UNIVERSITY OF MINNESOTA

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

Yong Zhan

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

Professor Sachin S. Sapatnekar

_____

Name of Faculty Advisor

_____

Signature of Faculty Advisor

_____

Date

GRADUATE SCHOOL

HIGH EFFICIENCY ANALYSIS AND OPTIMIZATION ALGORITHMS IN ELECTRONIC DESIGN AUTOMATION

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

YONG ZHAN

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

Sachin S. Sapatnekar, Advisor

May 2007

©Yong Zhan 2007

Acknowledgments

I owe most of the gratitude to my advisor, Professor Sachin S. Sapatnekar. Without his guidance and generous support, I could have never proceeded so far in my education. During the past few years, Professor Sapatnekar has been the primary source of my knowledge in electronic design automation, and at the same time, he has also been my role model in performing research in a highly technical and competitive field. From him, I have learned how to identify problems with both scientific and technological significance, from him, I have learned how to carry out research in a systematic way, and from him, I have learned how to be strict in research and be meticulous in writing. The help and encouragement that Professor Sapatnekar has given me are countless, and the influence he has on me in terms of my attitude to research will benefit me significantly in my future career.

I would like to thank the members of my Ph.D. examining committee, Professor Kia Bazargan, Professor Ramesh Harjani, and Professor Sean Garrick, for the precious time they have spent in reviewing my thesis and the insightful questions they have raised during my exams.

I would also like to thank the former and present members of the VLSI electronic

design automation lab and Professor Bazargan's reconfigurable computing lab at the University of Minnesota, Shrirang Karandikar, Hongliang Chang, Yan Feng, Sanjay Kumar, Xianghong Liu, Pongstorn Maidee, Hushrav Mogal, Vidyasagar Nookala, Haifeng Qian, Jaskirat Singh, Satish Sivaswamy, Anup Sultania, and Tianpei Zhang for the help they have given me during my Ph. D. study and the happy time we have had together at the University.

Finally, I would like to thank my parents for their ever present love and support. If my education gives me the knowledge and experience required to build my future career, they give me the meaning.

Abstract

The remarkable progress that has been made in semiconductor manufacturing technologies during the past few years has significantly improved the performance of VLSI circuits. However, it has also created many new challenges in the field of computer aided design of electronic circuits. Some of the key issues that need to be addressed include providing CAD tools the capability of handling the effects that have become important in today's high performance circuits and developing high efficiency analysis and optimization algorithms so that they can be used to effectively design ICs that contain hundreds of millions of transistors.

In this thesis, we tackle several problems that are currently of interest to the CAD community, and special attention is paid to the efficiency of the algorithms being developed. The first part of the thesis focuses on the development of a high efficiency thermal simulation strategy for early stages of physical design. Several Green function-based thermal simulation algorithms are presented, and they can be used, respectively, to perform localized thermal analysis, full chip temperature profiling, and thermal simulations where the accuracy requirement differs from place to place over the same chip. The accuracy and efficiency of the algorithms are demonstrated through comparisons with the results from a commercial computational fluid dynamic (CFD) software for thermal analysis.

The second part of the thesis is concerned with the I/O pin limitation problem in

VLSI circuits with high performance and high integration density. Technology trend shows that while the current consumption of a VLSI chip will increase in the future due to the enhanced functionality, the number of package pins does not change significantly. As a result, the number of pins for each unit of current delivered is actually reduced in future ICs, which makes the pin limitation problem a new bottle neck in chip design. The stacked-Vdd circuit paradigm, in which modules are assigned to different Vdd domains so as to reduce the currents flowing through the power grids, is a good candidate in resolving the pin limitation problem. However, it is crucial to maintain the current balance between the modules operating in different Vdd domains so that the power consumption of the circuit is not unnecessarily increased. In this part of the thesis, a graph partition-based algorithm is developed to assign modules to different Vdd domains efficiently and experimental results show that it is much more effective in reducing the power waste in stacked-Vdd circuits than a bin-packing technique.

Finally, in the last part of the thesis, we address the issue of optimizing on-chip spiral inductors, which are key components in many RF integrated circuits (RFICs). A sequential quadratic programming (SQP) based approach is used to optimize the quality factor Q of inductors. The SQP technique has the advantage of being easily adaptable to any physical model of the inductor, and therefore is not limited by model accuracy. In addition, fast convergence can be expected if the starting point of the numerical iterations is not too far away from an optimum of the objective function.

