will ultimately prevail. Therefore, developing 3D CAD tools need to be flexible and general enough to handle these different technologies and future developments. In its formulation, our thermal via placement method was designed to be as technology independent as possible. However, in our implementation, we focused on 3D technologies that have relatively thick insulative layers (such as wafer-bonded 3D ICs) that can be augmented with highly conductive metal vias to greatly reduce the thermal resistances in selective areas. In our experiments, thermal via placement was applied to 3D ICs produced using wafer bonding technology, specifically, the wafer bonding fabrication technology used by MIT Lincoln Labs [24] [89] [90]. With different fabrication technologies, different materials and geometries would change the thermal properties of the thermal vias, but the underlining thermal via placement methodology would remain the same, i.e., iteratively adjust the thermal conductivities of the thermal via regions using a thermal objective. For other fabrication methods, the relationship between thermal conductivity and thermal via density would differ.

Thermal via construction, availability, and effectiveness greatly depends on the specific fabrication method used. In order for thermal vias to be effective, they need to be made out of a highly thermal conductive material such as copper, there must be adequate capacity available to include them, their surrounding material must be of low thermal conductivity, and large thermal gradients need to be present. In particular, wafer bonded 3D ICs have been shown to obtain great improvements in this regard [14], but 3D ICs produced using crystallization and TFTs may also benefit from this method. However, it is unclear from the literature as to how much improvement can be made with thermal vias in these 3D ICs. Their thinner insulating layers would by themselves reduce thermal resistances and reduce the effectiveness of thermal vias, but higher interlayer via densities would increase the effectiveness of using thermal vias. In addition, thinner layers could create higher power densities that would result in higher thermal gradients making thermal vias more effective. This could be an area of future research as crystallization methods for producing 3D ICs matures. Although thermal vias have also been utilized in 3D MCMs as a way of mitigating thermal problems, there are limitations in applying our method to 3D MCMs. Generally, 3D MCMs provide connections to different dies at only the edges of the chips, and the numbers of inter-die connections are limited. In addition, the number of possible thermal vias and the ability to intersperse them throughout the structure is also limited in 3D MCMs.

7.3 Thermal Via Regions

In order to make the placement of thermal vias more manageable, certain areas of the chip are reserved for placing thermal vias. The location of the thermal via regions in a 3D IC is shown in Figure 63. These regions are evenly placed between the rows in a standard cell design within an inter-row space. An inter-row space is necessary for the interlayer vias because they go through the silicon. However, in other designs, white space could be used instead for interlayer vias and thermal vias. In Figure 63, a thermal via regions is enlarged to show its individual thermal vias. Each thermal via region would contain a uniform density of thermal vias, and the thermal via placement algorithm determines the density in each of these regions. In Figure 64, the 3D IC has been rotated 90^o about the z-axis and cut away to show the individual devices, interconnect, and thermal vias.

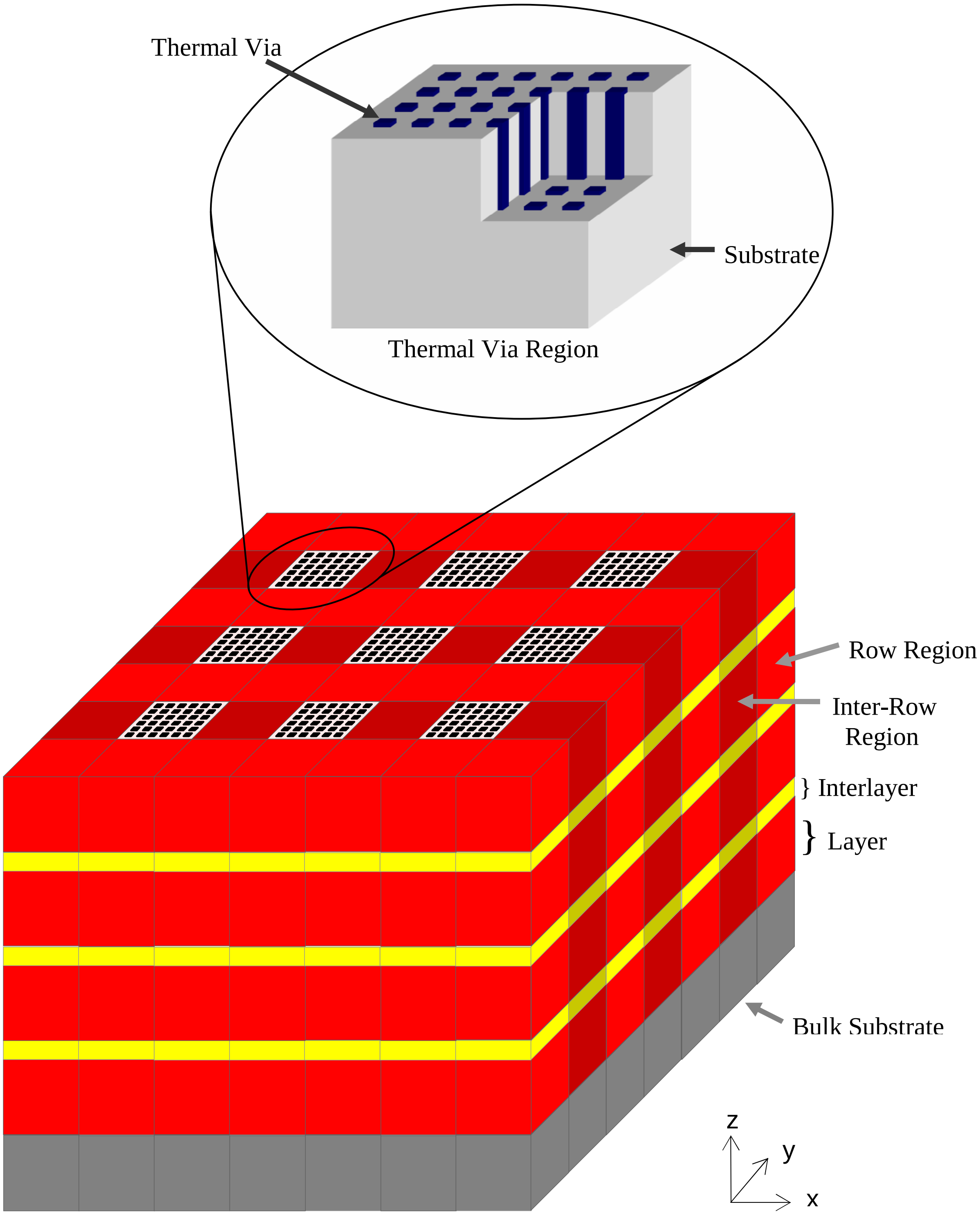

Figure 63. Thermal mesh for a 3D IC with thermal via regions.

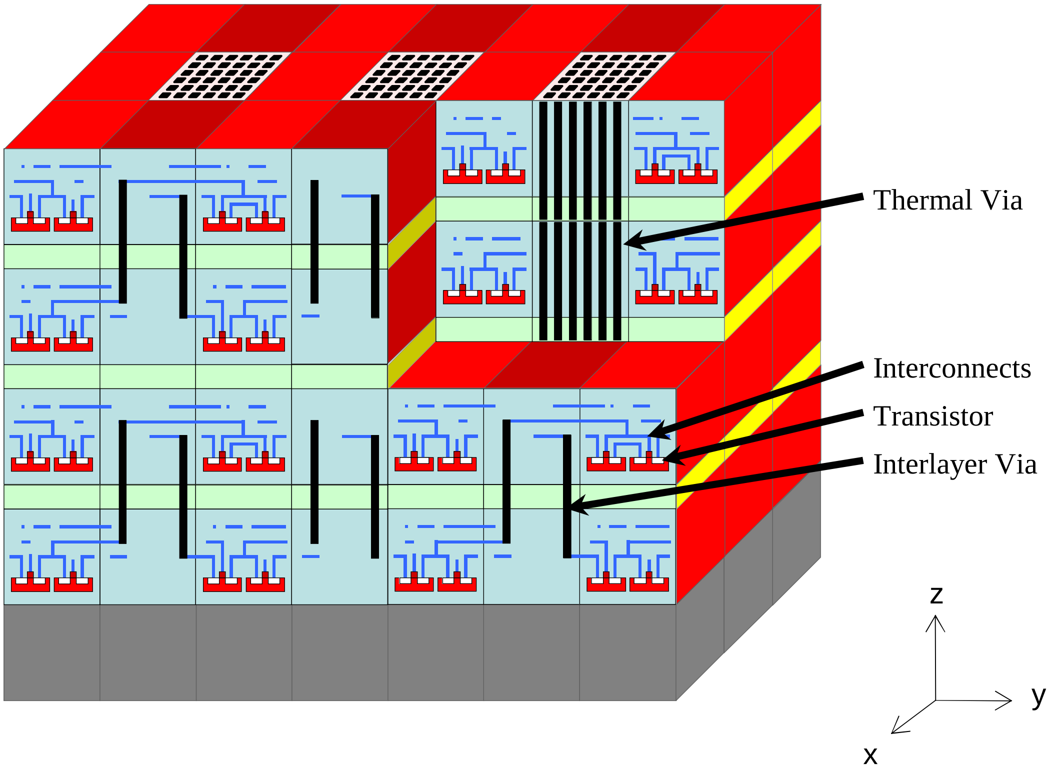

Figure 64. Cutaway showing transistors and interconnects.

The thermal via regions are composed of electrically isolated vias and are oriented vertically between the rows. The density of the thermal vias determines the thermal conductivity of the region which in turn determines the thermal properties of the entire chip. They are generally obstacles to routing except for regions that require only a low density of thermal vias. Placing thermal vias in specific regions allows for predictable obstacles to routing, and allows for regularity and uniformity in the entire design process. Moreover, it limits the density of these routing obstacles in any particular area so that the design does not become unroutable.

The value of the effective thermal conductivity in any particular direction depends on the density of thermal vias that are arranged in that direction. Increasing the number of thermal vias in one direction does increase the effective thermal conductivity in the other directions but at an order of magnitude less. For simplicity, the interdependence can be considered to be negligible, and the effective thermal conductivities in the *x*, *y*, and *z* directions can be considered to be independent to a certain extent.

Current integration technologies for producing 3D ICs result in the layers being closely stacked together and the design space being tightly compressed in the *z* direction. In addition, the location of the heat sink in relation to the heat sources produces a heat flux that is primarily downward in direction with very minor lateral components. Furthermore, with the thermal via regions being oriented vertically, lateral thermal vias would have little effect. As a result, lateral thermal conductivities (in the *x* and *y* directions) are generally unchanged by this method because the thermal gradients in the vertical (*z*) direction are almost two orders of magnitude larger than in the lateral directions. In this thesis, the method will be developed for all three directions, but for the reasons outlined above, only vertically oriented thermal vias will be considered in the implementation and results.

In this method, 3D ICs are meshed into regions (elements) as shown in Figure 63. Vertically (in the z direction), the chip is separated into bulk substrate, layer, and interlayer elements. The bulk substrate is located at the bottom attached to the heat sink. Above the bulk substrate elements are the layer and interlayer elements. In a 3D IC, the interlayers are composed of interlayer vias and bonding materials that connect the layers together, and the layers contain the device and metal levels. Transistors, the primary sources of heat, are located at the bottom of each layer in the row regions. In standard cell designs, cells are placed into rows with inter-row spaces between them. These inter-row regions are necessary to accommodate interconnects between different layers, and some of these areas can serve as thermal via regions. In our method, thermal via regions are represented as special elements having variable thermal conductivities in the FEA mesh.

7.4 Iterative Thermal Via Method

For a given placed 3D circuit, an iterative method was developed in which, during each iteration, the thermal conductivities of certain FEA elements (thermal via regions) are incrementally modified so that thermal problems are reduced or eliminated. Thermal vias are generically added to elements to achieve the desired thermal conductivities. The goal of this method is to satisfy given thermal requirements using as few thermal vias as possible, i.e., keeping the thermal conductivities as low as possible.

7.4.1 Updating the Thermal Conductivities

During each iteration, the thermal conductivities of thermal via regions are modified, and these thermal conductivities reflect the density of thermal vias needed to be utilized within the region. The new thermal conductivities are derived from the element FEA equations. Using Equation (8), we get:

$$[k_c]\{t\}=\{p\} \tag{90}$$

where $\{t\}$ are the nodal temperatures in the element and $\{p\}$ are the fractions of the nodal heat that transverse this particular element. Under the reasonable assumption that $\{p\}$ does not change between the old and new values in an iteration, we get the following expression:

$$[k_c]_{new}\{t\}_{new}=[k_c]_{old}\{t\}_{old} \tag{91}$$

If we multiply Equation (91) by $\frac{w}{2hd}[1\ -1\ -1\ 1\ 1\ -1\ -1\ 1]$, $\frac{h}{2wd}[1\ 1\ -1\ -1\ 1\ 1\ -1\ -1]$, and $\frac{d}{2wh}[1\ 1\ 1\ 1\ -1\ -1\ -1\ -1]$, respectively, we obtain the following equations:

$$K_x^{new}\Delta t_x^{new}=K_x^{old}\Delta t_x^{old} \tag{92}$$

$$K_y^{new}\Delta t_x^{new}=K_y^{old}\Delta t_y^{old} \tag{93}$$

$$K_z^{new}\Delta t_z^{new}=K_z^{old}\Delta t_z^{old} \tag{94}$$

where $\Delta t_x^{new}=\frac{t_0^{new}+t_3^{new}+t_4^{new}+t_7^{new}}{4}-\frac{t_1^{new}+t_2^{new}+t_5^{new}+t_6^{new}}{4}$,

$\Delta t_x^{old}=\frac{t_0^{old}+t_3^{old}+t_4^{old}+t_7^{old}}{4}-\frac{t_1^{old}+t_2^{old}+t_5^{old}+t_6^{old}}{4}$,

$\Delta t_y^{new}=\frac{t_0^{new}+t_1^{new}+t_4^{new}+t_5^{new}}{4}-\frac{t_2^{new}+t_3^{new}+t_6^{new}+t_7^{new}}{4}$,

$\Delta t_y^{old}=\frac{t_0^{old}+t_1^{old}+t_4^{old}+t_5^{old}}{4}-\frac{t_2^{old}+t_3^{old}+t_6^{old}+t_7^{old}}{4}$

$\Delta t_z^{new}=\frac{t_0^{new}+t_1^{new}+t_2^{new}+t_3^{new}}{4}-\frac{t_4^{new}+t_5^{new}+t_6^{new}+t_7^{new}}{4}$,

and $\Delta t_z^{old}=\frac{t_0^{old}+t_1^{old}+t_2^{old}+t_3^{old}}{4}-\frac{t_4^{old}+t_5^{old}+t_6^{old}+t_7^{old}}{4}$.

K_x^{old}, K_y^{old}, and K_z^{old} are the thermal conductivities in the *x*, *y*, and *z* directions before the iteration. K_x^{new}, K_y^{new}, and K_z^{new} are the thermal conductivities in the *x*, *y*, and *z* directions after the iteration. Δt_x^{old}, Δt_y^{old}, and Δt_z^{old} are the change in temperature across the element with respect to the x, y, and z directions before the iteration. Δt_x^{new}, Δt_y^{new}, and Δt_z^{new} are the change in temperature across the element with respect to the x, y, and z directions after the iteration.

The thermal gradients, $g_{new} = \{g_x^{new}, g_y^{new}, g_z^{new}\}$ and $g_{old} = \{g_x^{old}, g_y^{old}, g_z^{old}\}$, are functions of the position, (*x*, *y*, *z*), in the element, and at the center of the element, (x_c, y_c, z_c), they are equal to:

$$g_x^{new} = \frac{\Delta t_x^{new}}{w}, g_x^{old} = \frac{\Delta t_x^{old}}{w} \qquad (95),(96)$$

$$g_y^{new} = \frac{\Delta t_y^{new}}{h}, g_y^{old} = \frac{\Delta t_y^{old}}{h} \qquad (97),(98)$$

$$g_z^{new} = \frac{\Delta t_z^{new}}{d}, g_z^{old} = \frac{\Delta t_z^{old}}{d} \qquad (99),(100)$$

g_{new} is the desired new thermal gradient, and g_{old} is the thermal gradient of the element before the iteration. The new *K*'s can be found by combining Equations (92)-(100):

$$K_x^{new} = \frac{K_x^{old}\Delta t_x^{old}}{\Delta t_x^{new}} = \frac{K_x^{old} g_x^{old}}{g_x^{new}} \qquad (101)$$

$$K_y^{new} = \frac{K_y^{old}\Delta t_y^{old}}{\Delta t_x^{new}} = \frac{K_y^{old} g_y^{old}}{g_x^{new}} \qquad (102)$$

$$K_z^{new} = \frac{K_z^{old}\Delta t_z^{old}}{\Delta t_z^{new}} = \frac{K_z^{old} g_z^{old}}{g_z^{new}} \qquad (103)$$

$\{g_x^{new}, g_y^{new}, g_z^{new}\}$ is chosen so that its component magnitudes are closer to some ideal thermal gradient value, g_{ideal}, than $\{g_x^{old}, g_y^{old}, g_z^{old}\}$ using the following equations:

$$\left|g_x^{new}\right| = g_{ideal}\left(\frac{\left|g_x^{old}\right|}{g_{ideal}}\right)^{\alpha} \qquad (104)$$

$$\left|g_y^{new}\right| = g_{ideal}\left(\frac{\left|g_y^{old}\right|}{g_{ideal}}\right)^{\alpha} \qquad (105)$$

$$\left|g_z^{new}\right| = g_{ideal}\left(\frac{\left|g_z^{old}\right|}{g_{ideal}}\right)^{\alpha} \tag{106}$$

where g_{ideal} is a nonnegative value and α is a user defined parameter between 0 and 1. If the magnitude of the old thermal gradient is below g_{ideal}, the value is increased toward g_{ideal} for the new thermal gradient. If the magnitude of the old thermal gradient is above g_{ideal}, the value is decreased toward g_{ideal}. If the magnitude of the old thermal gradient equals g_{ideal}, then the thermal gradient is not modified.

Combining Equations (101)-(106) yields the following formulas that can be used to update the thermal conductivities during each iteration:

$$K_x^{new} = K_x^{old}\left(\frac{\left|g_x^{old}\right|}{g_{ideal}}\right)^{1-\alpha} \tag{107}$$

$$K_y^{new} = K_y^{old}\left(\frac{\left|g_y^{old}\right|}{g_{ideal}}\right)^{1-\alpha} \tag{108}$$

$$K_z^{new} = K_z^{old}\left(\frac{\left|g_z^{old}\right|}{g_{ideal}}\right)^{1-\alpha} \tag{109}$$

These formulas decrease the K's when the thermal gradient is below g_{ideal} and increase them when the thermal gradient is above g_{ideal}. In the process, major sources of thermal impedance are eliminated in areas of greatest heat transfer, but for areas that are not on a critical heat sinking path, the K's are decreased to eliminate unnecessary thermal vias.