To reduce the possibility that the algorithm is trapped at a local optimal point, we choose to run the SQP algorithm multiple times starting from randomly generated initial points, and experimental results show that high quality inductors can be obtained using this approach within a very reasonable amount of time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the major driving forces of the semiconductor industry is the capability of the foundries to manufacture progressively smaller features on chip, and therefore increase the device count per silicon area. Smaller feature size and higher device packing density are essential for improving the performance of VLSI circuits thanks to the reduced gate and interconnect delays. In Table 1.1, the technology trend as projected by the International Technology Roadmap for Semiconductors (ITRS) [Sem] is presented. It can be seen that by the year 2011, a state-of-the-art chip may contain over 700 million transistors and run at frequencies approaching 18GHz. In addition, because of the tremendous progress that has been made in CMOS technologies, circuit designers today routinely integrate heterogeneous functional units such as digital signal processors and RF front end circuitries over the same die so as to improve the cost

Year	Technology Node (nm)	Number of Transistors	Number of Wire Levels	f (GHz)	Vdd (V)	Power (W)
2005	54	193M	11	5.2	1.1	167
2006	48	193M	11	6.8	1.1	180
2007	42	386M	12	9.3	1.1	189
2008	38	386M	12	11.0	1.0	198
2009	34	386M	12	12.4	1.0	198
2010	30	773M	12	15.0	1.0	198
2011	27	773M	12	17.7	0.9	198

Table 1.1: Trends in IC technology parameters as projected by ITRS.

effectiveness of building complicated systems.

Although the future of the semiconductor industry appears promising, keeping the pace of progress that has been observed in the past and following the trend as projected by the ITRS need collaborated efforts between circuit designers, device and manufacturing engineers, and CAD tool developers. Due to the high complexity of VLSI circuits, the application of CAD tools has long been an indispensable component in circuit design, and with the demands for more functional but more complicated ICs continuously growing, CAD tools will play an even more important role in IC designs in the years to come.

There are several challenges facing CAD tool developers today. The first challenge concerns the incorporation into CAD tools the effects that have become critical for VLSI designs in the nanometer regime. Some of the prominent examples of these effects include process variation issues, thermal issues, and lithography related issues. These issues have only been of secondary importance during the past, and therefore

have not been the major considerations in many mainstream CAD tools. However, the shrinking feature size and escalating power consumption of contemporary VLSI circuits have made it imperative for CAD tools to incorporate these effects such that the design closure can be achieved with a reasonable amount of design effort. Another challenge that CAD tool developers must deal with is the increasing complexity of VLSI circuits. Nowadays, with designs containing hundreds of millions of transistors, the time complexity of a CAD algorithm used to assist the design process has become one of the most critical factors in determining the time-to-market of the final product. Therefore, the development of high efficiency CAD algorithms has gained significant interest in the CAD community. In this thesis, we will tackle several challenges that have been presented to CAD tool developers recently, and special attention is paid to the efficiency of the algorithms that are being developed.

The first part of the thesis focuses on the thermal analysis in physical design. As described previously, the technology trend in the semiconductor industry is to increase the device packing density and run circuits at higher frequencies. A side effect of this trend is that power consumptions of chips will escalate, which leads to elevated on-chip temperatures. High temperature could be detrimental to both the performance and reliability of circuits. As pointed out in [CLRK00], the delay of aluminum interconnect goes up by 30% when the temperature rises from 25°C to 100°C, and it was reported in [RBMH98] that the electromigration-induced mean-time-to-failure of

interconnect is reduced by 90% when the temperature increases from 25°C to 52.5°C. This situation has made it crucial to consider thermal issues during the design process so as to ensure the proper operation of the manufactured circuits. Generally, designers use thermal analysis softwares to verify that a design does not exceed the thermal budget, e.g., the maximum allowed on-chip temperature. This analysis can be performed in the final verification phase of the design if the circuit is not very complicated. For today's multi-million gate VLSI circuits, however, it is no longer practical to carry out the thermal analysis after the design is almost completed because if a thermal constraint is violated, it will be too expensive to go up the design hierarchy and fix the problem. This reality has made it obvious that thermal analysis should be incorporated into early stages of physical design such as thermal-aware floorplanning and placement where there is more flexibility in modifying the design and it is relatively inexpensive to fix the problems that arise. One of the key characteristics that a thermal analysis software must possess in order to be used effectively in early stages of physical design is high efficiency, since at these design stages, thermal analysis is often used as part of the simulation core of an optimization engine where a large design space of possible physical layouts must be explored and an independent calculation on temperature distribution has to be performed for each candidate layout. In this part of the thesis, a high efficiency thermal simulation strategy that can be used to facilitate the thermal analysis in early stages of physical design is developed. Three Green function-based