The ideal thermal gradients, g_{ideal}, must be chosen and specifically adjusted to satisfy some desired thermal objective. This method is flexible enough to handle a number of different thermal objectives, but only one thermal objective can be used at a time. Each objective type produces a different version of the thermal via placement method to specifically reach its objective value. For example, six different objective types were explored in these experiments: maximum thermal gradient, average thermal gradient, maximum temperature, average temperature, maximum thermal via density, and average thermal via density. Before the first iteration, the ideal thermal gradient is initialized to the magnitude of the average thermal gradient. Each objective type uses a different equation to update the ideal thermal gradient during each iteration, and they are listed as follows:

$$g_{ideal} = g_{ideal}\frac{g_{max}^{ideal}}{g_{max}} \text{ for an ideal maximum thermal gradient, } g_{max}^{ideal} \qquad (110)$$

$$g_{ideal} = g_{ideal}\frac{g_{ave}^{ideal}}{g_{ave}} \text{ for an ideal average thermal gradient, } g_{ave}^{ideal} \qquad (111)$$

$$g_{ideal} = g_{ideal}\frac{T_{max}^{ideal}}{T_{max}} \text{ for an ideal maximum temperature, } T_{max}^{ideal} \qquad (112)$$

$$g_{ideal} = g_{ideal}\frac{T_{ave}^{ideal}}{T_{ave}} \text{ for an ideal average temperature, } T_{ave}^{ideal} \qquad (113)$$

$$g_{ideal} = g_{ideal}\frac{m_{max}}{m_{max}^{ideal}} \text{ for an ideal maximum thermal via density, } m_{max}^{ideal} \qquad (114)$$

$$g_{ideal} = g_{ideal}\frac{m_{ave}}{m_{ave}^{ideal}} \text{ for an ideal average thermal via density, } m_{ave}^{ideal} \qquad (115)$$

For the thermal gradient and temperature objective types, if the value is too low in the previous iteration, g_{ideal} is increased for Equations (110)-(113). Likewise, if the previous value exceeds the desired value, g_{ideal} is decreased to lower the thermal effects. For the thermal via density objective types in Equations (114) and (115), if the previous iteration's value is too low, g_{ideal} is decreased so more thermal vias will be needed. If the previous iteration's value is too high, g_{ideal} is increased so fewer thermal vias will be used. Other thermal objectives are possible with this method as long as an appropriate equation is used to update the ideal thermal gradient and the value of the objective is reachable.

7.4.2 Thermal Via Density

As stated earlier, the thermal conductivity of a thermal via region is determined by its density of thermal vias. After thermal via placement determines the required thermal conductivities, the thermal via densities can be determined. Individual thermal vias are assumed to be much smaller than the thermal via region, are arranged uniformly within the thermal via region, and change the effective thermal conductivity of the thermal via region. The percentage of thermal vias or metallization, *m*, (also called thermal via density) in a thermal via region is given by the following equation:

$$m = \frac{nA_{via}}{wh} \tag{116}$$

where *n* is the number of individual thermal vias in the region, A_{via} is the cross sectional area of each thermal via, *w* is the width of the region, and *h* is the height of the region. The relationship between the percentage of thermal vias and the effective vertical thermal conductivity is given by:

$$K_z^{eff} = mK_{via} + (1-m)K_z^{layer} \tag{117}$$

where K_{via} is the thermal conductivity of the via material and K_z^{layer} is the thermal conductivity of the region without any thermal vias. Using this equation, the percentage of thermal vias can be found for any K_z^{new} provided that $K_z^{layer} \le K_z^{new} \le K_{via}$:

$$m = \frac{K_z^{new} - K_z^{layer}}{K_{via} - K_z^{layer}} \tag{118}$$

In this implementation, only the vertical thermal conductivities will be optimized using Equation (109). Equations (107) and (108) will not be used for updating the lateral thermal conductivities because the thermal vias are only oriented vertically in these experiments. During each iteration, the new vertical thermal conductivity is used to calculate the thermal via density, *m*, and the lateral thermal conductivities for each thermal via region. The effective lateral thermal conductivity can be found using the percentage of thermal vias, *m*:

$$K_x^{eff} = K_y^{eff} = \left(1-\sqrt{m}\right)K_{lateral}^{layer} + \frac{\sqrt{m}}{\frac{1-\sqrt{m}}{K_{lateral}^{layer}} + \frac{\sqrt{m}}{K_{via}}} \tag{119}$$

Table 21. Thermal Conductivities of Thermal Vias Regions

	Layer			Interlayer			Chip Average		
	Thermal Conductivity (W·m^{-1}·K^{-1})		Percent Thermal Via	Thermal Conductivity (W·m^{-1}·K^{-1})		Percent Thermal Via	Thermal Conductivity (W·m^{-1}·K^{-1})		Percent Thermal Via
	Vertical	Lateral		Vertical	Lateral		Vertical	Lateral	
Minimum	1.11	2.15	0%	1.10	1.10	0%	1.11	2.06	0%
Midrange	100.33	3.21	25%	50.71	1.31	12.5%	96.13	3.05	23.9%
Maximum	199.55	5.75	50%	100.33	1.65	25%	191.14	5.40	47.9%

Figure 65 and Table 21 show the relationship between the thermal vias densities and the thermal conductivities in the vertical and lateral directions for thermal via regions having vertically oriented thermal vias. In Figure 65, Equations (117) and (119) were used to plot the relationship between the percentage of thermal vias in a thermal via region and its effective thermal conductivities (in units of W·m^{-1}·K^{-1}). It can be seen that the vertically oriented vias (as shown in Figure 63) produce a much greater effect in the vertical thermal conductivity. In Table 21, the minimum, midrange, and maximum thermal conductivity values are given. In the midrange case, the average of the minimum and the maximum thermal via density values of each thermal via region was used.

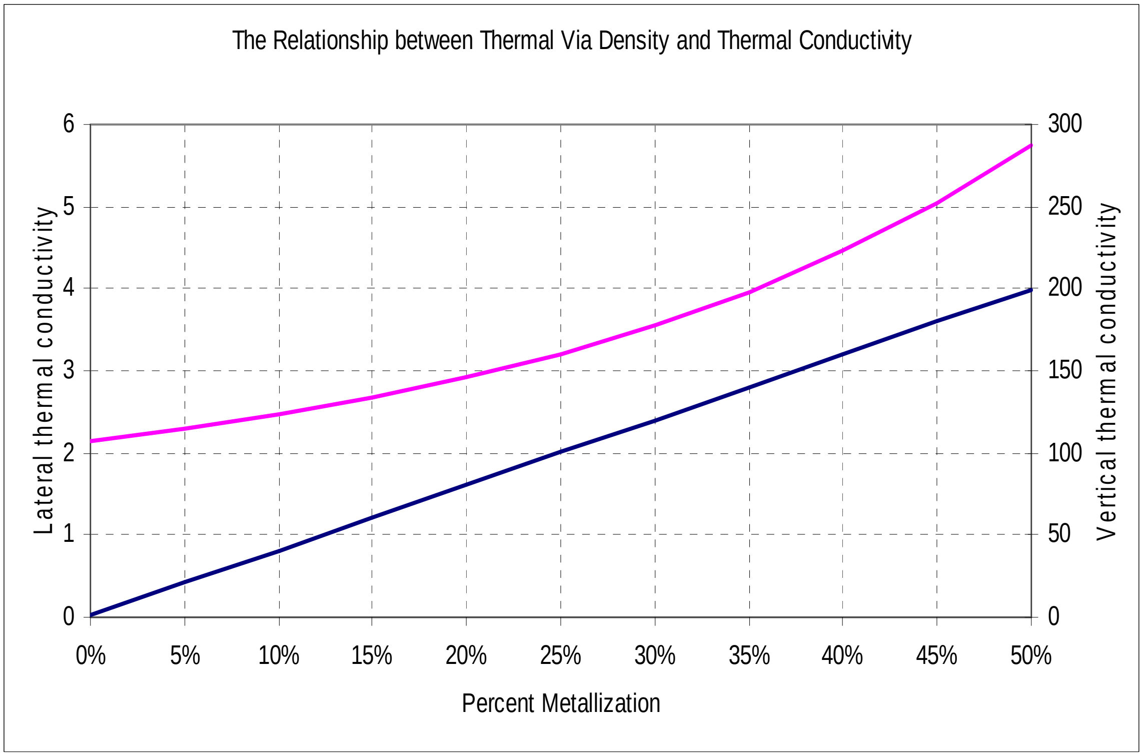

Figure 65. Percentage of thermal vias vs. thermal conductivity.

7.5 Implementation

In the thermal via placement method shown in Figure 66, the thermal gradients are used to update the thermal conductivities of the thermal via regions. In the process, the thermal gradients are compared to an ideal thermal gradient value which is modified during each iteration so that a certain *objective* is reached. In our implementation, the desired *objective* value for one of six different objective types is used by the algorithm: maximum thermal gradient, average thermal gradient, maximum temperature, average temperature, maximum thermal via density, or average thermal via density. Based on the objective type, the ideal thermal gradient is updated accordingly during each iteration.

```
THERMAL_VIA_PLACEMENT(objective) {
     SET $g_{ideal}$ TO $g_{ave}$
     SET $K$’s TO MININUM
     CALCULATE TEMPERATURE PROFILE
     WHILE NOT CONVERGED {
          FOR EACH THERMAL VIA REGION {
               $K_z = K_z \ (|g_z|/g_{ideal})^{1-\alpha}$
               UPDATE $K_{lateral}$
          }
          CALCULATE TEMPERATURE PROFILE
          UPDATE $g_{ideal}$ USING objective
     }
}
```

Figure 66. Pseudocode of the thermal via placement algorithm.

The thermal via placement algorithm is initialized by setting the ideal thermal gradient, g_{ideal}, to the average thermal gradient, g_{ave}, obtained in the midrange case. In addition, all thermal conductivities (*K*’s) are set to their minimum values, and an initial temperature profile is calculated before entering the main loop. Temperature profiles are calculated using FEA, as described in Section 3, and this gives the temperatures of the nodes and thermal gradients of the elements.

During each iteration of the main loop, the thermal conductivities of the thermal via regions are modified, and the temperature profile of the chip is recalculated using the new thermal conductivities. For each thermal via region, the vertical thermal gradient, g_z, at the center of the element is calculated using Equation (100). Using the magnitude of g_z, a new vertical thermal conductivity, K_z, is calculated using Equation (109) (α is set to 0.5 in these experiments). If the new thermal conductivity exceeds the minimum or maximum value for that element, it is set to the value of the bound that it exceeds. Using the new vertical thermal conductivity, the thermal via density and lateral thermal conductivities, $K_{lateral}$, are also updated using Equations (118) and (119).

Using the new temperature profile of the chip, the ideal thermal gradient, g_{ideal}, is modifying with the desired *objective* value using Equation (110), (111), (112), (113), (114), or (115) depending on which objective type is used. The algorithm terminates after the 1-norm of the change in the *K*'s is less than some small $\varepsilon > 0$ and percent difference between the current value and desired *objective* value is also within ε.

In choosing a desired *objective* value to be given to the algorithm, it must be between the values obtained when all the *K*'s are minimized and all the *K*'s are maximized. The algorithm simply finds the configuration of thermal vias between these two extremes that gives the desired *objective* value. For example, if an ideal maximum temperature, T_{max}^{ideal}, for the chip is desired, it must be less than the maximum temperature obtained when all the *K*'s are minimized and greater than the maximum temperature obtained when all the *K*'s are maximized in order for this maximum temperature to be realizable. In these experiments, the desired *objective* values were used from the case where all the thermal via regions were given the midrange values as shown in Table 21. This gives thermal gradient and temperature values that are greatly reduced from the case when no thermal vias were used and provides a comparison to the case where all the thermal via regions are given the same via density. In practice, it would be useful to use some maximum allowable value of the design for the desired *objective* value. For example, a maximum allowable operating temperature of the chip could be used for T_{max}^{ideal} in order to determine minimum amount of thermal vias and configuration needed to achieve this maximum temperature.

7.6 Results

The algorithm for thermal via placement was implemented as a computer program, written in C++ and run on a Linux workstation with a Pentium 4 3.2GHz CPU and 2GB memory. The conjugate gradient solver with ILU factorization preconditioning from the LASPack package [106] was used in our program to solve the FEA systems of equations. The thermal via placement method was tested using benchmark circuits (as shown in Table 22) from the MCNC suite [107] and the IBM-PLACE benchmarks [86] that were previously placed using a 3D placement tool [27].

Table 22. Benchmark Circuits

name	cells	nets
struct	1888	1921
biomed	6417	5743
ibm01	12282	11754
ibm04	26633	26451
ibm09	51746	50679
ibm13	81508	84297
ibm15	158244	161580

Dimensions used in this thesis for the 3D ICs were based on the design specifications for MIT Lincoln Labs' 0.18μm 3D FD-SOI technology [24] [89] [90]. However, we simulated the future progression of 3D IC technology by using four layers, copper interconnects, and higher interlayer via densities. In order to accommodate multiple layers, only face-to-back bonding was used to orient layers [26]. With face-to-back bonding, adjacent wafers are bonded together with the device layer of one wafer being adjacent to the top metal layer of another. In addition, power usage is increased to simulate the projected power densities at a more advanced technology node (~65nm).

The bulk substrate was given a thickness of 500μm, layers were given thicknesses of 5.7μm, and interlayers were given thicknesses of 0.7μm. Four layers were used, and the chip size was set at 2cm × 2cm with the cell sizes adjusted accordingly. Thermal via regions were even distributed between the rows and given 10% of the total chip area. The thermal conductivity of the silicon in the bulk substrate was set to 150W/mK, and the thermal vias were assumed to be copper with a thermal conductivity of 398W/mK. The thermal conductivities used in the layer and interlayer elements are shown in Table 21. Thermal via regions had variable thermal conductivities ranging from the minimum to maximum values given in Table 21. All other elements used the thermal conductivities corresponding to the lateral midrange values.

The power densities used in these experiments were derived from [108] [109] [110] and reflect the power usage at approximately the 65nm node. Power densities are becoming more unevenly distributed in modern microprocessors with the power densities of more active areas being much greater than that of less active blocks [109] [110]. Therefore, an asymmetrical power distribution was used to account for this and accurately simulate the hot spots and large thermal gradients that can occur in these circuits. Because the switching activity and power dissipation information is not available for the benchmark circuits, a random power distribution was used with 90% of the cells given power densities ranging from 0 to 2 x 10^6 W/m^2 and 10% of the cells given power densities ranging from 2 x 10^6 to 4 x 10^6 W/m^2. However, if more information about the power dissipation of the cells is known, it can easily be used instead of a random power distribution. In these experiments, the bottom of the chip was made isothermic with the ambient temperature to represent the heat sink, and the top and sides of the chip were made insulated in order to simulate the low heat sinking properties of the packaging. If desired, a more sophisticated thermal model for the heat sink and packaging could be used instead. The ambient temperature was set to 0^oC for convenience, but the temperatures can be translated by the amount of any other ambient as desired.

For the benchmark circuits, FEA meshes were produced in which the number of elements increases linearly with the number of cells in the circuit. Because the internal structures of the cells are not provided in the benchmark data and because the specific interconnect structures can not be obtained until after routing, the FEA elements were given cell-level granularity. In addition, detailed power profiles of the cells are also not provided in the benchmark data so single, uniform heat sources were used to represent the cells in the thermal mesh, and the granularity of the element sizes was commensurate with this. Because these structural details are not available, a finer granularity of element sizes would not significantly improve accuracy and would unnecessarily increase run times. As see in Figure 63, the heights of the row and inter-row regions corresponding to the heights of the elements. Likewise, the depths of the layer and interlayer regions correspond to the depths of the elements, and the widths of the elements were made similar to the element heights.

7.6.1 Using Uniform Thermal Via Densities

In the first set of experiments, thermal via densities were increased from their minimum to maximum values, and the change in the temperatures and thermal gradients was observed as shown in Table 23. In this table, T_{ave} is the average temperature, T_{max} is the maximum temperature, g_{ave} is the average thermal gradient, and g_{max} is the maximum thermal gradient. The units used in this table and the proceeding tables are in ^{o}C for the temperatures and K/m for the thermal gradients. The thermal via regions were all assigned the same minimum, midrange, and maximum thermal conductivity values from Table 21. The minimum values correspond to the case where no thermal vias are used, the maximum values correspond to maximum thermal via usage, and in the midrange case, thermal via regions were assigned thermal via densities that were the average of the minimum and maximum values. In Table 23, we can see that as the thermal via densities of the thermal via regions are increased, the temperatures and thermal gradients decrease. The temperatures and thermal gradients in the minimum and maximum cases define the bounds on the thermal values that can be obtained from adjusting the thermal via densities. At the midrange with an average thermal via density of 23.9% in the thermal via regions, the maximum temperatures were 47.3% lower and the average temperatures were 28.3% lower than in the case where no thermal vias were used.

Table 23. Thermal Properties with a Uniform Distribution of Thermal Via Densities

Benchmark Circuit	Thermal Via Densities of Thermal Vias Regions											
	Minimum (0%)				Midrange (23.9%)				Maximum (47.9%)			
name	T_{ave} (^{o}C)	T_{max} (^{o}C)	g_{ave} (K/m)	g_{max} (K/m)	T_{ave} (^{o}C)	T_{max} (^{o}C)	g_{ave} (K/m)	g_{max} (K/m)	T_{ave} (^{o}C)	T_{max} (^{o}C)	g_{ave} (K/m)	g_{max} (K/m)
struct	15.4	58.9	1.86E+5	2.07E+6	10.9	35.0	5.67E+4	6.14E+5	10.4	31.3	4.08E+4	3.87E+5
biomed	14.0	62.2	1.67E+5	2.25E+6	10.0	32.0	4.92E+4	6.29E+5	9.5	26.1	3.35E+4	4.03E+5
ibm01	14.2	45.1	1.71E+5	1.64E+6	10.1	26.2	4.96E+4	5.92E+5	9.6	22.7	3.36E+4	3.77E+5
ibm04	13.5	54.0	1.51E+5	2.00E+6	10.0	26.5	4.42E+4	4.71E+5	9.6	21.4	2.98E+4	2.94E+5
ibm09	13.8	53.0	1.56E+5	1.46E+6	10.2	26.8	4.55E+4	5.32E+5	9.8	21.4	3.04E+4	3.41E+5
ibm13	14.6	47.3	1.84E+5	1.88E+6	10.3	23.6	5.34E+4	6.53E+5	9.7	19.3	3.55E+4	4.25E+5
ibm15	15.1	52.8	2.01E+5	2.00E+6	10.5	26.5	5.78E+4	6.97E+5	9.9	20.6	3.83E+4	4.54E+5

The previous set of experiments demonstrate the baseline thermal improvements that can be made by using a simple thermal via placement scheme where thermal via regions are given a uniform distribution of thermal via densities. However, with the more sophisticated thermal via placement method from Figure 66, larger thermal improvements can be made with fewer thermal vias and a nonuniform distribution of thermal via densities. As will be seen in the following experiments, the thermal via placements that are generated by this algorithm lie along a continuous curve of optimized thermal via placements between the minimum and maximum cases of Table 23. This stems from the fact that the algorithm converges at a specific value for the internal variable, g_{ideal}, and produces a thermal via placement that corresponds to it. This algorithm finds the point, representing a specific thermal via placement, along this curve that satisfies the desired thermal objective.