thermal simulation algorithms are presented. They take advantage of the special characteristics of the thermal problems encountered in chip design so as to improve the efficiency, and they can be used, respectively, to perform localized thermal analysis, full-chip temperature profiling, and thermal simulations where the accuracy requirement differs from place to place over the same chip. The accuracy and efficiency of the three algorithms are demonstrated through comparisons with the results from a commercial computational fluid dynamic (CFD) software for thermal analysis.

The second part of the thesis tackles the I/O pin limitation problem, which is an important issue in the design of high performance circuits that have enhanced functionalities and switch at high frequencies. I/O pins fall into two categories: those used to deliver signals and those that deliver power (Vdd or ground). While increased on-chip functionalities lead to a demand for more signal I/O pins, the demand for power is stronger. In contemporary and future technologies, one-half to two-thirds of all I/O pins must be dedicated to power delivery so as to reduce the worst case IR and $\mathrm{L}\frac{\mathrm{di}}{\mathrm{dt}}$ noise in the power grids, while the total number of package pins does not increase significantly. As a result, as the total current consumption of a chip goes up due to the increasing circuit complexity and higher switching frequency, each power pin must deliver a larger amount of current to the chip. This trend can be clearly seen in Fig. 1.1, which is created based on the data from ITRS [Sem]. In other words, as IC technologies advance, the number of pins for each unit of current delivered is actually reduced.

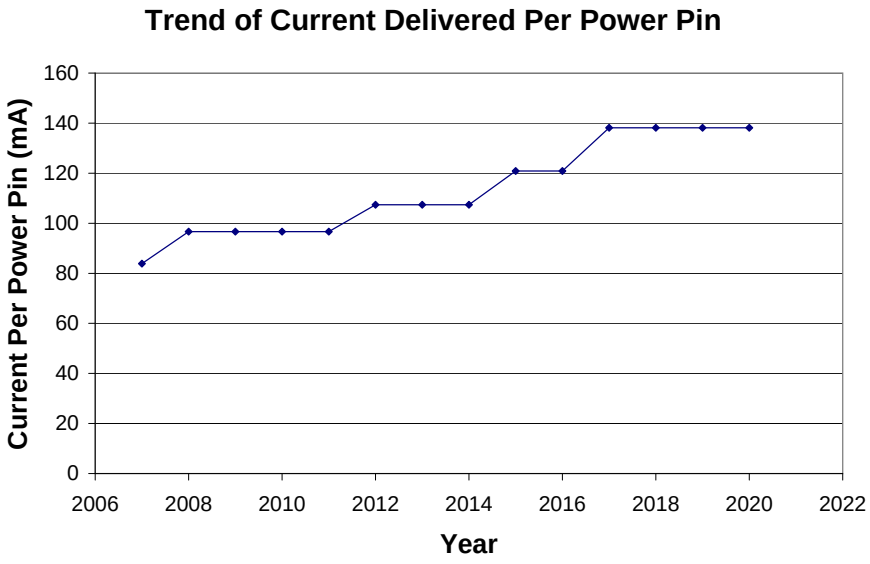


Figure 1.1: Trend of current delivered per power pin based on the data from ITRS.

The pin limitation problem is especially pronounced in the emerging 3D IC technology which has appeared recently as a promising alternative to the prevailing 2D IC technology. One of the primary advantages of the 3D IC technology is that by stacking multiple dies vertically while reducing the footprint area of each die, interconnect delays of long wires can be significantly reduced. Nevertheless, the pin limitation problem is exacerbated here: for a 3D chip with k tiers, the amount of current to be supplied is k times as much as for a 2D chip with the same footprint, but the number of available pins is essentially the same. Viewing this another way, if we were to trans-

form a 2D IC to a k-tier 3D IC implementing the same functionality, the number of pins accessible to the circuit will be reduced to $\frac{1}{k}$ of the original value because of the much smaller footprint area. This situation has made it imperative to consider the pin limitation issue in the design of 3D ICs.