Six objective types were examined in the following experiments: maximum thermal gradient, average thermal gradient, maximum temperature, average temperature, maximum thermal via density, and average thermal via density. For each of these experiments, the values obtained in the midrange case from Table 23 were used as the desired objective values for the algorithm. These values are used only to illustrate the use and effectiveness of this thermal via placement method, and any other values can be used as the objective as long as they are between the values obtained at the minimum and maximum cases.

7.6.2 Thermal Gradient Objectives

In Figure 67 the average and maximum thermal gradients were plotted against the thermal via densities for thermal via placements obtained using our method and a uniform distribution of thermal vias for the benchmark circuit struct. The solid curves represent the values obtained using our thermal via placement method, and as can be seen, these curves are significantly better than the dashed curves obtained using a uniform distribution of thermal via densities.

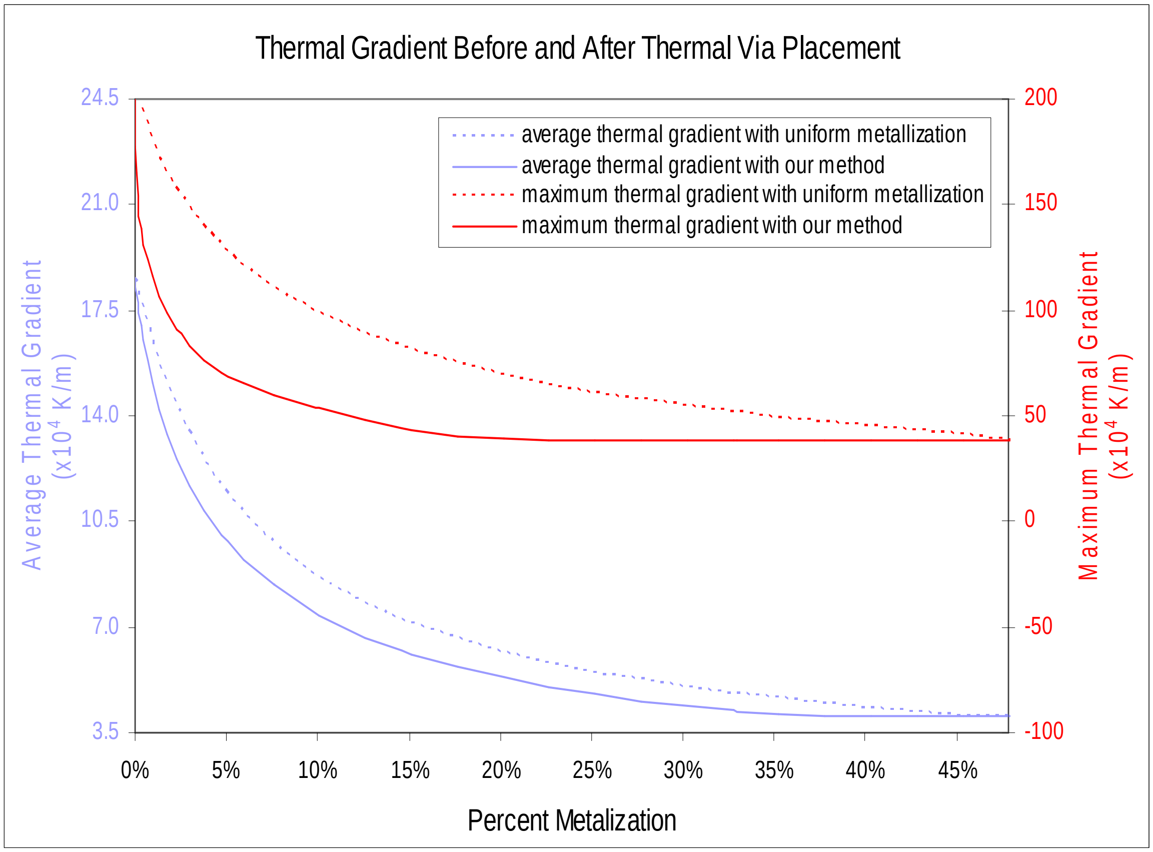

Figure 67. Thermal gradient optimization curves for struct.

In Table 24, thermal via placements were obtained using the maximum thermal gradients from the midrange case (in Table 23) as the objectives. In this table, K_{ave} is the average vertical thermal conductivity of the thermal via regions, and m_{ave} is the average density of thermal vias in the thermal via regions. On average, thermal via placements had a thermal via density of only 10.2% in the thermal via regions in order to obtain the same maximum thermal gradients as in the midrange case. This means that 57.3% fewer thermal vias were needed with thermal via placement than in the midrange case to obtain the same maximum thermal gradients. These maximum thermal gradients were 68.1% lower than in the case where no thermal vias were used. In these experiments, thermal via regions were assigned to only 10% of the total chip area, and thermal vias require only 10.2% of this area to satisfy this objective so thermal vias would occupy only 1.02% of the total chip area. With this small amount of blockages, it is expected that routability would be minimally affected.

Table 24. Optimization to Maximum Thermal Gradient

Circuits	K_{ave} (W·m^{-1}·K^{-1})	m_{ave}	T_{ave} (^{o}C)	T_{max} (^{o}C)	g_{ave} (K/m)	g_{max} (K/m)
struct	29.3	7.1%	11.6	36.1	8.61E+04	6.14E+05
biomed	44.7	11.0%	10.2	33.1	6.22E+04	6.29E+05
ibm01	29.6	7.2%	10.8	30.7	7.71E+04	5.92E+05
ibm04	52.1	12.8%	10.1	26.4	5.14E+04	4.71E+05
ibm09	39.7	9.7%	10.5	29.0	6.15E+04	5.32E+05
ibm13	42.3	10.4%	10.6	26.0	7.03E+04	6.53E+05
ibm15	54.3	13.4%	10.6	25.1	6.80E+04	6.97E+05

In Table 25, thermal via placements were obtained using the average thermal gradients from the midrange case as the objectives. The thermal via placement method used an average thermal via density of 17.6% in the thermal via regions instead of 23.9%. These thermal via placements gave lower reductions in the thermal via densities than with the maximum thermal gradient objectives because there is less room for improvement with the average thermal gradient values as can seen in Figure 67 by the smaller gap between the curves.

Table 25. Optimization to Average Thermal Gradient

Circuits	K_{ave} (W·m^{-1}·K^{-1})	m_{ave}	T_{ave} (^{o}C)	T_{max} (^{o}C)	g_{ave} (K/m)	g_{max} (K/m)
struct	71.1	17.6%	10.7	32.3	5.67E+04	4.01E+05
biomed	70.5	17.5%	9.8	29.3	4.92E+04	5.13E+05
ibm01	71.2	17.7%	10.0	24.9	4.96E+04	3.88E+05
ibm04	69.3	17.2%	9.9	24.4	4.42E+04	4.23E+05
ibm09	70.6	17.5%	10.1	24.4	4.55E+04	3.96E+05
ibm13	70.8	17.6%	10.1	22.4	5.34E+04	4.92E+05
ibm15	72.9	18.1%	10.3	22.5	5.78E+04	5.92E+05

7.6.3 Temperature Objectives

In Figure 68, the average and maximum temperatures were plotted against the thermal via densities for thermal via placements obtained using our method and the simple method with uniform thermal via densities. The solid curves represent the temperatures obtained using our thermal via placement method and are significantly better than the dashed curves obtained using a uniform distribution of thermal via densities.

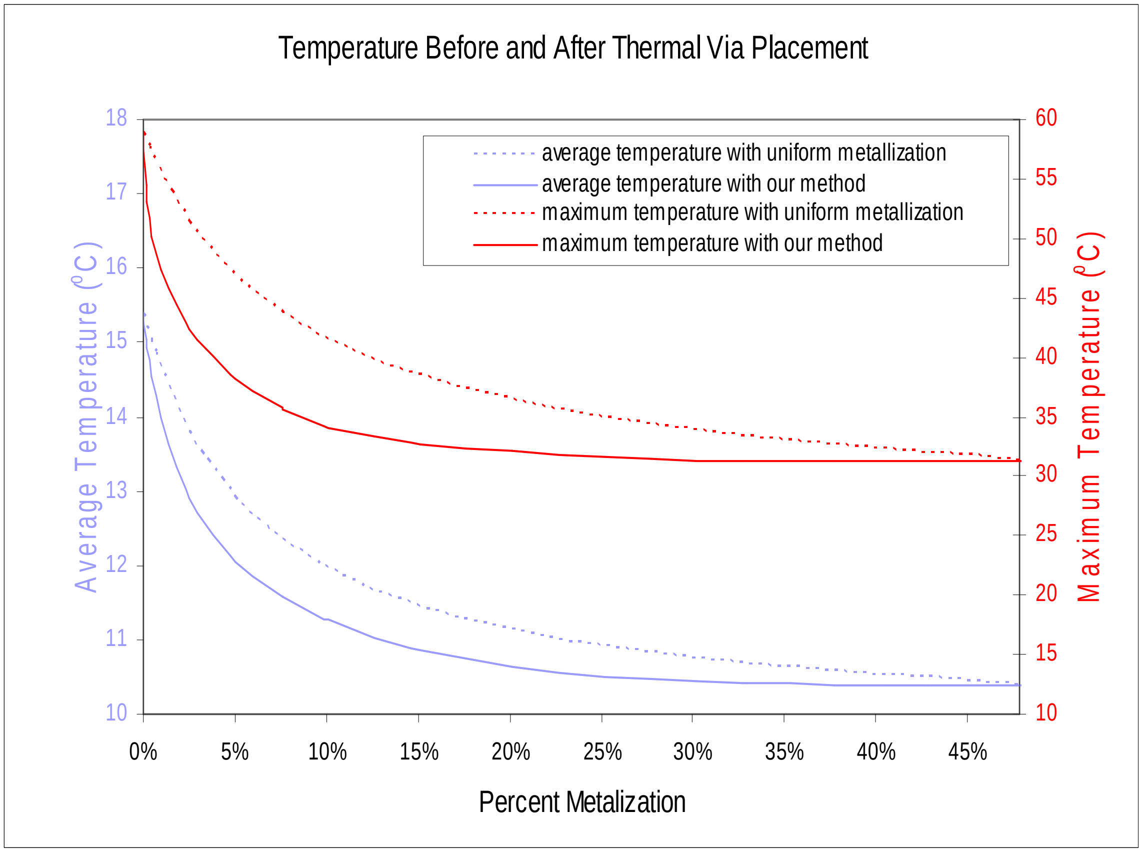

Figure 68. Temperature optimization curves for struct.

In Table 26, thermal via placements were obtained using the maximum temperatures from the midrange case as the objectives. On average, the thermal via placements used only a thermal via density of 12.3% in the thermal via regions in order to obtain maximum temperatures that are 47.3% lower than in the minimum case. 48.5% fewer thermal vias were needed than in the midrange case to obtain the same maximum temperatures. With only 10% of the chip area assigned to thermal via regions, thermal vias would occupy only 1.23% of the total chip area.

Table 26. Optimization to Maximum Temperature

Circuits	K_{ave} ($W \cdot m^{-1} \cdot K^{-1}$)	m_{ave}	T_{ave} (^oC)	T_{max} (^oC)	g_{ave} (K/m)	g_{max} (K/m)
struct	34.9	8.5%	11.4	35.0	7.97E+04	5.74E+05
biomed	50.1	12.3%	10.1	32.0	5.88E+04	5.94E+05
ibm01	57.1	14.1%	10.1	26.2	5.58E+04	4.20E+05
ibm04	51.6	12.7%	10.1	26.5	5.16E+04	4.72E+05
ibm09	51.1	12.6%	10.3	26.8	5.40E+04	4.72E+05
ibm13	59.8	14.8%	10.3	23.6	5.85E+04	5.43E+05
ibm15	46.0	11.3%	10.8	26.5	7.44E+04	7.55E+05

The average temperatures from the midrange case were used as the desired objective values for the results in Table 27. With this objective, an average thermal via density of 14.3% was needed. Similar to the average thermal gradient curves, the gap between the average temperature curves in Figure 68 show that little improvement can be expected with the average temperature.

Table 27. Optimization to Average Temperature

Circuits	K_{ave} ($W \cdot m^{-1} \cdot K^{-1}$)	m_{ave}	T_{ave} (^oC)	T_{max} (^oC)	g_{ave} (K/m)	g_{max} (K/m)
struct	57.9	14.3%	10.9	32.9	6.27E+04	4.45E+05
biomed	57.0	14.1%	10.0	30.9	5.50E+04	5.64E+05
ibm01	58.0	14.3%	10.1	26.1	5.53E+04	4.16E+05
ibm04	57.2	14.1%	10.0	25.7	4.89E+04	4.55E+05
ibm09	58.2	14.4%	10.2	25.7	5.04E+04	4.41E+05
ibm13	57.7	14.3%	10.3	23.8	5.97E+04	5.54E+05
ibm15	59.1	14.6%	10.5	24.3	6.50E+04	6.67E+05

7.6.4 Thermal Via Density Objectives

The maximum thermal via densities produced using our method and the uniform thermal via density method are plotted in Figure 69. The solid curve was obtained using our thermal via placement method, and it rapidly increases and plateaus at the maximum value as the average thermal via density increases. The dashed curve was obtained using a uniform distribution of thermal via densities and increases linearly as a result.

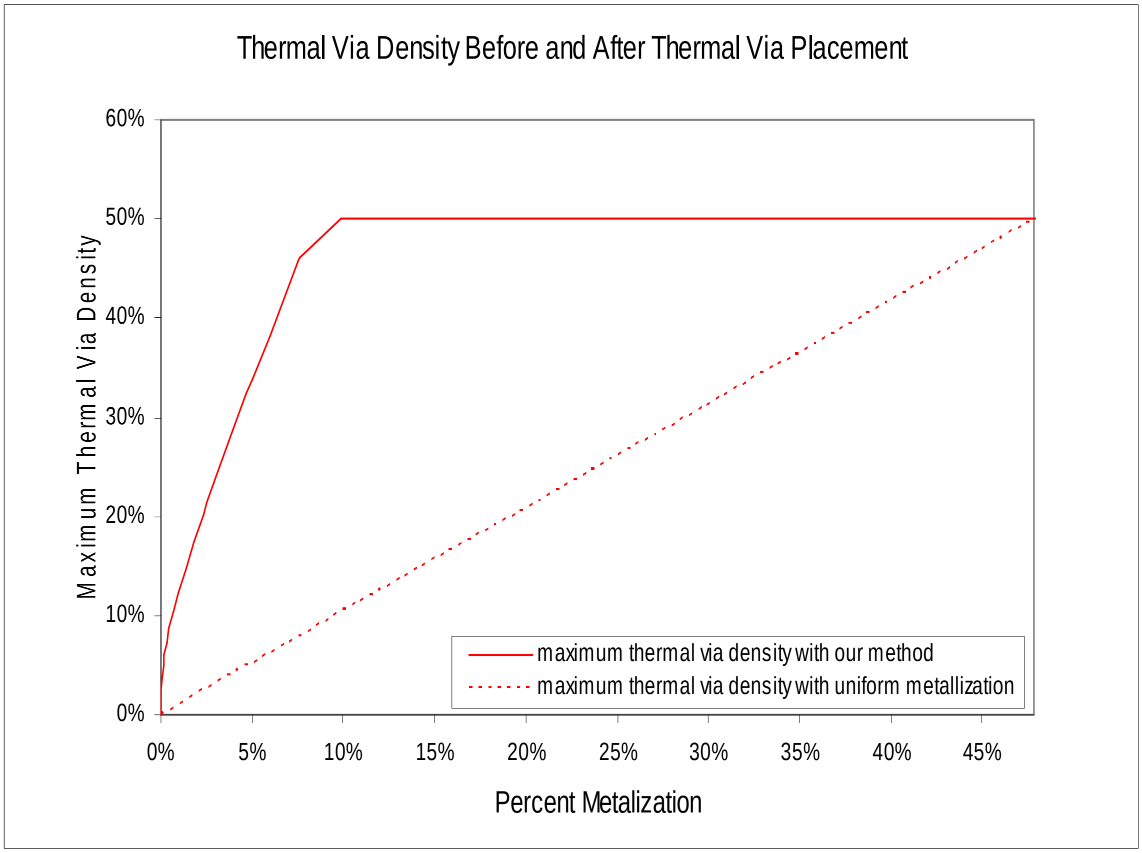

Figure 69. Maximum thermal via densities for struct.

Table 28. Optimization to Maximum Thermal Via Density

Circuits	m_{ave}	m_{max}	T_{ave} (°C)	T_{max} (°C)	g_{ave} (K/m)	g_{max} (K/m)
struct	3.3%	25.0%	12.6	40.9	1.14E+05	8.06E+05
biomed	3.3%	25.0%	11.5	44.8	1.01E+05	1.02E+06
ibm01	5.2%	25.0%	11.2	32.8	8.82E+04	6.85E+05
ibm04	4.1%	25.0%	11.2	36.2	8.61E+04	7.18E+05
ibm09	5.2%	25.0%	11.2	34.6	8.19E+04	6.89E+05
ibm13	4.4%	25.0%	11.6	31.4	1.02E+05	9.26E+05
ibm15	4.0%	25.0%	12.0	35.6	1.17E+05	1.07E+06

Table 28 shows the thermal via placements obtained using a maximum thermal via density of 25% (from the midrange case) as the objective. The average thermal via densities obtained for this objective were much lower and had an average of 4.2%. The other thermal properties were not as good as in the midrange case but were much better than the minimum case where no thermal vias were used. With the value of 25% for the maximum thermal via density, significant thermal improvements can be made with very little thermal via utilization. At this point on the optimization curve, thermal via regions use only 4.2% of its area, but the maximum thermal gradient is reduced by 55.5% and the maximum temperature is reduced by 31.4% as compared to the case where no thermal vias are present. This objective could also be used to ensure that no thermal via regions use more thermal vias than some specified amount that is lower than the actual maximum possible utilization.

The average thermal via density of 23.9%, same as the midrange case, was used as the desired objective value for Table 29. With the same average thermal via density, the thermal properties are improved considerably over the thermal via placements with uniform thermal via densities. The results from this will be more clearly summarized in the next section.

Table 29. Optimization to Average Thermal Via Density

Circuits	K_{ave} ($W \cdot m^{-1} \cdot K^{-1}$)	m_{ave}	T_{ave} (^oC)	T_{max} (^oC)	g_{ave} (K/m)	g_{max} (K/m)
struct	96.1	23.9%	10.5	31.7	4.89E+04	3.87E+05
biomed	96.1	23.9%	9.6	27.3	4.13E+04	4.08E+05
ibm01	96.1	23.9%	9.8	23.4	4.19E+04	3.77E+05
ibm04	96.1	23.9%	9.7	22.2	3.67E+04	3.34E+05
ibm09	96.1	23.9%	9.9	22.5	3.81E+04	3.41E+05
ibm13	96.1	23.9%	9.9	20.6	4.46E+04	4.25E+05
ibm15	96.1	23.9%	10.0	21.1	4.91E+04	4.66E+05

7.6.5 Comparing Different Objectives

A number of observations can be obtained from the optimization curves shown in Figure 67, Figure 68, and Figure 69. Not only can the absolute improvement of thermal via placements be compared with the minimum case where no thermal vias are present, but also the relative improvement can be seen against the simple method with uniform thermal via densities. The thermal improvements can be observed at any average thermal via density by comparing the curves vertically. In addition, the reduction in the average thermal via densities can observed for any particular thermal objective by comparing the curves horizontally.

Table 30. Summary of Results for Different Objectives

Objective	Average percent change from the midrange case					
	g_{max}	g_{ave}	T_{max}	T_{ave}	m_{max}	m_{ave}
g_{max}	0.0%	33.5%	5.2%	3.3%	79.6%	-57.3%
g_{ave}	-23.3%	0.0%	-8.3%	-1.7%	100.0%	-26.5%
T_{max}	-8.6%	20.9%	0.0%	1.4%	100.0%	-48.5%
T_{ave}	-15.3%	11.3%	-3.5%	0.0%	100.0%	-40.3%
m_{max}	40.9%	93.3%	30.7%	12.7%	0.0%	-82.4%
m_{ave}	-34.5%	-15.7%	-14.3%	-3.7%	100.0%	0.0%

The results for these six objective types are summarized in Table 30. The average percent differences between the thermal via placement and the midrange case values are given in this table. The thermal via placement method was very accurate in achieved the desired objective values as can be seen by the zero percents (< 0.05%) in the diagonal. This is not surprising since ε was set to 0.001 in these experiments. When given the same average thermal via density, the maximum thermal gradient is reduced by 34.5% and the maximum temperatures are reduced by 14.3%. With the maximum thermal gradient objective, temperature values are increased only slightly, but the thermal via densities are reduced greatly having an average reduction of 57.3%. The percent difference between the values obtained with this method and with no thermal vias is shown in Table 31. With each case, thermal properties are greatly improved at the expense of larger thermal via densities.

Table 31. The Results compared to the Minimum Case

Objective	Average percent change from the minimum case					
	g_{max}	g_{ave}	T_{max}	T_{ave}	m_{max}	m_{ave}
g_{max}	-68.1%	-60.8%	-44.5%	-25.9%	44.9%	10.2%
g_{ave}	-75.7%	-70.7%	-51.6%	-29.5%	50.0%	17.6%
T_{max}	-71.1%	-64.5%	-47.3%	-27.3%	50.0%	12.3%
T_{ave}	-73.2%	-67.4%	-49.2%	-28.3%	50.0%	14.3%
m_{max}	-55.5%	-43.3%	-31.4%	-19.2%	25.0%	4.2%
m_{ave}	-79.2%	-75.3%	-54.7%	-31.0%	50.0%	23.9%

7.6.6 Run Time

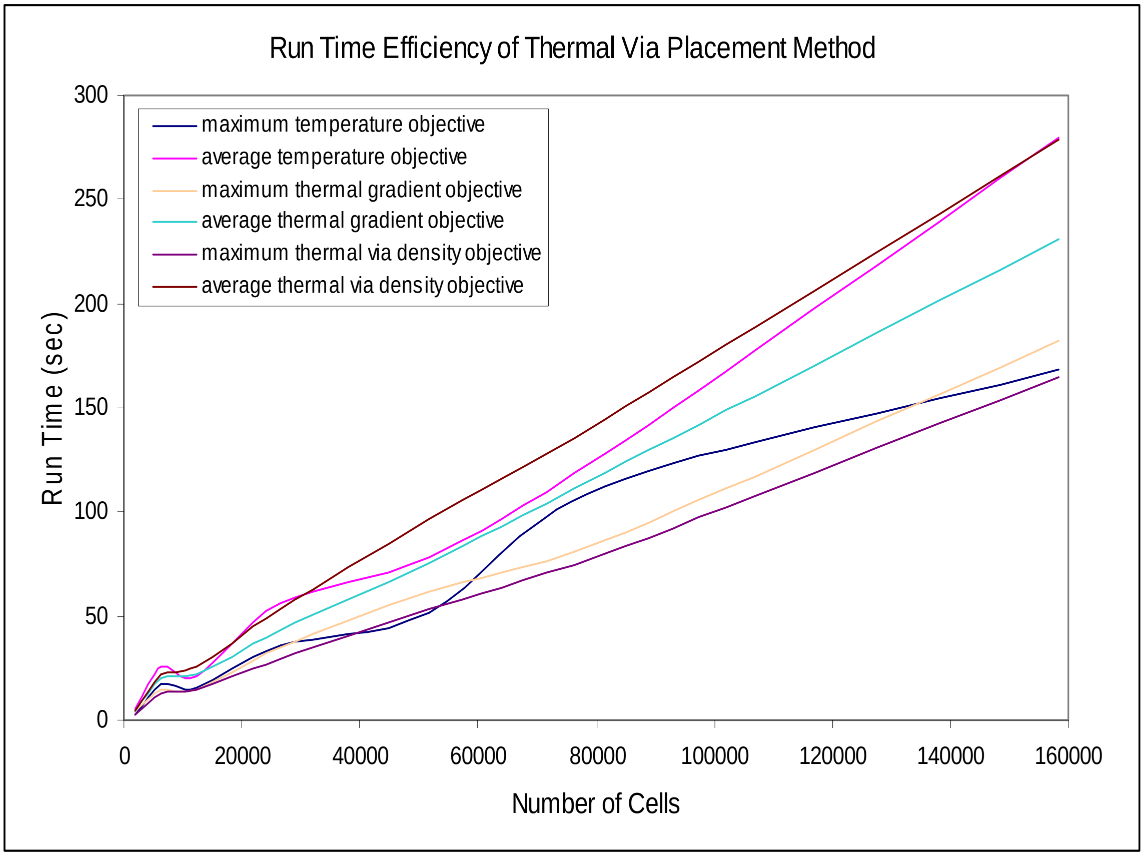

Figure 70. Run Time Efficiency of our Method.

The run time efficiency of the thermal via placement algorithm was also examined as shown in Figure 70. In this figure we see that the thermal via placement method has linear time efficiency across a wide range of circuit sizes for all six thermal objectives used. This is achieved by using an efficient thermal solver and because the thermal via placement method convergences in roughly the same number of iterations.

7.6.7 Thermal Profile of Struct

The temperature profiles of the struct benchmark before and after thermal via placement with a maximum temperature objective are shown in Figure 71 and Figure 72 respectively. The resulting thermal via placement is shown in Figure 73. In these figures, distances are in meters, the x-axis is oriented along the rows, the y-axis goes across the rows, and the z-axis goes across the layers. In Figure 71 and Figure 72, the heat sink is located at the bottom of the chip, and temperature contours are superimposed on the standard cells. Black indicates areas of high temperature, and white represents areas of low temperature. As can be seen, temperatures increase toward the upper layers and middle of the chip and decrease near the heat sink. After thermal via placement as shown in Figure 72, the temperatures are greatly reduced.

In Figure 73, the thermal via regions after thermal via placement are shown as small squares arranged in a grid pattern. The percentage of thermal vias in the thermal via regions is represented by its color. A black square represents a thermal via region with a maximum number of thermal vias being utilized. A white square represents a thermal via region with no thermal vias being needed. The heat sink is located at the bottom, and the thermal via regions were of greatest strength at the bottom of the chip where the thermal gradients are the highest and the most impact can be made in reducing thermal problems. However, at the top of the chip where the temperatures are the highest, the thermal vias are minimally used. In these areas, the thermal gradients are quite low and little impact can be made there.

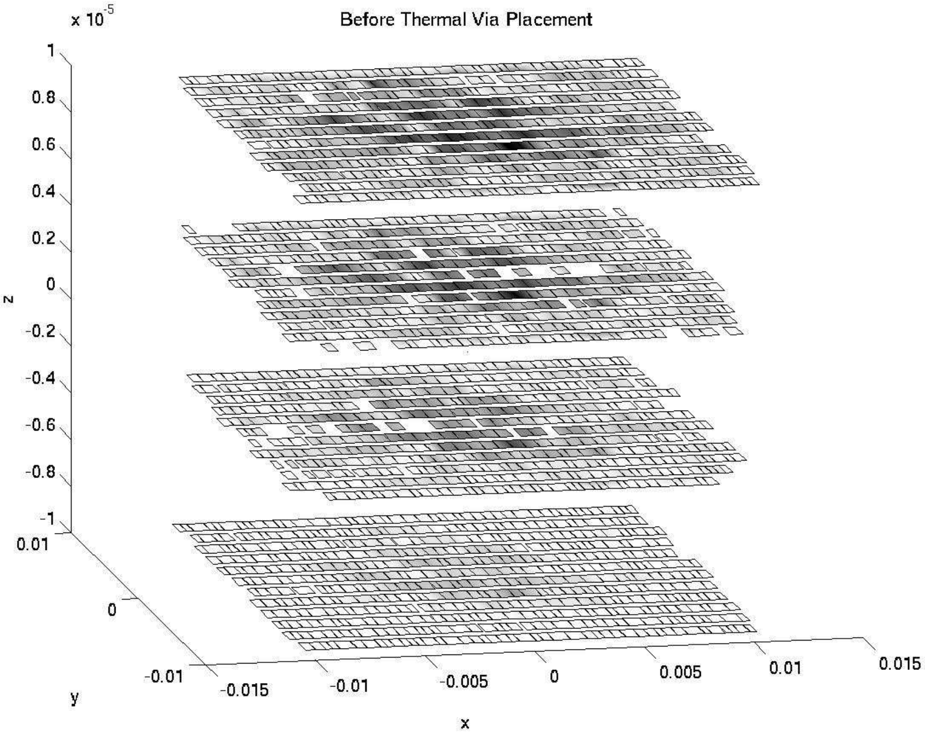

Figure 71. The thermal profile of struct before thermal via placement.

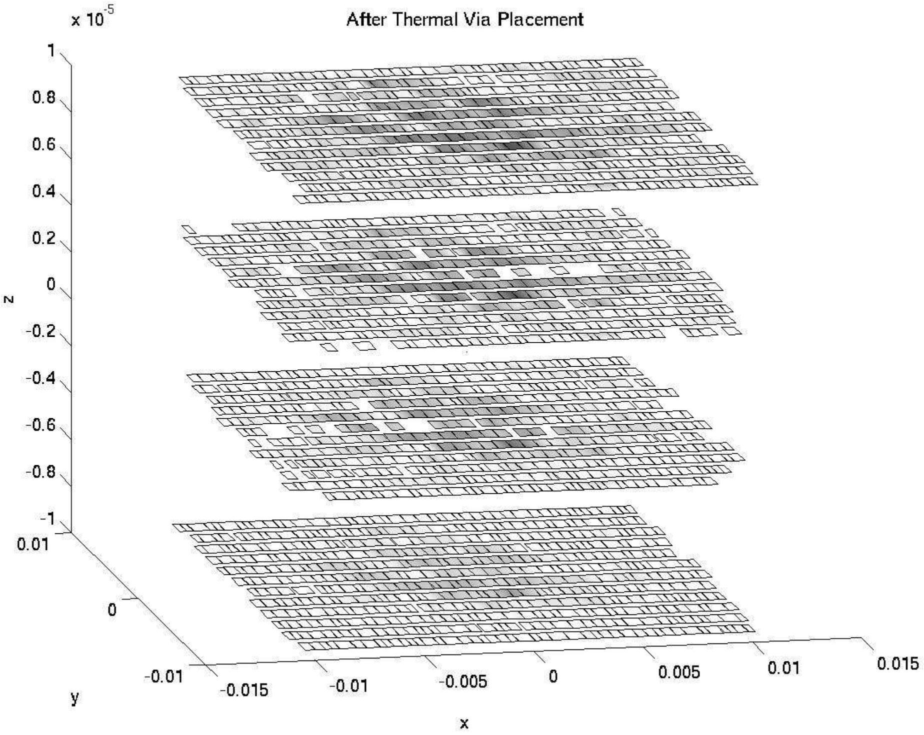

Figure 72. The thermal profile of struct after thermal via placement.

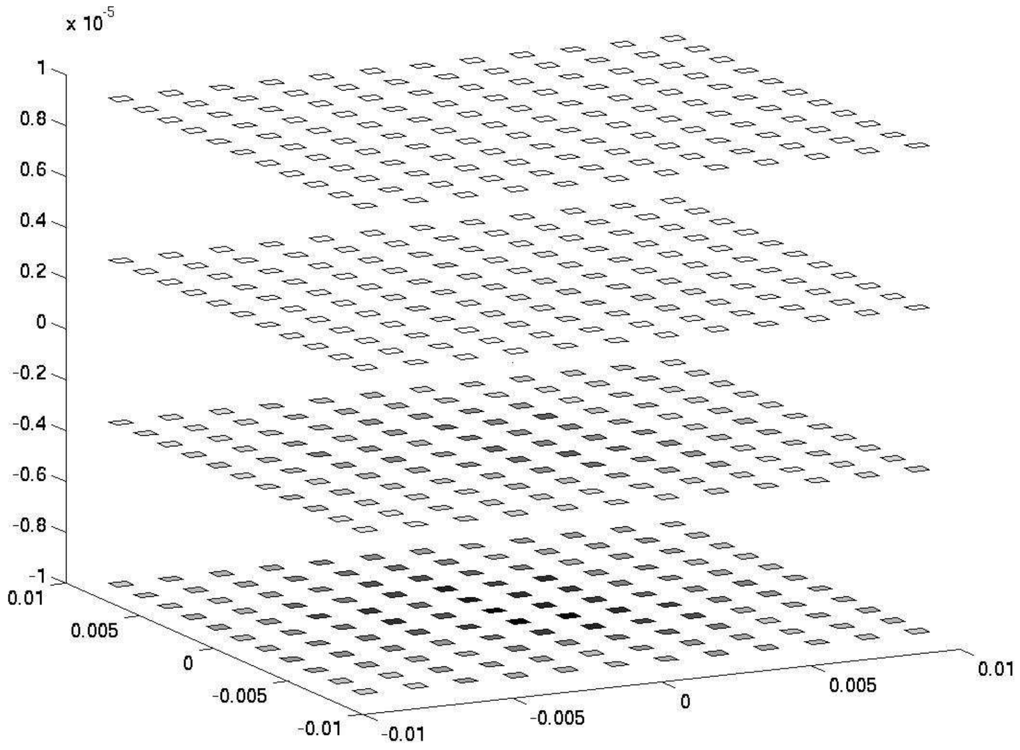

Figure 73. Thermal vias regions of struct after thermal via placement.

7.7 Conclusion

An efficient thermal via placement method was presented that attempts to overcome the thermal issues produced in the design of 3D ICs. The resulting thermal via placements have lower temperatures and thermal gradients with minimal use of thermal vias. The method is flexible enough to handle different thermal objectives such as obtaining a specific maximum temperature for the chip. As shown in Figure 67, Figure 68, and Figure 69, thermal via placements produced by this method lie on a continuous path from the minimum to maximum cases. These curves show significant improvement over using a uniform distribution of thermal via densities. In these experiments, the thermal via placement method used 48.5% fewer thermal vias to reach the same 47.3% reduction in the maximum temperatures that was obtained by given all the thermal via regions the same midrange thermal via densities.

The method makes iterative improvements to the thermal via placement until the desired objective value is reached. In the process, the thermal conductivities of the thermal via regions are modified in order to satisfy this objective. Each thermal conductivity corresponds to a particular percentage of thermal vias in the thermal via region. There is a tradeoff between thermal effects and thermal via densities as seen in Table 23. There is also a tradeoff between area used for routing and area used for thermal vias. Consequently, this produces a tradeoff between thermal problem reduction using thermal vias and routability. An important observation is that thermal vias placed in areas of high temperature, such as in the upper-most layer, have little impact in reducing thermal problems. This algorithm places thermal vias where they will have the most impact using the thermal gradient as a guide. High temperatures can only be reduced by alleviating the high thermal gradients leading up to them. The thermal resistance of these heat conduction paths is reducing by lowering the thermal conductivities of elements along it.

8 Net Weighting to Reduce Repeater Counts during Placement

8.1 Introduction

With decreasing feature sizes, interconnect delays do not scale as well as gate delays. Consequently, they are becoming a dominant part of the total delay in deep submicron technologies, especially as overall chip areas do not shrink with scaling. Since the delay of an unbuffered wire grows quadratically with wirelength, repeaters are needed to bring this delay trend down to a linear one, as well as to restore signal slews. However, the inter-repeater separation unfortunately scales poorly (0.586x per generation, in contrast to the normal shrink factor of 0.7x) [111], leading to an explosion in the expected number of repeaters and a consequent breakdown of many of today's CAD algorithms and methodologies [112]. As repeaters become more prominent in future designs, they are predicted to cause a number of problems including increased power consumption and degraded design convergence.

As the number of repeaters increases, it becomes more difficult to integrate them into the design. Efforts are needed to keep their number minimized while satisfying traditional design constraints such as performance, power and area. It has been observed that the signal propagation speed on a long buffered wire does not vary appreciably over a significant range of inter-repeater distances [113]. This allows the inter-repeater distances to be increased slightly without compromising the performance significantly, thus allowing fewer repeaters to suffice. However, although this method can help reduce the number of repeaters on individual nets in a *placed* design (at the cost of slightly degraded delays and slews), it does not modify the placement taking the global view of the entire netlist into consideration. In this paper, we show that it is possible to reduce the overall repeater count even further during placement by trading off the lengths of nets that are just on the threshold of requiring (additional) repeaters, against those of less-critical nets that can afford to grow in length without needing more repeaters. This can be accomplished by the judicious application of context-sensitive net-weights to these nets. However, if it is to be practical, any proposed net-weight modification scheme must rely on only a small number (ideally, one) of parameters. Furthermore, it should be robust enough that its controlling parameter(s) requires no individual tuning for each net, or even for each testcase. Another complication in any such scheme is introduced by the fact that the number of repeaters required on a net is dependent on the layer assignment and routing of that net – information that is unknown at the time of the placement because of the unaffordable cost of invoking global routing within each placement iteration.