One of the viable ways of alleviating the pin limitation problem is to use the stacked-Vdd circuit paradigm where power is delivered to the chip as multiples of the regular supply voltage and modules are distributed to different Vdd domains so that the total current flowing through the power grids is reduced, which brings down the number of power pins required to supply currents to the circuit. However, a significant amount of power may be wasted in a circuit designed using this paradigm if the currents consumed by the modules operating in different Vdd domains are not balanced. In this part of the thesis, we present a partition-based algorithm for efficiently assigning modules at the floorplanning level so as to maximally reduce the power waste during the operation of the circuit. Experimental results on a DLX architecture show that compared with assigning modules to different Vdd rails using a bin-packing technique, the circuit generated by our algorithm has about 32% less wasted power, on average, and therefore is more suitable for low power operations. In addition, experiments on a 3D IC example show that our module assignment approach is equally effective in reducing the power waste in 3D ICs.

The third part of the thesis is devoted to a problem that circuit designers have

encountered while integrating heterogeneous functional units such as digital and RF circuits over the same die. Experiences have shown that for some integrated RF devices, it is extremely difficult to make them as good as their discrete counterparts. The most pronounced example of this is the integrated spiral inductor whose quality factor Q is much lower than that of an equivalent discrete inductor. As a result, optimization has become indispensable during the design of on-chip inductors because designers must strive to find the optimal design parameters in order for the device to achieve a reasonable performance. High efficiency is again an important criterion in judging the usefulness of an inductor optimization algorithm because a complicated circuit may contain many integrated RF devices and a highly efficient optimization tool may significantly improve the throughput of the design process. In this part of the thesis, we will develop an efficient on-chip inductor optimization algorithm based on the sequential quadratic programming (SQP) technique. The SQP approach has the advantage of not relying on a specific form of the inductor model, and therefore is not restricted by model accuracy. In addition, high convergence rate can be expected if the starting point of the numerical iterations is not too far away from an optimum of the objective function. To reduce the possibility that the algorithm is trapped at a local optimal point, we choose to run the SQP algorithm multiple times starting from randomly generated initial points, and experimental results show that high quality inductors can be obtained using this approach within a very reasonable amount of time.

Chapter 2

High Efficiency Thermal Simulation Algorithms

2.1 Overview of Thermal Simulation Algorithms

As pointed out in the previous chapter, thermal simulation has become an indispensable component of many physical design processes during the past few years due to the escalating power consumption of today's high performance VLSI circuits and the resulting elevated on-chip temperature. In a nutshell, thermal simulation involves the solution of a parabolic partial differential equation (PDE)

$$\frac{\partial T(\mathbf{r},t)}{\partial t} = \frac{k}{\rho c_p}\nabla^2 T(\mathbf{r},t) + \frac{g(\mathbf{r},t)}{\rho c_p} \tag{2.1}$$

under an appropriate set of boundary conditions, where T, t, g, k, and c_p are the temperature distribution, time, power density distribution, thermal conductivity, and heat capacity of the chip material, respectively. Based on the type of analysis they perform, thermal simulation algorithms can be generally divided into two categories, i.e., those for transient analysis and those for steady-state analysis.

Transient analysis is concerned with the time evolution of the temperature distribution within a chip given a time-varying power density distribution, and can be performed efficiently using the thermal ADI algorithm proposed by Wang *et al.* in [WC02]. To understand how the ADI algorithm works, we must first discretize the PDE (2.1) in both the time and space domains. For simplicity, we assume that the time domain discretization takes a backward Euler scheme and the space domain discretization is carried out using finite differentiation. The discretized PDE is given in the form

$$\frac{T_{i,j,k}^{n+1} - T_{i,j,k}^{n}}{\Delta t} = \frac{k}{\rho c_p}\left[\frac{T_{i+1,j,k}^{n+1} - 2T_{i,j,k}^{n+1} + T_{i-1,j,k}^{n+1}}{(\Delta x)^2} + \frac{T_{i,j+1,k}^{n+1} - 2T_{i,j,k}^{n+1} + T_{i,j-1,k}^{n+1}}{(\Delta y)^2} + \frac{T_{i,j,k+1}^{n+1} - 2T_{i,j,k}^{n+1} + T_{i,j,k-1}^{n+1}}{(\Delta z)^2}\right] + \frac{g_{i,j,k}^{n}}{\rho c_p} \tag{2.2}$$

where $T_{i,j,k}^{n} = T(i\Delta x, j\Delta y, k\Delta z, n\Delta t)$ and $g_{i,j,k}^{n} = g(i\Delta x, j\Delta y, k\Delta z, n\Delta t)$ are the temperature and power density at node $(i\Delta x, j\Delta y, k\Delta z)$ at time step $n\Delta t$, respectively. The node indexing scheme is visualized in Fig. 2.1. A linear equation similar to (2.2) can be written for each node, and the resulting system of linear equations can

be solved to obtain the temperature distribution at time step $(n+1)\Delta t$ given the temperature distribution at time step $n\Delta t$. The time complexity of solving this system of linear equations is superlinear in terms of the number of nodes.