Net weighting has previously been used primarily in the context of timing-driven placement and low-power design, in order to reduce the lengths (and consequently, wire loads) of critical nets [114] [115] [116] and reduce the power consumption [92] [93] [117]. We use net weighting in a completely novel way, *viz*., to nudge nets away from repeater insertion and towards deletion thresholds. In these experiments, we modified industrial implementations of the Kraftwerk global placer [36] that incorporates native repeater modeling [118], as well as the force-directed-Mongrel (FD-Mongrel) coarse legalizer [72], in order to investigate the effects of net weighting on repeater count reduction during placement. The net weights are modified during each iteration in context-sensitive manner with layer assignment as well as valid inter-repeater distance ranges being modeled. As a result, repeater counts are significantly decreased with minimal impact on wirelength.

Being essentially a net weight modification scheme, our method is quite general in scope – as can be seen from its successful application to two different placement algorithms (*viz*. KraftWerk and FD-Mongrel) that is described in this chapter. In general, any placer that supports net weights can benefit from this scheme; this includes the vast majority of commercial and academic placers. The empirical layer prediction and repeater prediction models that are used in our experiments are not dependent on the choice of the specific placer used.

8.2 Placement Infrastructure

At the global placement stage, we apply our net-weighting scheme within the force-directed placement paradigm, using an implementation of Kraftwerk. Kraftwerk extends the traditional quadratic analytical model (in which one seeks to minimize the weighted sum of the squared Euclidean distances of connected cells, in analogy with finding the equilibrium for a system of springs as discussed in Section 4.2) by introducing a spreading force field. The forces in this spreading field are computed using the density profile of the cells in the design. Furthermore, our implementation leverages the “linearization” of the quadratic objective function [119] that usually results in improved solution quality.

This implementation of Kraftwerk has been augmented further with MorePlace [118], which is a scheme to model repeaters natively during analytical placement. This scheme allows the system to avoid the massive perturbations caused when the large numbers of repeaters required at future process technologies are patched directly into the netlist in an interleaved or iterated fashion during global placement [118]. In MorePlace, virtual repeaters are added and deleted as needed during each iteration of Kraftwerk, contributing repulsive and/or attractive forces (in addition to the usual density-derived forces for spreading) without fragmenting the original netlist. Not only do virtual repeaters contribute to the spreading forces but also provide attractive nets with quadratic costs that cause them to spread out equidistantly along their nets. The virtual repeater insertion and deletion is done at the beginning of each Kraftwerk iteration, prior to the calculation of the new cell positions (including the virtual repeaters), using a length-based repeater prediction scheme [120] [118]. If the average inter-repeater distance along a net is below a minimum value, repeaters are deleted from the net. On the other hand, if the average inter-repeater distance is greater than a maximum value, one or more repeaters are added to the net. The difference between these maximum and minimum values captures the range of tolerable inter-repeater distances [113]. Finally, the surviving virtual repeaters are instantiated after the placement terminates; the repeater force model helps ensure that sufficient space is available to do so.

After global placement with Kraftwerk/MorePlace, we apply our net weighting scheme to FD-Mongrel [68] that is used for coarse-grained legalization. FD-Mongrel uses a hybrid approach that maintains the quality of the force-directed placement while making significant improvements on overlap removal. It begins with a coarse grid approach that uses forces to remove overlaps globally. In the second phase, detailed placement is performed using a fine grid in which cells are ripple-moved from the densest bin to the least dense bin following a monotonic path of least resistance, i.e., the path causing the least amount of quality degradation. Ripple-moves that result in wirelength constraint violations are avoided. The cost (or gain) computed for each potential move using the wirelength and net constraints is used to determine which cells to move during the detailed placement phase of FD-Mongrel. Finally, the almost-legal placement from the coarse legalization is passed onto a fine-grained legalizer.

8.3 Proposed Approach

In our approach, net weighting is used at the global placement and coarse legalization stages to reduce the number of repeaters needed in the placement. Assume, for the purpose of illustration, that we introduce a repeater on a wire every *x* microns because of signal slew constraints. Then, if we have two approximately equi-critical wires of lengths 1.05*x* and 1.5*x* respectively, a simple length-based repeater insertion engine will add a repeater to each of them. However, if we shrink the length of the first one marginally to 0.95*x*, even if it means allowing the other net to grow to a length of 1.6*x*, we can avoid one of the repeaters without violating our slew constraints, having a significant impact on performance or degrading total wirelength. This is the intuition behind our approach.

However, in practice, the decision of when a repeater should be inserted is considerably more complicated. Not only are ranges of inter-repeater distances acceptable (in contrast to the sharp threshold of *x* in our simple illustration), inter-repeater distances also depend intrinsically on the layer on which the net will eventually be routed. At the same time, one cannot afford to invoke a global router within each placement iteration because of runtime constraints. This leads to the need for a mechanism to predict the repeater needs of a net. Fortunately, this problem is not as intractable as it seems. Many industrial flows not only use length-based schemes[1] for repeater insertion (on all but a few high-fanout critical nets) [120], they also use length-based schemes to decide the layer assignments for different nets. The former heuristic owes its existence to the fact that most nets in a mapped design are two-pin nets, and greedy length-based

[1] Although we focus on improving the handling of repeaters required for the “common case” two-pin nets, the repeater needs of multi-pin nets can be estimated in a similar fashion. The net weighting method could also be improved with regard to multi-pin nets by using a length-based heuristic, as from [121].

repeater insertion on such nets is almost as good as more sophisticated dynamic programming based approaches, with the advantage of being much faster. The correlation between net length and layer assignment arises from the observation that short wires are best routed on the more resistive lower layers, while longer wires benefit more from the upper layers where their improved wire delays amortize the via stack penalties. Furthermore, since routing architectures usually follow preferred direction routing on each layer, process designers often architect pairs of adjacent metal layers to be electrically similar. Thus, metals M3 and M4 may form a pair, as may metals M5 and M6 (the upper metal layers are usually not available for block level synthesis).

Consequently, we use the length-based repeater prediction scheme described in [118], which has been validated against tape-out data. In this scheme, l_{rep}^{M} is the optimal inter-repeater distance on metal layer *M* and can easily be determined using simulation as in [118]. We insert a new repeater on a net routed on *M* only if its length is greater than $1.4\,l_{rep}^{M}$; similarly, we delete an existing repeater when the inter-repeater distance has shrunk to less than $0.7\,l_{rep}^{M}$. We assume that since short nets will be routed on the lower metal layers, we can use l_{rep}^{M3} (or the average of l_{rep}^{M3} and l_{rep}^{M4}) to determine their repeater needs. Similarly, we determine the repeater needs for the longer nets (with unrepeated length greater than $4\,l_{rep}^{M3}$) using l_{rep}^{M6}. (Routes on M1 and M2 are usually too short to require repeaters).

While any repeater prediction scheme is, by its very nature, an approximation of the actual requirements, it does serve the purpose of allowing a designer to budget space for the appropriate numbers of repeaters along the eventual routings of the nets that will need them. The fine-tuning of these repeaters and their exact sizing can be carried out in ECO (Engineering Change Order) mode subsequent to the placement phase.

Although the repeater prediction scheme used by us captures the impact of layer assignment and valid inter-repeater distance ranges on repeater insertion, it considerably complicates the design of a net-weighting scheme to reduce repeater counts. We next describe how we overcome these complications.

8.3.1 Threshold-based Net-weighting

For each net, a net weight multiplier is created based on its net length and multiplied to the original net weight before the system of equations is solved in Kraftwerk. The net weight multiplier is a function of the current net length and the nearest threshold length at which a repeater would be deleted or inserted (i.e., $0.7\, l_{rep}^{M}$ or $1.4\, l_{rep}^{M}$ for the relevant layer *M* in our scheme). One can use any one of several different functional templates for this net-weighting function; however, its key features are that it has a value of one at a center point away from the repeater insertion and deletion thresholds, and that it gradually increases as one moves away from this center point, reaching a maximum value at the net length in which a repeater would be inserted or deleted (as shown in Figure 74). For all these templates, a single parameter, *viz.*, the maximum possible value of net weight multiplier, is used to adjust the strength of the functions. Later in this chapter, we present an empirical evaluation of several different functional templates.

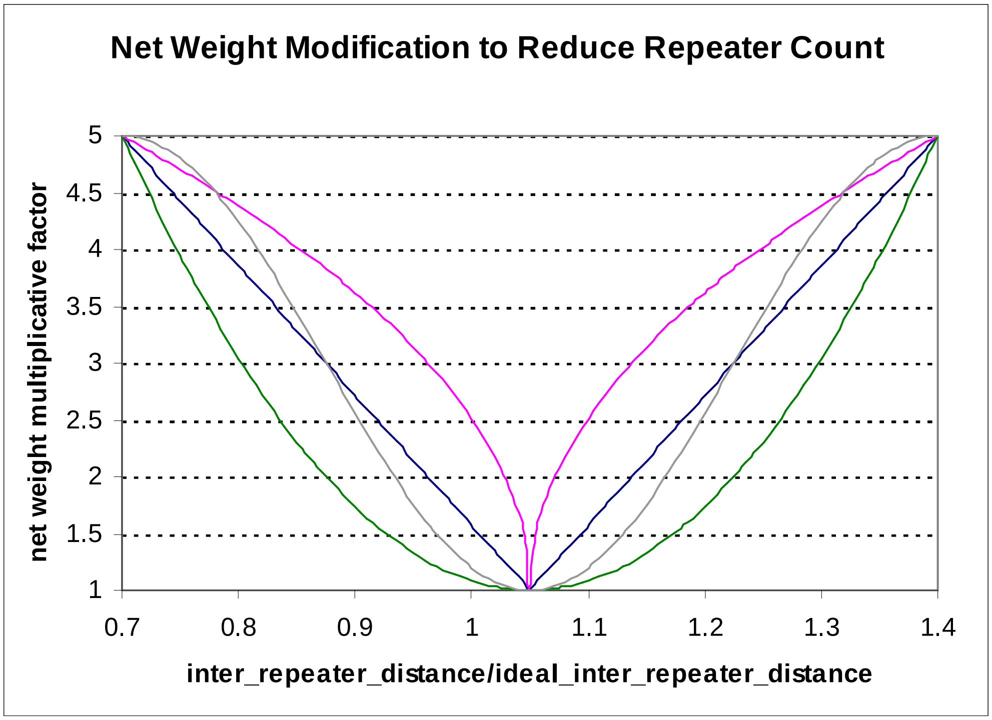

Figure 74. Possible net weight multiplier functions for repeater count reduction.

As can be seen in Figure 74, the functions consist of two halves. The right half discourages the net length from increasing beyond the threshold at which another repeater would be inserted. The left side encourages the net length to shrink beyond the next deletion threshold. This functional form is made symmetric to avoid introducing additional control parameters. It is replicated for each valid range of inter-repeater distances. When a net does not have any repeaters, the left side is ignored and set to one; since there are no repeaters to delete, encouraging the net to shrink further does not help reduce the repeater count.

Rapid changes in the net weight multipliers in successive iterations can cause the placement quality to degrade, and the net weight multiplier function itself may not be smooth. Therefore, we use exponential smoothing of the net weight multiplier based on the history of that multiplier over past iterations, in order to provide stability and promote convergence by ensuring that the multiplier does not change too rapidly. A smoothing constant of s ($0 \leq s \leq 1$) implies that the new (smoothed) value of a net weight is given by $w'_{new} = (1 - s)w'_{old} + s \cdot w_{new}$, where w'_{old} is the (smoothed) weight of that net in the previous iteration, and w_{new} is the unsmoothed weight for that net in the current iteration. Our scheme increases the average variation among different net weights at any time, and can potentially increase localized cell congestion after global placement. However, we faced no problems in the legalization of these regions even when our coarse legalizer FD-Mongrel was also using our net weighting scheme. Smoothing of the net weight multiplier also prevents instability issues in the initial iterations when net lengths are changing rapidly (because the multiplier asserts itself fully only if the length of a net remains near a repeater threshold over multiple iterations). Although we encountered no stability issue of this type, one could depreciate the net weight multipliers in the early iterations if needed, so that their full effect would be felt only in the later iterations when the spreading has stabilized.

In general, parameter tuning is not needed across different designs. The only control parameter, *viz.*, the maximum multiplier value, is not very sensitive because our implementation of Kraftwerk automatically scales the spreading forces to ensure that the connectivity-induced attractive forces are balanced with the density-induced spreading forces. Consequently, changing our maximum net weight multiplier parameter merely changes the extent of wire length tradeoff between nets that are close to a threshold and the nets that are far from it, without impacting the spreading significantly. In other words, it changes only the spread (i.e. variance) of the connectivity forces, but not their mean value vis-à-vis the spreading forces.

8.3.2 Layer Transitions

The multiplier function must be handled carefully around the point at which the net switches layer pairs, so that the net weight multiplier remains continuous across different layer pairs, and net weighting produces its intended effect. Recall that the primary purpose of this multiplier is to encourage repeater deletions that are possible, and to discourage possible repeater insertions. If such an insertion or deletion is not possible because of a predicted layer transition, then the net weight should not be blindly increased even if a minimum or maximum inter-repeater distance threshold is being approached. This can happen in the transition region from the lower layer pair (say, M3-M4) to the higher layer pair (say, M5-M6). This is illustrated in Figure 75 and Figure 76 using a linear net weighting function and generic wirelength values. The critical inter-repeater distances used in these two graphs are meant only to illustrate the two cases and do not represent the actual distances used in later experiments.

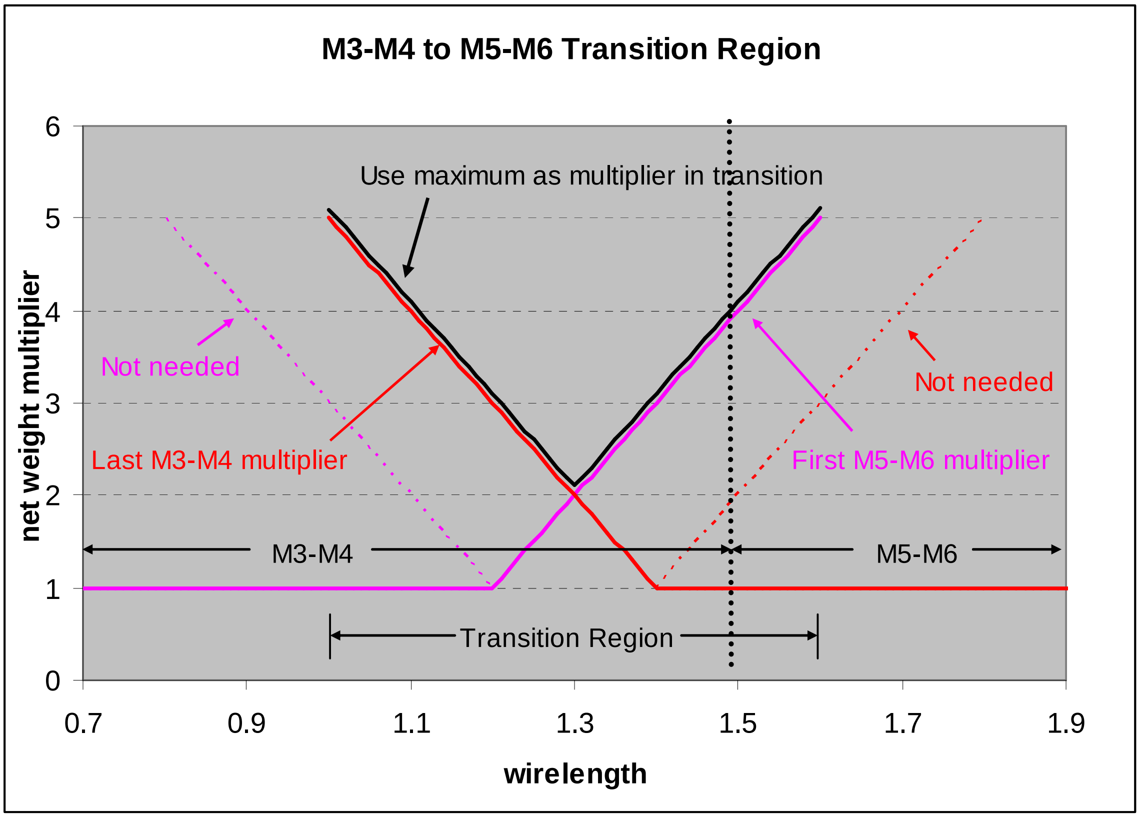

Figure 75. Overlapping M3-M4 deletion and M5-M6 insertion curve.

If a net is being modeled with the maximum possible repeaters for the lower pair, and it is approaching a net length at which it would switch over to the upper layer pair, then the weighting function should not increase even if it is approaching a M3-M4 maximum inter-repeater length threshold (see, for example, the dashed line labeled as "Not needed" in the right half of Figure 75). The net switches over to M5-M6 before it grows to a length at which another repeater would be inserted in a routing on M3-M4. Similarly, if a net has the fewest possible M5-M6 repeaters (in the sense that a further decrease in net length would cause the routing model to switch to using M3-M4), the net weight should not be increased for the purpose of deleting a repeater in the net weighting curve corresponding to M5-M6 (see the dashed line labeled as "Not Needed" in the left half of Figure 75).

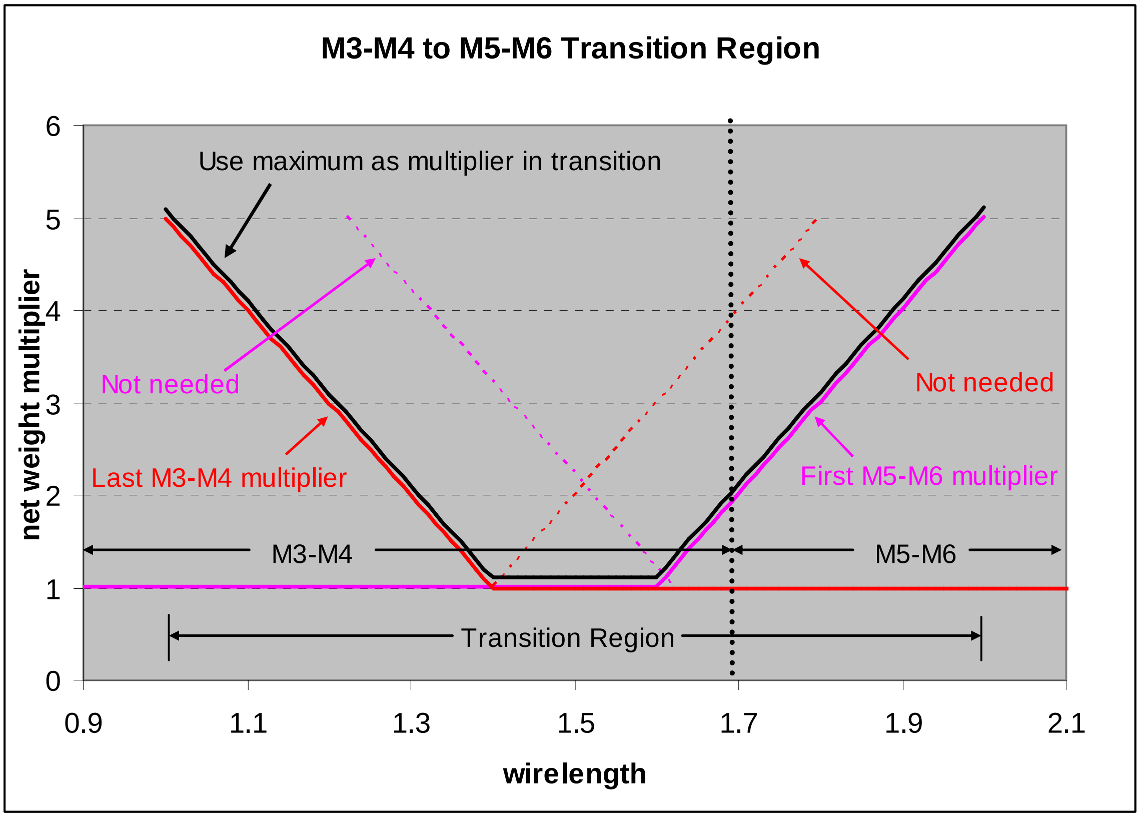

Figure 76. Overlapping M3-M4 insertion and M5-M6 deletion.

In Figure 75, the portions of the curves used for the last M3-M4 repeater deletion and the first M5-M6 insertion overlap directly; therefore, their max envelope is used. The other situation that can arise in the transition region (based on the amount of overlap between the last M3-M4 net weighting function and the first M5-M6 weighting function) is shown in Figure 76. In this figure, there is a certain amount of separation between the useful portions of the last M3-M4 curve and the first M5-M6 curve; so the multiplier is set to one in the intervening region. In both cases (*viz*., those illustrated in Figure 75 and Figure 76), the insertion portion of the last M3-M4 curve and the deletion portion of the first M5-M6 curve are ignored and replaced by a value of one. The maximum of the resulting curves can simply be used in the transition region as a continuous net weight multiplier function in both cases.

8.4 Implementation

In each iteration of Kraftwerk/MorePlace, we compute the net weight modifiers immediately after calling the repeater insertion/deletion procedure and before the system is solved for the new positions. This ensures that the net weights used by the solver are modified according to the most recent repeater configuration.

In FD-Mongrel, the calculation of the net weight modifiers is interleaved with the legalizer's iterations. The net weight modifiers are used as user-defined weights in Mongrel and are multiplied by the wirelengths in the gain function that determines which cells ripple-move in the fine grid from the most congested bin to the least congested bin. For convenience, these multipliers use the same net weighting function for both FD-Mongrel and Kraftwerk/MorePlace. However, exponential smoothing of the net weight multiplier is not needed in FD-Mongrel for stability[2]. Since the legalizer cannot work with virtual repeaters (unlike Kraftwerk/MorePlace), we instantiate the repeaters prior to passing the design to an FD-Mongrel iteration, but use the original nets to compute our net weight modifiers (in order to avoid fragmenting our netlist).

8.5 Experimental Results

In our first set of experiments, we explored the extent of repeater count reduction possible using different net weighting functions. For each net weighting function,

[2] Unlike global placement where the large flexibility available for each move necessitates an explicit history mechanism on net weights in order to create sufficient inertia to guard against extreme oscillations, the moves possible during legalization tend to be restricted enough that oscillations are avoided even without exponentially smoothed net weights.

placements were created using Kraftwerk/MorePlace and FD-Mongrel with repeater reduction net weighting, and then legalized fully. A set of circuits from a recent microprocessor was used in these experiments, with inter-repeater distances corresponding to the 45 nm and 32 nm technology nodes as in [118]. The placements were generated using a 2.8 GHz Intel® Xeon™ server with 4 GB memory. In these experiments, exponential smoothing with a smoothing constant of 0.05 was applied to the net weight multipliers between iterations as discussed earlier.

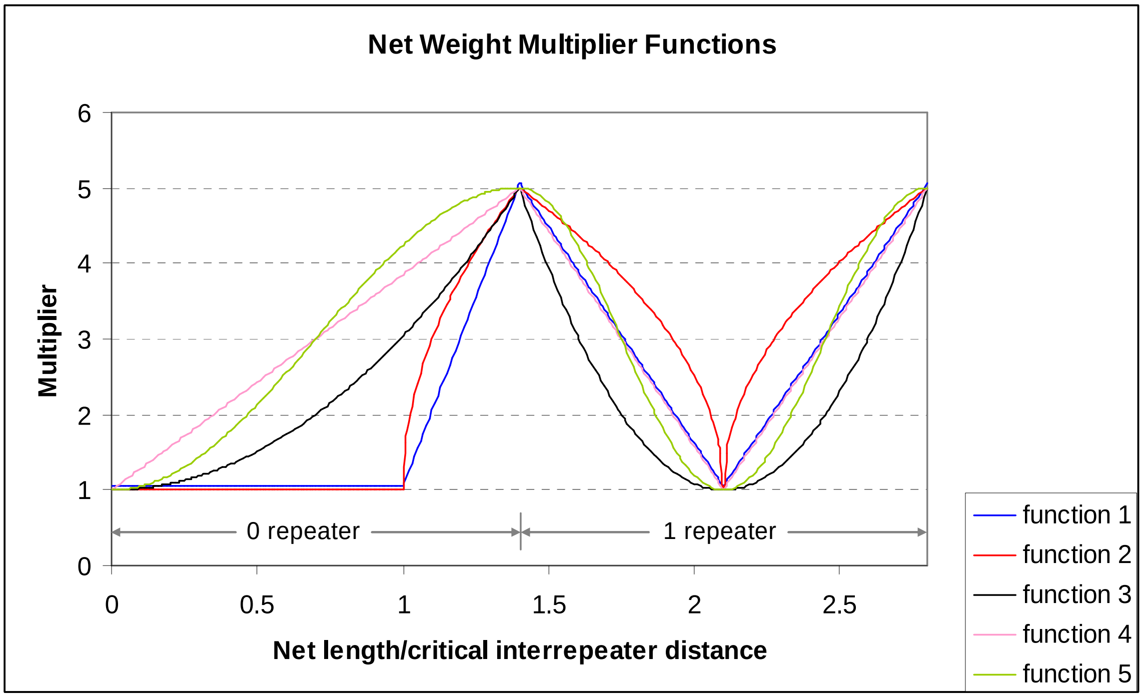

Figure 77. Possible net weight multiplier functions for the repeater count reduction.

The five weighting functions shown in Figure 77 were compared in Table 33 for the testcases from Table 32 at the 32 nm node. The functions begin with an initial curve when no repeaters are present and continue to increase until the first maximum inter-repeater distance is reached. For one or more repeaters, the functions use somewhat of a v-shaped curve. The M3-M4 to M5-M6 transition region is handled as described earlier. Function 1 has a value of one up to the first critical inter-repeater distance; from there it increases linearly to the maximum multiplier at the threshold corresponding to the maximum inter-repeater distance. For one or more repeaters, the function has a linear v-shape with the midpoint having a net weight multiplier of one and the minimum and maximum thresholds having the maximum multiplier value. Function 2 is similar except that it increases and decreases using a square-root function scaled appropriately to the minimum and maximum values. Function 3 increases quadratically from a value of one at an inter-repeater distance of zero to its maximum at the first insertion threshold. For one or more repeaters, the v-shaped curve is quadratic. Functions 4 and 5 are similar to Function 3 except they use linear and sinusoidal curves respectively.

Table 32. Testcases used for repeater count reduction experiments.

Testcase	Ckt_A	Ckt_B	Ckt_C	Ckt_D	Ckt_E
Cells	3978	4014	12312	13343	42127
Nets	4268	4384	13073	17685	42247

The results of the different weighting functions are shown in Table 33. In this table, Δ rptrs is the percent decrease in the number of repeaters with net weighting as compared to the control flow in which neither of Kraftwerk/MorePlace or FD-Mongrel uses our net weight modifiers, and Δ L_{total} is the percent increase in the total wirelength with repeater reduction. Functions 1 and 2 seemed to perform poorly with the larger testcases, causing large wirelength increases. Function 4 was slightly better, but Functions 3 and 5 gave the most improvement in repeater reduction and the least wirelength increase. Overall, it appears that Function 3 gave the best results, particularly for the larger circuits which are more indicative of future trends. Therefore, we decided to use Function 3 for the rest of our experiments, calculating the net weight multipliers using a quadratic function of the inter-repeater distance.

Table 33. Comparing function types at the 32nm node for repeater count reduction.

Testcase	Function Type									
	Function 1		Function 2		Function 3		Function 4		Function 5	
	Δ rptrs	Δ L_{total}	Δ rptrs	Δ L_{total}	Δ rptrs	Δ L_{total}	Δ rptrs	Δ L_{total}	Δ rptrs	Δ L_{total}
Ckt_A	25%	4%	28%	4%	36%	3%	23%	3%	33%	2%
Ckt_B	44%	3%	38%	6%	43%	3%	35%	3%	40%	5%
Ckt_C	11%	5%	13%	6%	20%	1%	17%	1%	18%	3%
Ckt_D	13%	2%	12%	3%	15%	2%	16%	1%	17%	1%
Ckt_E	13%	-9%	15%	-9%	25%	-15%	16%	-9%	21%	-12%

We next studied the impact of our control parameter (*viz*. the maximum multiplier value) on the quality of the fully legalized placement for our median-sized testcase Ckt_C at 32 nm; the resulting change in repeater count reduction and wirelength is plotted in Figure 78. Smaller maximum values yield less repeater count reduction and smaller perturbations in the wirelength, while larger values cause more wirelength increase and better improvement in the repeater count reduction to a certain extent. However, if the value is too high, repeater count reduction decreases and more repeaters may actually be needed as wirelengths increase too much. A maximum value of 5 was chosen for rest of the experiments because the wirelength increase is low and the repeater count reduction is high at this point.

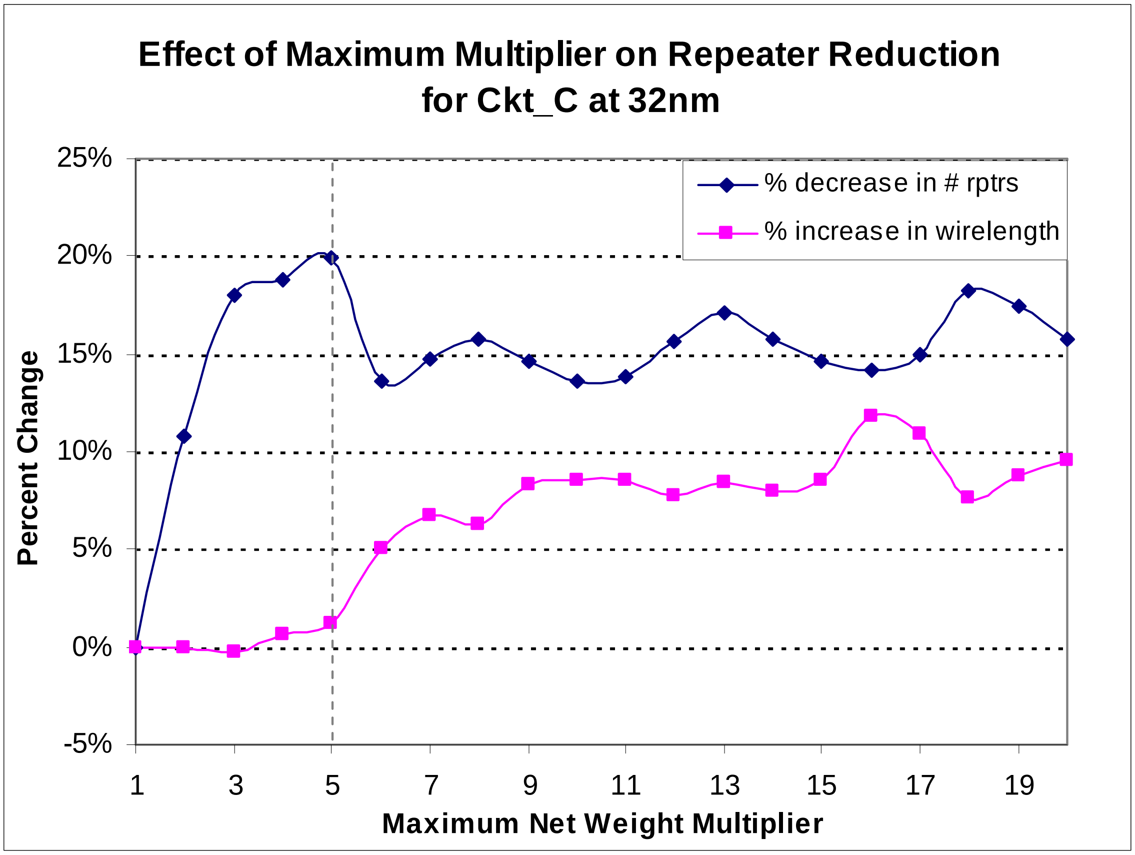

Figure 78. The effect of the maximum net weight multiplier on repeater count reduction.

The results for placements scaled to the 45 nm and 32 nm nodes and generated using three different design flows are shown in Table 34 and Table 35. All data is reported for fully legalized placements. Wirelengths are reported as half-perimeter measures.

MP/FDM uses the original MorePlace and FD-Mongrel without repeater reduction. In MP-RR/FDM, MorePlace with repeater reduction is followed by regular FD-Mongrel. Finally, MP-RR/FDM-RR uses repeater reduction net weighting in both MorePlace and FD-Mongrel. All three design flows are followed by a fine-grained legalization. In these tables, # rptrs is the number of repeaters, L_{total} is the total wirelength, Δ rptrs and Δ L_{total} are respectively the percent decrease in repeater count and increase in total wirelength (compared to the MP/FDM results).

Table 34. Comparing repeater count reduction at 32 nm.

Testcase	MP/FDM		MP-RR/FDM		MP-RR/FDM-RR		
	# rptrs	L_{total}	Δ rptrs	Δ L_{total}	Δ rptrs	Δ L_{total}	Time Overhead
Ckt_A	447	1.98E+07	10.7%	4.3%	36.2%	3.1%	25.7%
Ckt_B	431	1.75E+07	16.5%	1.9%	42.9%	3.4%	114.8%
Ckt_C	2869	8.62E+07	11.9%	1.3%	20.0%	1.2%	472.2%
Ckt_D	6304	1.57E+08	12.2%	0.3%	15.0%	1.8%	266.4%
Ckt_E	22984	5.42E+08	18.5%	-14.6%	25.3%	-14.5%	52.1%

Table 35. Comparing repeater count reduction at 45 nm.

Testcase	MP/FDM		MP-RR/FDM		MP-RR/FDM-RR		
	# rptrs	L_{total}	Δ rptrs	Δ L_{total}	Δ rptrs	Δ L_{total}	Time Overhead
Ckt_A	124	1.99E+07	30.6%	1.1%	42.7%	2.2%	23.4%
Ckt_B	133	1.74E+07	30.1%	1.1%	63.2%	0.7%	114.0%
Ckt_C	1655	8.97E+07	26.0%	-1.9%	32.3%	-1.1%	20.6%
Ckt_D	3700	1.60E+08	22.7%	-1.3%	26.8%	-0.3%	188.1%
Ckt_E	11004	4.56E+08	12.8%	-2.6%	23.0%	-1.5%	-3.7%

Table 34 and Table 35 show that, as expected, the best repeater reduction results are obtained when net weighting is applied to both MorePlace and FD-Mongrel. With this flow, 38% fewer repeaters are needed at 45 nm and 28% fewer repeaters at 32 nm. The wirelength deterioration is usually very low; in fact, the use of repeater reduction even decreases the total wirelength in several cases (because a design with fewer repeaters is more easily legalizable, thus causing less wirelength degradation after global placement). Even the results of the MP-RR/FDM design flow are better than the basic MP/FDM flow. In this flow, there is an average of 24% (14%) fewer repeaters at the 45 nm (32 nm) node respectively. The improved benefits of the MP-RR/FDM-RR flow over the MP-RR/FDM flow can be explained by the fact that the former continues active repeater reduction during the coarse legalization process. In contrast, a detailed analysis (omitted here due to space constraints) of the data for the MP-RR/FDM flow shows that some of the repeater reduction visible after the global placement phase is frittered away during coarse legalization in the absence of threshold-based net weights. Compared to coarse legalization, the smaller magnitude of the moves during fine-grained legalization does not impact the repeater gains significantly. Repeater count reduction does add a certain amount of overhead onto the runtime as shown in Table 34 and Table 35 because of the additional computation and slower convergence. However, this overhead tends to be lower for the largest test cases because of the difficulty of fine-grained legalization with a larger number of repeaters in the baseline MP/FDM flow.