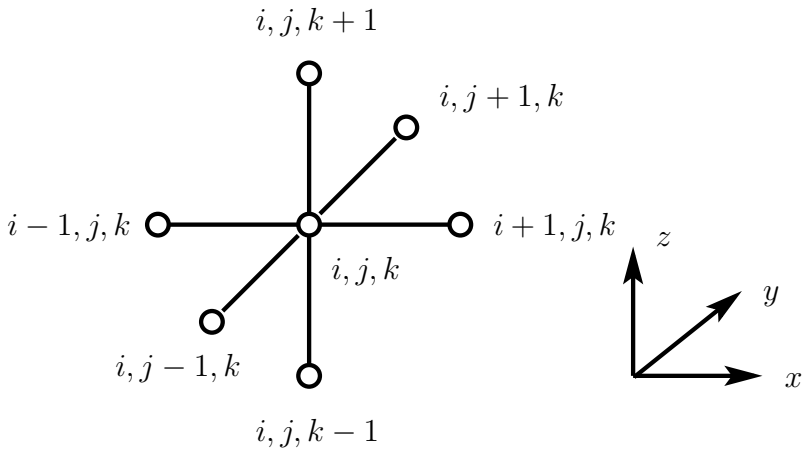


Figure 2.1: The node indexing scheme for the spatial domain discretization.

To accelerate the solution process, the thermal ADI algorithm takes a different approach in finding the temperature distribution at time step $(n+1)\Delta t$ given the temperature distribution at time step $n\Delta t$. Instead of solving the system of linear equations corresponding to (2.2) directly, it divides each time step into three sub-steps and solves three systems of linear equations in tandem. In mathematical form, these three systems of linear equations can be represented by

Step I

$$\frac{T_{i,j,k}^{n+\frac{1}{3}} - T_{i,j,k}^{n}}{\Delta t} = \frac{k}{\rho c_p}\left[\frac{T_{i+1,j,k}^{n+\frac{1}{3}} - 2T_{i,j,k}^{n+\frac{1}{3}} + T_{i-1,j,k}^{n+\frac{1}{3}}}{(\Delta x)^2} + \frac{T_{i,j+1,k}^{n} - 2T_{i,j,k}^{n} + T_{i,j-1,k}^{n}}{(\Delta y)^2} + \frac{T_{i,j,k+1}^{n} - 2T_{i,j,k}^{n} + T_{i,j,k-1}^{n}}{(\Delta z)^2}\right] + \frac{g_{i,j,k}^{n}}{\rho c_p} \tag{2.3}$$

Step II

$$\frac{T_{i,j,k}^{(n+\frac{2}{3})} - T_{i,j,k}^{(n+\frac{1}{3})}}{\Delta t} = \frac{k}{\rho c_p}\left[\frac{T_{i+1,j,k}^{n+\frac{1}{3}} - 2T_{i,j,k}^{n+\frac{1}{3}} + T_{i-1,j,k}^{n+\frac{1}{3}}}{(\Delta x)^2} + \frac{T_{i,j+1,k}^{n+\frac{2}{3}} - 2T_{i,j,k}^{n+\frac{2}{3}} + T_{i,j-1,k}^{n+\frac{2}{3}}}{(\Delta y)^2} + \frac{T_{i,j,k+1}^{n+\frac{1}{3}} - 2T_{i,j,k}^{n+\frac{1}{3}} + T_{i,j,k-1}^{n+\frac{1}{3}}}{(\Delta z)^2}\right] + \frac{g_{i,j,k}^{n}}{\rho c_p} \tag{2.4}$$