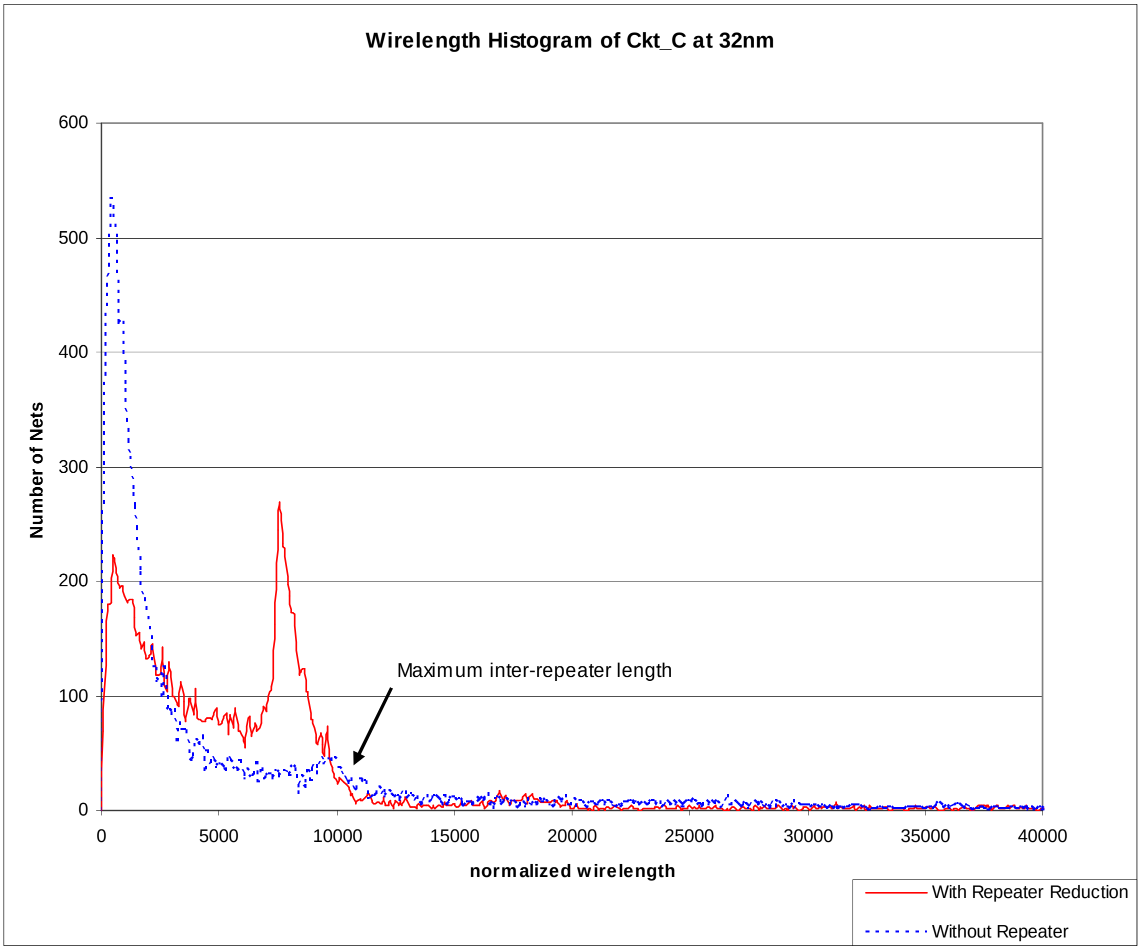

Figure 79. The wirelength histogram with repeater count reduction.

The impact of our threshold-based net weighting scheme on the wiring histogram of a design is illustrated using data for our median-sized testcase Ckt_C in Figure 79. It can be seen that repeater count reduction decreases the length of nets near the repeater insertion threshold at the expense of nets with very short wirelengths (which do not require repeaters or have significant wire loads). This results in an increased aggregate of nets just before the wirelength where the first repeater would be added, just as one would expect. (Note that MorePlace causes a small perturbation in the wirelength histogram by itself near the first repeater insertion threshold, due to the additional quadratic forces in its attractive repeater force model).

In Figure 80 and Figure 81, the scatter plots show the change in wirelength for each net in Ckt_C during the final iteration of Kraftwerk/MorePlace. In the plots, each net is represented by a color coded point based on the number of repeaters that it had prior to the iteration. The x-axis represents the final wirelength for each net, while the y-axis represents the change in wirelength during the final iteration. Figure 80 reflects the results obtained by MorePlace without repeater count reduction, and Figure 81 contains the results obtained by MorePlace with repeater count reduction net weighting.

In Figure 80, the change in wirelength is spread out almost randomly in both directions, independent of the number of repeaters or the inter-repeater length. In contrast, the distribution of the data points in Figure 81 is strongly affected by repeater reduction, indicating the effects of net weighting on specific nets. Firstly, the changes in the wirelength are smaller, and the distribution indicates more stability overall with fewer large changes in the net wirelengths. Secondly, the distribution of the wirelength changes depends on the net’s wirelength and its current repeater count, indicating that our net weighting scheme is producing its intended effects (i.e. shrinking certain nets at the expense of less critical nets). The contraction in the net wirelengths is concentrated near the repeater insertion and deletion thresholds, not only in frequency (i.e. numbers of contracting nets) but also in magnitude (i.e., extent of contraction). In contrast, wirelength increases are concentrated far from the thresholds. Unlike the plot in Figure 80, the distribution of wirelength change tends to be one-sided in certain areas depending on the repeater count reduction net weighting.

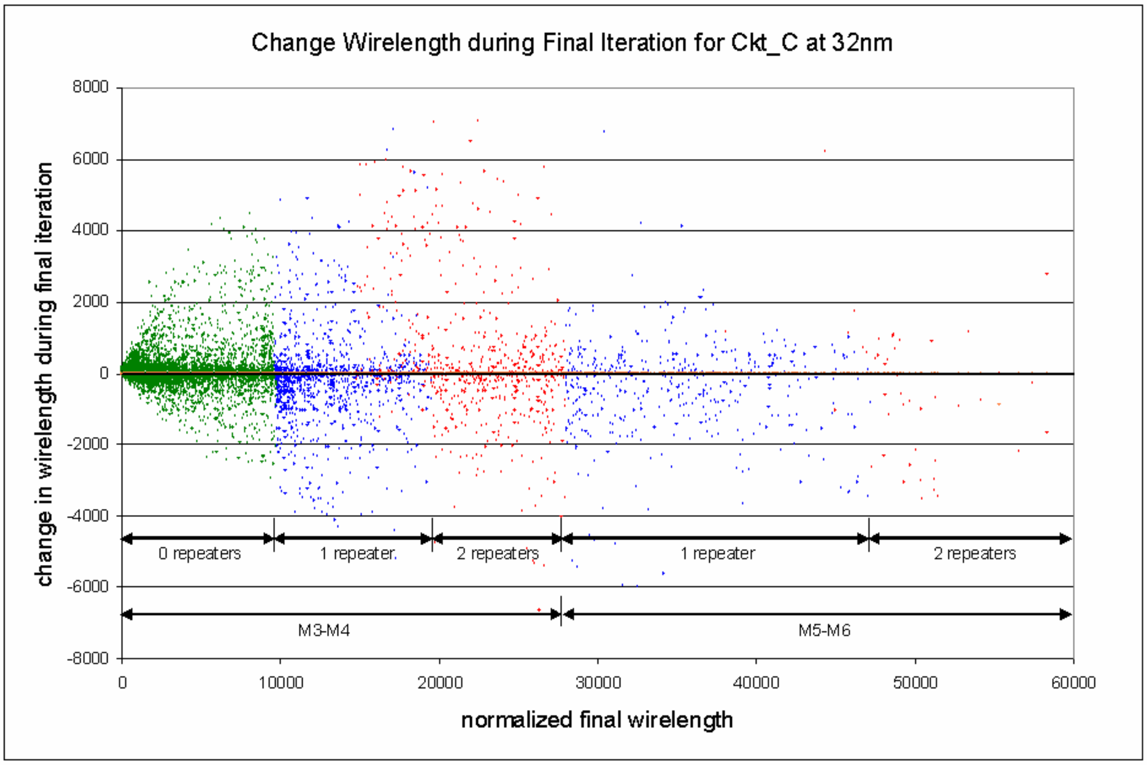

Figure 80. Final wirelength change without repeater count reduction.

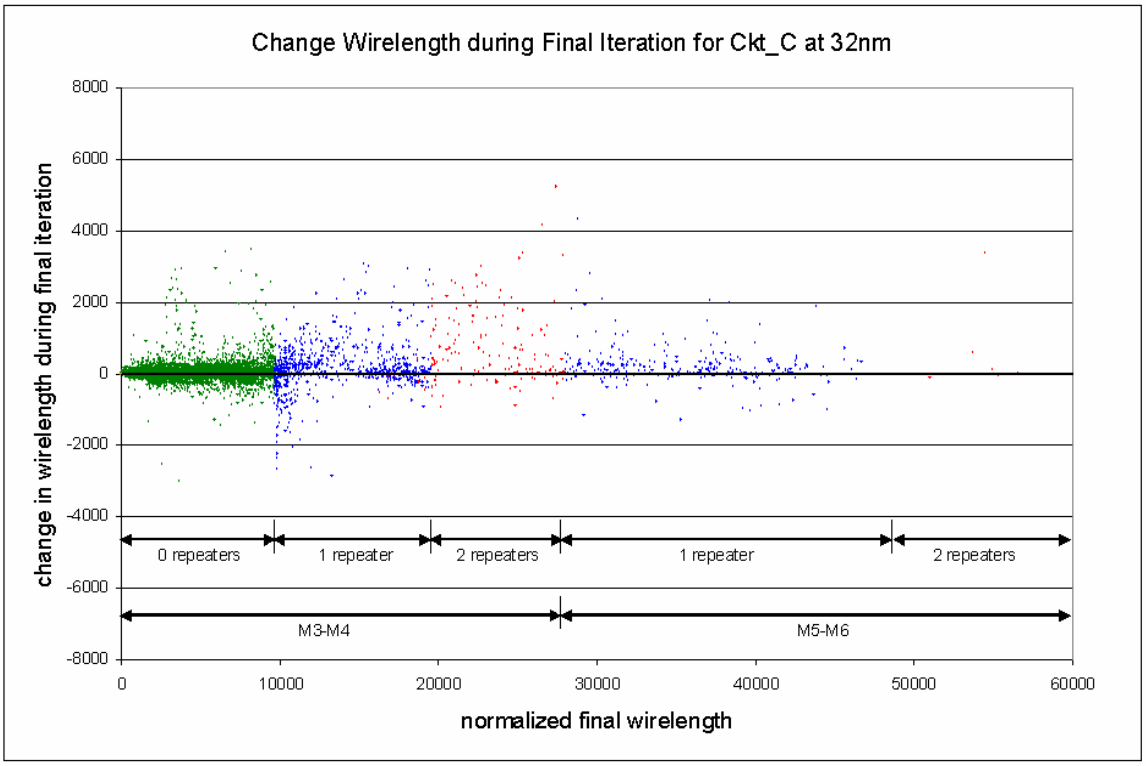

Figure 81. Final wirelength change with repeater count reduction.

Although this method was tested only with wire-length driven placement, we are optimistic that our approach could be extended to timing-driven placement without deteriorating significantly under timing constraints. The rationale for our optimism is as follows. We found considerable robustness in the quality of our results even as different parameters were varied. Therefore, given that the majority of nets do not lie on timing-critical paths, one can guarantee that the most performance-sensitive timing-critical nets will not be unnecessarily lengthened (by forcing their "net weight multiplier" to its maximum value) and still allow for considerable flexibility in trading off net length between nets that are close to the repeater insertion thresholds and those non-critical nets that are not close to the thresholds. Furthermore, with lower repeater counts, our experiments demonstrate that legalization occurs with considerably less perturbation as measured by wirelength degradation and additional repeater requirements; this significant reduction in backend layout degradation that is a collateral benefit of our repeater count reduction mechanism is especially important in ensuring timing closure after timing-driven placement.

8.6 Conclusion

Net weighting is useful not only in traditional timing- and power-driven placement, but also in reducing the number of repeaters needed in the design of future ICs. In this chapter, we have shown that these repeater count reductions can be made without sacrificing placement quality. We have presented the mechanics of constructing a context-sensitive net weighting scheme that incorporates the effects of layer assignment and inter-repeater distance back-offs. Our scheme produced placements with significantly fewer repeaters, with only minor wirelength impact.

9 Conclusions

In future VLSI circuits, feature size reduction and three-dimension integration can be utilized to reduce wirelengths and increase transistor packing densities. With these advances in fabrication technology, improvements are needed in the EDA tools not only to produce feasible designs with the new technologies but also to take full advantage of their potential benefits. To accomplish this, several problems that arise with decreasing feature size and three-dimension integration need to be addressed. First, routability between layers in 3D ICs is limited so interlayer via densities should be carefully managed. Second, increasing packing densities can cause the power densities to rapidly increase and result in correspondingly higher temperatures. If left unchecked, these higher temperatures can drastically reduce performance and reliability. In addition, greater thicknesses and lower thermal conductivities in 3D ICs intensify thermal problems. Next, interconnect delays scale poorly with decreasing feature sizes and result in rapidly increasing repeater counts that can complicate the design process. Finally, increasing transistor densities and chip areas result in an exponential increase in the number of transistors so efficiency is needed in order to ensure future utility of the EDA tools that tackle these issues.

In this thesis, these challenges to the design of future VLSI circuits were addressed at the placement stage of the design flow. First, an efficient placement method was created to explore the tradeoff between interlayer via counts and wirelength in 3D ICs. This can enable wirelength to be minimized and performance to be maximized for any interlayer via density limitation imposed by fabrication. Second, this method was extended to include thermal considerations by moving cells to more favorable thermal environments and by reducing dynamic power. This thermal placement method allows the tradeoff between thermal improvement, wirelength, and interlayer via counts to be explored. This tradeoff showed that with very minor degradation in the wirelength, large reductions in the temperature can be achieved. Next, thermal improvements can also be made in 3D ICs immediately after placement by incorporating thermal vias. However, thermal vias take up valuable area needed for routing, particularly when considering the limitation on the interlayer via densities. The tradeoff between thermal effects and thermal via usage was explored by this method and allows significant thermal improvements to be made with a minimal usage of thermal vias. Finally, the problematic increase in repeater counts was mitigated during global placement and legalization by using net weighting to contracts nets near repeater insertion and deletion thresholds. Placements were produced with significantly fewer repeaters and only minor impacts on wirelength and run time. With these methods to address the future design challenges, efficiency was an important objective. In addition, concerns were maintained throughout the process so that improvements made by global placement were not lost by legalization.

References

[1] K. Saraswat, S. J. Souri, K. Banerjee, and P. Kapur, “Performance Analysis and Technology of 3-D ICs,” *Proceedings of the International Workshop on System-level Interconnect Prediction*, pp. 85-90, 2000.

[2] A. Rahman, A. Fan, and R. Reif, “Comparison of Key Performance Metrics in Two- and Three-Dimensional Integrated Circuits,” *Proceedings of the Interconnect Technology Conference*, pp. 18-20, 2000.

[3] 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, No. 5, pp. 602-633, 2001.

[4] S. Das, A. Chandrakasan, and R. Reif, “Three-Dimensional Integrated Circuits: Performance, Design Methodology, and CAD Tools,” *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pp. 13-18, 2003.

[5] Y. Deng and W. Maly, “2.5D System Integration: A Design Driven System Implementation Schema,” *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 450-455, 2004.

[6] S. Das, A. Fan, K.-N. Chen, C. S. Tan, N. Checka, and R. Reif, “Technology, Performance, and Computer-Aided Design of Three-Dimensional Integrated Circuits,” *Proceedings of the International Symposium on Physical Design*, pp. 108-115, 2004.

[7] Y. Xie, G. Loh, B. Black, and K. Bernstein, “Design Space Exploration for 3D Architectures,” *ACM Journal on Emerging Technologies in Computing Systems*, Vol. 2, No. 2, pp. 65-103, 2006.

[8] Y. Deng and W. Maly, “Interconnect Characteristics of 2.5-D System Integration Scheme,” *Proceedings of the International Symposium on Physical Design*, pp. 171-175, 2001.

[9] J. W. Joyner, P. Zarkesh-Ha, and J. D. Meindl, “A Stochastic Net-Length Distribution for a Three-Dimensional System-on-a-Chip (3D-SoC),” *Proceedings of the 14th Annual IEEE ASIC/SOC Conference*, pp. 147-151, 2001.

[10] B. S. Meyerson, “How Does One Define ‘Technology’ Now That Classical Scaling Is Dead (and Has Been for Years)?” *Proceedings of the Design Automation Conference*, pp. xxv, 2005.

[11] J. Cong, and Y. Zhang, "Thermal-Driven Multilevel Routing for 3-D ICs," *Proceedings of the Asia South Pacific Design Automation Conference*, pp.121-126, 2005.

[12] T-Y. Chiang, S. J. Souri, C. O. Chui, and K. C. Saraswat, “Thermal Analysis of

Heterogeneous 3D ICs with Various Integration Scenarios," *International Electron Devices Meeting Technical Digest*, pp. 681-684, 2001.

[13] S. Im and K. Banerjee, "Full Chip Thermal Analysis of Planar (2-D) and Vertically Integrated (3-D) High Performance ICs," *International Electron Devices Meeting Technical Digest*, pp. 727-730, 2000.

[14] A. Rahman and R. Reif, "Thermal Analysis of Three-Dimensional (3-D) Integrated Circuits (ICs)," *Proceedings of the Interconnect Technology Conference*, pp. 157-159, 2001.

[15] S. F. Al-Sarawi, D. Abbott and P. D. Franzon, "A Review of 3-D Packaging Technology," *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging*, Vol. 21, No. 1, pp. 2-14, 1998.

[16] Y. K. Tsui, S. W. R. Lee, J. S. Wu, J. K. Kim, and M. M. F. Yuen, "Three-Dimensional Packaging for Multi-Chip Module with Through-the-Silicon Via Hole," *Electronics Packaging Technology Conference*, pp. 1-7, 2003.

[17] T. Kunio, K. Oyama, Y. Hayashi, and M. Morimoto, "Three-Dimensional IC's Having Four Stacked Active Device Layers," *International Electron Devices Meeting Technical Digest*, pp. 837–840, 1989.

[18] G. W. Neudeck, S. Pae, J. P. Denton, and T. Su, "Multiple Layers of Silicon-On-Insulator (SOI) for Nanostructure Devices," *Journal of Vacuum Science and Technology-B*, Vol. 17, No. 3, pp. 994–998, 1999.

[19] V. Subramanian, P. Dankoski, L. Degertekin, B. T. Khuri-Yakub, and K. C. Saraswat, "Controlled Two-step Solid-Phase Crystallization for High Performance Polysilicon TFTs", *IEEE Electron Device Letters*, Vol. 18, pp. 378-81, 1997.

[20] V. Subramanian and K. C. Saraswat, "High-Performance Germanium-Seeded Laterally Crystallized TFT's for Vertical Device Integration," *IEEE Transactions on Electron Devices*, Vol. 45, No. 9, pp. 1934–1939, 1998.

[21] M. Chan and P. K. Ko, "Development of a Viable 3D Integrated Circuit Technology," *Science in China*, Vol. 44, No. 4, pp. 241-248, 2001.

[22] K.W. Guarini, A. W. Topol, M. Ieong, R. Yu, L. Shi, M. R. Newport, D. J. Frank, D. V. Singh, G. M. Cohen, S. V. Nitta, D. C. Boyd, P. A. O'Neil, S. L. Tempest, H. B. Pogge, S. Purushothaman, and W. E. Haensch, "Electrical Integrity of State-of-the-Art 0.13 μm SOI CMOS Devices and Circuits Transferred for Three-Dimensional (3D) Integrated Circuit (IC) Fabrication," *International Electron Devices Meeting Technical Digest*, pp. 943-945, 2002.

[23] R. J. Gutmann, J.-Q. Lu, A. Y. Zeng, S. Devarajan, and K. Rose, "Wafer-Level Three-Dimensional Monolithic Integration for Heterogeneous Silicon ICs", *Silicon Monolithic Integrated Circuits in RF Systems*, pp. 45-8, 2004.