Step III

$$\frac{T_{i,j,k}^{(n+1)} - T_{i,j,k}^{(n+\frac{2}{3})}}{\Delta t} = \frac{k}{\rho c_p}\left[\frac{T_{i+1,j,k}^{n+\frac{2}{3}} - 2T_{i,j,k}^{n+\frac{2}{3}} + T_{i-1,j,k}^{n+\frac{2}{3}}}{(\Delta x)^2} + \frac{T_{i,j+1,k}^{n+\frac{2}{3}} - 2T_{i,j,k}^{n+\frac{2}{3}} + T_{i,j-1,k}^{n+\frac{2}{3}}}{(\Delta y)^2} + \frac{T_{i,j,k+1}^{n+1} - 2T_{i,j,k}^{n+1} + T_{i,j,k-1}^{n+1}}{(\Delta z)^2}\right] + \frac{g_{i,j,k}^{n}}{\rho c_p} \tag{2.5}$$

It is easy to see that in each sub-step, the system of linear equations to be solved is implicit in only one spatial direction. Therefore, the corresponding coefficient matrix is tridiagonal, and it is well known that a tridiagonal linear system can be solved in linear time. As a result, the time complexity of obtaining the temperature distribution at time step $(n+1)\Delta t$ given the temperature distribution at time step $n\Delta t$ becomes

linear in terms of the number of nodes, which is a significant improvement over solving (2.2) directly. In the original thermal ADI work, the Crank-Nicolson scheme was used for the discretization in time domain to provide added numerical accuracy.

Steady-state analysis, on the other hand, is interested in the stabilized temperature distribution given a time-independent power density distribution or a power density distribution averaged over time. In this chapter, we will focus on the steady-state thermal analysis. Mathematically, a steady-state thermal analysis problem can be considered as solving the Poisson's equation

$$\nabla^2 T(\mathbf{r}) = -\frac{g(\mathbf{r})}{k} \tag{2.6}$$

under an appropriate set of boundary conditions. During the past few years, several steady-state thermal simulation algorithms have been used in chip design. The finite difference method (FDM) [TK00] discretizes the differential operator ∇^2 in (2.6) so that the differential equation becomes

$$\frac{T_{i+1,j,k} - 2T_{i,j,k} + T_{i-1,j,k}}{(\Delta x)^2} + \frac{T_{i,j+1,k} - 2T_{i,j,k} + T_{i,j-1,k}}{(\Delta y)^2} + \frac{T_{i,j,k+1} - 2T_{i,j,k} + T_{i,j,k-1}}{(\Delta z)^2}$$
$$= -\frac{g_{i,j,k}}{k} \tag{2.7}$$

where $T_{i,j,k} = T(i\Delta x, j\Delta y, k\Delta z)$ and $g_{i,j,k} = g(i\Delta x, j\Delta y, k\Delta z)$ are the temperature

and power density at node $(i\Delta x, j\Delta y, k\Delta z)$, respectively. Then the finite difference equations corresponding to each of the nodes in the space domain are collected and the resulting system of linear equations are solved to obtain the temperature distribution. In [TK00], a thermal circuit analogous to an electrical network was used to visualize the discretization scheme shown in (2.7).

A competing method that has been used in chip design to perform the steady-state thermal analysis is the finite element method (FEM) [GS03]. The FEM is based on the observation that if a function $T(\mathbf{r})$ solves equation (2.6), it will also minimize the functional

$$I(T(\mathbf{r})) = \frac{1}{2}\int_V [k\nabla^2 T(\mathbf{r}) - 2g(\mathbf{r})T(\mathbf{r})]dV - \int_{\Gamma_q} q(\mathbf{r})T(\mathbf{r}) + \int_{\Gamma_h} [hT_\infty T(\mathbf{r}) - \frac{1}{2}hT^2(\mathbf{r})] \tag{2.8}$$

where the integration over Γ_q corresponds to portions of the boundary where the heat flux boundary conditions are enforced, and the integration over Γ_h corresponds to portions of the boundary where the convective boundary conditions are enforced. Finding the exact function $T(\mathbf{r})$ so that $I(T(\mathbf{r}))$ in (2.8) is minimized is difficult. Therefore, the FEM tries to find a function that minimizes $I(T(\mathbf{r}))$ within a much reduced functional space. To achieve this objective, the FEM meshes the space domain of the problem and constructs $T(\mathbf{r})$ using a linear combination of a set of local basis functions. The coefficient of each basis function approximates the temperature at a particular node within

the mesh. If we set the first derivative of $I(T(\mathbf{r}))$ with respect to each coefficient of the basis functions to zero, we will obtain a system of linear equations

$$K\mathbf{T} = \mathbf{P} \tag{2.9}$$

where K is the global stiffness matrix, $\mathbf{T}$ is the vector of unknown nodal temperatures, and $\mathbf{P}$ is the vector representing the power distribution. Equation (2.9) can be solved using any linear solver to obtain the temperature distribution.

The advantages of the FDM and FEM include their robustness and high accuracy. In addition, the FEM also possesses the capability of handling complicated boundary conditions. The primary drawback of the FDM and FEM rests on the fact that they always require volume meshing of the entire substrate even though the devices are usually fabricated only in a thin layer close to the top surface of the IC chip. Hence, even for the cases where only the temperature distribution within the device layer is of interest, we still have to solve a large system of linear equations corresponding to the volume meshing, which leads to low efficiency. In [LPAC04], a thermal simulation algorithm based on the solution of the finite difference equations using the multigrid approach was proposed, and its high efficiency has made the full-chip thermal simulation practical for the optimizations in physical designs.

The boundary element method (BEM) constitutes another class of thermal simula-

tion algorithms in which the volume meshing of the substrate is avoided. An important underlying concept in the BEM is the Green function, which describes the temperature distribution within the chip when a unit point power source is present. For the simple geometries encountered in chip design, the explicit form of the Green function can be obtained, and the temperature field under an arbitrary power density distribution can be calculated by integrating the corresponding Green function. Because the BEM only meshes the power generating surfaces in thermal simulations as opposed to the meshing of the entire substrate by the FDM and FEM, it naturally leads to a smaller problem size, and hence has the potential of achieving high efficiency. However, the actual runtime of an algorithm implemented using the BEM depends critically on how efficient the Green function is evaluated and how the temperature distribution is calculated given the power density distribution. In [HS90], the classical Green function approach was used in thermal simulations where the Green function was utilized directly to evaluate the temperature field in a rectangular-shaped substrate. Because the underlying Green function is expressed as a multiple-infinite summation and it has to be truncated at high indices in actual implementations to maintain a reasonable accuracy, the efficiency of this method is rather low. In [CK99], the method of images was used to obtain the Green function in closed form at the expense of relaxing the boundary conditions by assuming that the chip is infinitely large horizontally. The advantage of this method is that the Green function can be computed on-line efficiently

and thus it is suitable for optimization purposes. However, by assuming that the chip is infinitely large horizontally, the on-chip temperature will be severely under-estimated especially near the boundaries of the actual chip, although the locations of the hot spots can be correctly identified as shown in [CK99]. In [WM04], an efficient algorithm for evaluating the temperature field in VLSI chips using a semi-analytical form of the Green function was proposed which takes into account the multilayered nature of the semiconductor substrates used in IC fabrications. Nevertheless, this method also assumes that the chip is infinitely large horizontally, and therefore it has the same problem as [CK99].

Note that the computation of the steady-state temperature distribution T in thermal problems is very similar to the computation of the potential field ϕ in electrical problems. Both T and ϕ satisfy the Poisson's equation, and the power source P in thermal problems corresponds to the charge q in electrical problems. In [GM96] and [NGM98], the discrete cosine transform (DCT) is combined with a table look-up approach to improve the efficiency of using the Green function to calculate the electrical potential distribution within a rectangular-shaped substrate. In this method, the multiple-infinite summation contained in the expression of the Green function is not evaluated on-line. Instead, look-up table and vectors are established in advance so that each evaluation of the Green function is reduced to the summation of a constant and 80 terms in the look-up table and vectors. This is a significant improvement over the direct evaluation

of the multiple-infinite summation in the classical Green function method, which may involve thousands or even more terms to ensure a reasonable accuracy. Since the look-up table and vectors only have to be computed once for each technology and substrate geometry, but are independent of where the devices are located on the chip, they can be obtained in the pre-characterization phase of the design and used many times in the optimization process. As a result, the amortized cost of establishing the look-up table and vectors can be ignored in practice. Our first thermal simulation algorithm (Algorithm I) follows a similar line of analysis as in [GM96] and [NGM98]. The difference is that since the boundary conditions encountered in thermal problems are different from those in electrical problems, the Green function and the look-up table and vectors must be re-derived to reflect the special characteristics of the thermal problems.

The improvement in efficiency of Algorithm I, as compared with that of the classical Green function method, comes from its faster evaluation of the expressions involving the Green function in calculating the temperature field, and compared with other fast algorithms such as the ones presented in [CK99] and [WM04], our algorithm can achieve a much higher accuracy because it does not assume that the chip is infinitely large horizontally, and hence it can take the proper boundary conditions into consideration. Asymptotically, however, the classical Green function method, the algorithms in [CK99] and [WM04], and our Algorithm I all have the same time complexity of $O(\mathcal{N}_s \cdot \mathcal{N}_f)$, where $\mathcal{N}_s$ is the number of power source regions and $\mathcal{N}_f$ is the number of

temperature field regions. For cell level full-chip thermal simulations where the number of heat sources is large and the temperature profile over the entire chip is sought, however, a still faster algorithm is required.