[24] K. Warner, J. Burns, C. Keast, R. Kunz, D. Lennon, A. Loomis, W. Mowers, and D. Yost, "Low-Temperature Oxide-Bonded Three-Dimensional Integrated

Circuits," *IEEE International SOI Conference Proceedings*, pp. 123-124, 2002.

[25] A. Fan, A. Rahman, and R. Reif, "Copper Wafer Bonding," *Electrochemical and Solid-State Letters*, Vol. 2, pp. 534-536, 1999.

[26] R. Reif, A. Fan, K.-N. Chen, and S. Das, "Fabrication Technologies for Three-Dimensional Integrated Circuits," *Proceedings of the International Symposium on Quality Electronic Design*, pp. 33-37, 2002.

[27] B. Goplen and S. S. Sapatnekar, "Efficient Thermal Placement of Standard Cells in 3D ICs using a Force Directed Approach," *Proceedings of the International Conference on Computer-Aided Design*, pp. 86-89, 2003.

[28] 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*, Vol. 19, No. 2, pp. 253-266, 2000.

[29] P. Wilkerson, A. Raman, and M. Turowski, "Fast, Automated Thermal Simulation of Three-Dimensional Integrated Circuits," *The Ninth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, Vol. 1, pp. 706-713, 2004.

[30] V. Szekely, M. Rencz, and B. Courtois, "Tracing the Thermal Behavior of ICs," *Design & Test of Computers*, Vol. 15, No. 2, pp. 14-21, 1998.

[31] T-Y. Wang and C. C-Y. Chen, "Thermal-ADI: a Linear-Time Chip Level Transient Thermal Simulator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 21, No. 12, pp. 1434-1445, 2002.

[32] Y. Zhan and S. S. Sapatnekar, "Fast Computation of the Temperature Distribution in VLSI Chips Using the Discrete Cosine Transform and Table Look-up," *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 87-92, 2005.

[33] D. L. Logan, *A First Course in the Finite Element Method*, 3rd ed., Brooks/Cole Pub. Co., 2002.

[34] L. T. Pillage, R. A. Rohrer, and C. Visweswariah, *Electronic Circuit and System Simulation Methods*. New York: McGraw-Hill, 1995.

[35] J. Cong, J. R. Shinnerl, M. Xie, T. Kong, and X. Yuan, "Large-Scale Circuit Placement," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 10, No. 2, pp. 389-430, 2005.

[36] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," *Proceedings of the Design Automation Conference*, pp. 269-274, 1998.

[37] K. P. Vorwerk, A. Kennings, and A. Vannelli, "Engineering Details of a Stable Force-Directed Placer," *Proceedings of the International Conference on Computer-Aided Design*, pp. 573-580, 2004.

[38] A. Kennings, K. P. Vorwerk, “Force-Directed Methods for Generic Placement,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2006.

[39] B. Hu and M. Marek-Sadowska, “FAR: Fixed-Points Addition and Relaxation Based Placement,” *Proceedings of the International Symposium on Physical Design*, pp. 161-166, 2002.

[40] B. Hu, Y. Zeng, and M. Marek-Sadowska, “mFAR: Fixed-Points-Addition-Based VLSI Placement Algorithm,” *Proceedings of the International Symposium on Physical Design*, pp. 239-241, 2005.

[41] Natarajan Viswanathan and Chris Chong-Nuen Chu, “FastPlace: Efficient Analytical Placement using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model,” *Proceedings of the International Symposium on Physical Design*, pp. 26-33, 2004.

[42] Z. Xiu, J. Ma, S. Fowler, and R. Rutenbar, “Large-Scale Placement by Grid-Warping,” *Proceedings of the Design Automation Conference*, pp. 351–356, 2004.

[43] A. B. Kahng, and Q. Wang, “Implementation and Extensibility of an Analytic Placer,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 5, pp. 734-747, 2005.

[44] T. Chan, J. Cong, J. Shinnerl, T. Kong, and K. Sze. “An Enhanced Multilevel Algorithm for Circuit Placement,” *Proceedings of the International Conference on Computer Aided Design*, pp. 299-306, 2003.

[45] T. Chan, J. Cong, and K. Sze, “Multilevel Generalized Force-directed Method for Circuit Placement,” *Proceedings of the International Symposium on Physical Design*, pp. 185-192, 2005.

[46] J. Kleinhans, G. Sigl, F. Johannes and K. Antreich, “GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 356-365, 1991.

[47] A. E. Caldwell, A. B. Kahng, and I. L. Markov, “Can Recursive Bisection Alone Produce Routable Placements?” *Proceedings of the Design Automation Conference*, pp. 693-698, 2000.

[48] A. E. Caldwell, A. B. Kahng, I. L. Markov, “Optimal Partitioners and End-case Placers for Standard-cell Layout,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 19, No. 11, pp. 1304-1314, 2000.

[49] M. C. Yildiz and P. H. Madden, “Improved Cut Sequences for Partitioning Based Placement,” *Proceedings of the Design Automation Conference*, pp. 776-779, 2001.

[50] A. R. Agnihotri, M. C. Yildiz, A. Khatkhate, A. Mathur, S. Ono, and P. H. Madden, “Fractional Cut: Improved Recursive Bisection Based Placement,” *Proceedings of the International Conference on Computer-Aided Design*, pp. 307-

310, 2003.

[51] A. R. Agnihotri, S. Ono, and P. H. Madden, "Recursive Bisection Placement: Feng Shui 5.0 Implementation Details," *Proceedings of the International Symposium on Physical Design*, pp. 230-232, 2005.

[52] M. Wang, X. Yang, and M. Sarrafzadeh, "Dragon2000: Standard-cell Placement Tool for Large Industry Circuits," *Proceedings of the International Conference on Computer-Aided Design*, pp. 260-263, 2000.

[53] T. Taghavi, X. Yang, B.-K. Choi, "Dragon2005: Large-Scale Mixed-size Placement Tool," *Proceedings of the International Symposium on Physical Design*, pp. 245-247, 2005.

[54] W.-J. Sun and C. Sechen, "Efficient and Effective Placement for Very Large Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 3, pp. 349–359, 1995.

[55] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain," *IEEE Transactions on VLSI Systems*, Vol. 7, No. 1, pp. 69-79, 1999.

[56] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel Circuit Partitioning," *Proceedings of the Design Automation Conference*, pp. 530-533, 1997.

[57] A. E. Dunlop and B. W. Kernighan, "A Procedure for Placement of Standard Cell VLSI Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 4, No. 1, pp. 92-98, 1985.

[58] A. E. Caldwell, A. B. Kahng, I. L. Markov, "Hierarchical Whitespace Allocation in Top-down Placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 22, No. 11, pp. 716-724, 2003.

[59] M. A. Breuer, "A Class of Min-Cut Placement Algorithms," *Proceedings of the Design Automation Conference*, pp. 284-290, 1997.

[60] B. Landman and R. Russo, "On a Pin Versus Block Relationship for Partitioning of Logic Graphs," *IEEE Transactions On Computers*, Vol. C-20, No. 12, pp. 1469-1479, 1971.

[61] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu, and I. L. Markov, "Capo: Robust and Scalable Open-Source Min-cut Floorplacer," *Proceedings of the International Symposium on Physical Design*, pp. 224-226, 2005.

[62] J. A. Roy, D. A. Papa, A. N. Ng, and I. L Markov, "Satisfying Whitespace Requirements in Top-down Placement," *Proceedings of the International Symposium on Physical Design*, pp. 206-208, 2006.

[63] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Relaxed Partitioning Balance Constraints in Top-Down Placement," *Proceeding of the IEEE ASIC Conference*,

pp. 229-232, 1998.

[64] A. B. Kahng and S. Reda, "Placement Feedback: A Concept and Method for Better Min-cut Placement," *Proceedings of the Design Automation Conference*, pp. 357-362, 2004.

[65] D. J.-H. Huang and A. B. Kahng, "Partitioning-Based Standard-Cell Global Placement With An Exact Objective," *Proceedings of the International Symposium on Physical Design*, pp. 18-25, 1997.

[66] P. H. Madden, "Partitioning by Iterative Deletion," *Proceedings of the International Symposium on Physical Design*, pp. 83-89, 1999.

[67] M. C. Yildiz and P. H. Madden, "Global Objectives for Standard Cell Placement," *Proceedings of the Great Lakes Symposium on VLSI*, pp. 68-72, 2001.

[68] S.-W. Hur, T. Cao, K. Rajagopal, Y. Parasuram, A. Chowdhary, V. Tiourin, and B. Halpin, "Force Directed Mongrel with Physical Net Constraints," *Proceedings of the Design Automation Conference*, pp. 214-219, 2003.

[69] K. Doll, F. Johannes, and K. Antreich, "Iterative Placement Improvement by Network Flow Methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 10, pp. 1189-1200, 1994.

[70] J. Vygen, "Algorithms for Detailed Placement of Standard Cells," *Proceedings of the Design Automation and Test in Europe*, pp.321-324, 1998.

[71] B. Halpin, N. Sehgal, and C. Y. R. Chen, "Detailed Placement with Net Length Constraints," *IEEE International Workshop on System-on-Chip*, pp. 22-27, 2003.

[72] S.-W. Hur, J. Lillis, "Mongrel: Hybrid Techniques for Standard Cell Placement," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 165-170, 2000.

[73] U. Brenner and J. Vygen, "Legalizing a Placement with Minimum Total Movement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 12, pp. 597-613, 2004.

[74] H. Zhou, W. Wu, and X. Hong, "PAFLO: A Fast Standard-Cell Detailed Placement Algorithm," *International Conference on Communications, Circuits and Systems and West Sino Expositions*, pp. 1401-1405, 2002.

[75] A. Kahng, P. Tucker, and A. Zelikovsky, "Optimization of Linear Placements for Wirelength Minimization with Free Sites," *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 241-244, 1999.

[76] M. Sarrafzadeh and M. Wang, "NRG: Global and Detailed Placement," *Proceedings of the International Conference on Computer-Aided Design*, pp. 164-169, 1997.

[77] M. Pan, N. Viswanathan, and C. Chu, "An Efficient and Effective Detailed Placement Algorithm," *Proceedings of the International Conference on Computer-*

Aided Design, pp. 48-55, 2005.

[78] S. Ono and P. H. Madden, “On Structure and Suboptimality in Placement,” *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 331-336, 2005.

[79] T. Tanprasert, “An Analytical 3-D Placement that Reserves Routing Space,” *Proceedings of the International Symposium on Circuits and Systems*, Vol. 3, pp. 69-72, 2000.

[80] I. Kaya, M. Olbrich, and E. Barke, “3-D Placement Considering Vertical Interconnects,” *Proceedings of the IEEE International SOC Conference*, pp. 257-258, 2003.

[81] S. T. Obenaus and T. H. Szymanski, “Gravity: Fast Placement for 3-D VLSI,” *ACM Transactions on Design Automation of Electronic Systems*, Vol. 8 No. 3, pp. 298-315, 2003.

[82] R. Hentschke, G. Flach, F. Pinto, and R. Reis, “Quadratic Placement for 3D Circuits Using Z-Cell Shifting, 3D Iterative Refinement and Simulated Annealing,” *Proceedings of the 19th Annual Symposium on Integrated Circuits and Systems Design*, pp. 220-225, 2006.

[83] S. Das, A. Chandrakasan, and R. Reif, “Design Tools for 3-D Integrated Circuits,” *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 53-56, 2003.

[84] C. Ababei, H. Mogal, and K. Bazargan, “Three-dimensional Place and Route for FPGAs,” *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 773-778, 2005.

[85] K. Balakrishnan, V. Nanda, S. Easwar, and S. K. Lim, “Wire Congestion and Thermal Aware 3D Global Placement,” *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 1131-1134, 2005.

[86] http://er.cs.ucla.edu/benchmarks/ibm-place/

[87] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, “The ISPD2005 Placement Contest and Benchmark Suite,” *Proceedings of the International Symposium on Physical Design*, pp. 216-220, 2005.

[88] S. Goto, “An Efficient Algorithm for the Two-Dimensional Placement Problem in Electrical Circuit Layout,” *IEEE Transitions on Circuits and Systems*, Vol. CAS-28, No. 1, pp. 12-18, 1981.

[89] K. Warner, C. Chen, R. D’Onofrio, C. Keast, and S. Poesse, “An Investigation of Wafer-to-Wafer Alignment Tolerances for Three-Dimensional Integrated Circuit Fabrication,” *IEEE International SOI Conference Proceedings*, pp. 71-72, 2004.

[90] J. Burns, L. McIlrath, C. Keast, C. Lewis, A. Loomis, K. Warner, and P. Wyatt, “Three-Dimensional Integrated Circuits for Low-Power, High-Bandwidth Systems

on a Chip," *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 268-269, 2001.

[91] C. N. Chu and D. F. Wong, "A Matrix Synthesis Approach to Thermal Placement," *Proceedings of the International Symposium on Physical Design*, pp. 163-168, 1997.

[92] Y.-S. Cheon, P.-H. Ho, A. B. Kahng, S. Reda, and Q. Wang, "Power-Aware Placement," *Proceedings of the Design Automation Conference*, pp. 795-800, 2005.

[93] B. Obermeier and F. M. Johannes, "Temperature-Aware Global Placement," *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 143-148, 2004.

[94] G. Chen and S. S. Sapatnekar, "Partition-Driven Standard Cell Thermal Placement," *Proceedings of the International Symposium on Physical Design*, pp. 75-80, 2003.

[95] Chin-Chih Chang, Jason Cong, Michail Romesis, and Min Xie, "Optimality and Scalability Study of Existing Placement Algorithms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 4, pp. 537-549, 2004.

[96] J. Cong, "Challenges and Opportunities for Design Innovations in Nanometer Technologies," *Invited Semiconductor Research Corporation Design Sciences Concept Paper*, pp. 1-15, 1998.

[97] B. Goplen and S. S. Sapatnekar, "Thermal Via Placement in 3D ICS," *Proceedings of the International Symposium on Physical Design*, pp. 167-174, 2005.

[98] B. Goplen and S. S. Sapatnekar, "Placement of Thermal Vias in 3D ICs using Various Thermal Objectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No. 4, pp. 692-709, 2006.

[99] S. Lee, T. F. Lemczyk, and M. M. Yovanovich, "Analysis of Thermal Vias in High Density Interconnect Technology," *Eighth Annual IEEE Semiconductor Thermal Measurement and Management Symposium*, pp. 55-61, 1992.

[100] R. S. Li, "Optimization of Thermal Via Design Parameters Based on an Analytical Thermal Resistance Model," *Thermal and Thermomechanical Phenomena in Electronic Systems*, pp. 475-480, 1998.

[101] D. Pinjala, M. K. Iyer, C. S. Guan, and I. J. Rasiah, "Thermal Characterization of Vias Using Compact Models," *Proceedings of the Electronics Packaging Technology Conference*, pp. 144-147, 2000.

[102] Y. Yamaji, T. Ando, T. Morifuji, M. Tomisaka, M. Sunohara, T. Sato, and K. Takahashi, "Thermal characterization of Bare-Die Stacked Modules with Cu through-Vias," *Proceedings of the 51st Electronic Components and Technology Conference*, pp. 730-732, 2001.

[103] T.-Y. Chiang, K. Banerjee, and K. C. Saraswat, "Effect of Via Separation and Low-k Dielectric Materials on the Thermal Characteristics of Cu Interconnects," *International Electron Devices Meeting Technical Digest*, pp. 261-264, 2000.

[104] T.-Y. Chiang, K. Banerjee, and K. C. Saraswat, "Compact Modeling and SPICE-Based Simulation for Electrothermal Analysis of Multilevel ULSI Interconnects," *Proceedings of the International Conference on Computer-Aided Design*, pp. 165-172, 2001.

[105] T.-Y. Chiang, K. Banerjee, and K. C. Saraswat, "Analytical Thermal Model for Multilevel VLSI Interconnects Incorporating Via Effect," *IEEE Electron Device Letters*, Vol. 23, No. 1, pp. 31-33, 2002.

[106] www.tu-dresden.de/mwism/skalicky/laspack/laspack.html

[107] www.cbl.ncsu.edu/pub/Benchmark_dirs/LayoutSynth92

[108] S. Borkar, "Microarchitecture and Design Challenges for Gigascale Integration," *Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, 2004.

[109] S. Heo, K. Barr, and K. Asanovic, "Reducing Power Density through Activity Migration," *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 217-222, 2003.

[110] J. Deeney, "Thermal Modeling and Measurement of Large High Power Silicon Devices with Asymmetric Power Distribution," *35th International Symposium on Microelectronics*, 2002.

[111] H. B. Bakoglu, *Circuits, Interconnects and Packaging for VLSI*. Addison-Wesley: Reading MA, 1990.

[112] P. Saxena, N. Menezes, P. Cocchini, and D. A. Kirkpatrick, "Repeater Scaling and its Impact on CAD," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 4, pp. 451-463, 2004.

[113] J. Cong, T. Kong, and D. Z. Pan, "Buffer Block Planning for Interconnect-Driven Floorplanning," *Proceedings of the International Conference on Computer-Aided Design*, pp. 358-363, 1999.

[114] R. S. Tsay, and J. Koehl, "An Analytic Net Weighting Approach for Performance Optimization in Circuit Placement," *Proceedings of the Design Automation Conference*, pp. 620-625, 1991.

[115] H. Ren, D. Z. Pan, and D. S. Kung, "Sensitivity Guided Net Weighting for Placement Driven Synthesis," *Proceedings of the International Symposium on Physical Design*, pp. 10-17, 2004.

[116] K. Rajagopal, T. Shaked, Y. Parasuram, T. Cao, A. Chowdhary, and B. Halpin, "Timing Driven Force Directed Placement with Physical Net Constraints," *Proceedings of the International Symposium on Physical Design*, pp. 60-66, 2003.

[117] H. Vaishnav and M. Pedram, “PCUBE: A Performance Driven Placement Algorithm for Low-Power Design,” *Proceedings of the European Design Automation Conference*, pp. 72-77, 1993.

[118] P. Saxena, and B. Halpin, “Modeling Repeaters Explicitly Within Analytical Placement,” *Proceedings of the Design Automation Conference*, pp. 699-704, 2004.

[119] G. Sigl, K. Doll, and F. M. Johannes, “Analytical Placement: A Linear or a Quadratic Objective Function?” *Proceedings of the Design Automation Conference,* pp. 427-432, 1991.

[120] N. Akkiraju and M. Mohan, “Spec-based Flip-flop and Buffer Insertion,” *Proceedings of the International Conference on Computer-Aided Design,* pp. 270-275, 2003.

[121] C. J. Alpert, J. Hu, S. S. Sapatnekar, and P. Villarrubia, “A Practical Methodology for Early Buffer and Wire Resource Allocation,” *Proceedings of the Design Automation Conference*, pp. 189-194, 2001.