In [CCS99], Costa *et al.* proposed an elegant algorithm for efficiently performing the full-chip electrical potential profiling, which is a key step in solving substrate parasitic extraction problems. This algorithm combines the concept of functional eigendecomposition with the technique of the DCT to reduce the overall runtime of full-chip electrical potential profiling from $O(\mathcal{N}_{gc}^2)$ to $O(\mathcal{N}_{gc} \times \log(\mathcal{N}_{gc}))$, where $\mathcal{N}_{gc}$ is the total number of grid cells. Because of the parallelism between the thermal problem and the electrical problem, we can use a similar approach to reduce the asymptotic runtime of full-chip temperature profiling. We have implemented such an approach in our second algorithm (Algorithm II), and we will present it from the perspective of spectral domain computations that are familiar to engineers. Note that the temperature distribution can be obtained by convolving the power density distribution with the underlying Green function, and it is well known that convolutions in the space domain correspond to point-wise multiplications in the spectral domain. Therefore, using the spectral domain computations in conjunction with the DCT for transforming the data between space and spectral domains, we will be able to significantly reduce the runtime of full-chip temperature profiling. Our algorithm takes a piece-wise constant power density map as the input and generates a piece-wise constant temperature map as the output.

The primary steps of the algorithm include:

1. Obtaining the spectral domain representation of the power density map using the 2D DCT. The order of the DCT expansion is determined dynamically by the power density map instead of being set *a priori* to ensure the accuracy.

2. Calculating the spectral domain representation of the temperature map by multiplying each spectral component of the power density map by the corresponding spectral response of the linear system determined by the Green function.

3. Using a 2D inverse discrete cosine transform (IDCT) to obtain the temperature map from its spectral domain representation.

Both the 2D DCT and the 2D IDCT can be calculated efficiently using the 2D fast Fourier transform (FFT). The asymptotic time complexity of the overall algorithm is $O(\mathcal{N}_{gs}\times\log(\mathcal{N}_{gs})) + O(\mathcal{N}_{gf}\times\log(\mathcal{N}_{gf}))$, where $\mathcal{N}_{gs}$ and $\mathcal{N}_{gf}$ are the number of grid cells in the power source layer and temperature field layer, respectively. Hence, for calculating the full-chip temperature profile, the time complexity of Algorithm II is much smaller than that of Algorithm I, which is $O(\mathcal{N}_{gs}\cdot\mathcal{N}_{gf})$. Note that the lower asymptotic time complexity of Algorithm II does not invalidate the usefulness of Algorithm I because, as will be elaborated in Section 2.3.4, Algorithm I often works better for localized analysis, where the effects of a few critical circuit blocks on the temperature distribution in a few key regions are of interest.

Our third algorithm (Algorithm III) is a combination of Algorithm I and II, and it possesses the capability of performing thermal simulations where the accuracy requirement differs from place to place over the same chip, e.g., in mixed signal designs where analog circuits are fabricated on the same chip as digital circuits, the analog blocks often have more stringent accuracy requirements on thermal simulations because the operations of the analog circuits are more sensitive to temperature. Algorithm III reflects the idea of the pre-corrected FFT, which has been used extensively in the IC parasitic extraction works [PW97] [HBZ+03] [CCS98]. The algorithm first uses coarse grids to divide the source and field planes where each grid cell in the source plane can contain several logic gates or analog functional units, and the size of each grid cell in the field plane satisfies the accuracy requirements of the digital circuits. The power density of each grid cell in the source plane can be obtained by adding up the contributions from the logic gates and analog functional units that are located in it. A coarse temperature map for the field plane is then obtained from the coarse power density map using Algorithm II and is used for the digital blocks. Finally, for each analog functional unit on the field plane whose temperature is to be calculated more accurately, we use Algorithm I to compute the contributions to its temperature rise from the nearby logic gates and analog function units on the source plane, and use this result to correct the temperature obtained by Algorithm II over the coarse grid cell.

Our algorithms are all implemented in C++ and experimental results show that

they can achieve relative errors of around 1% compared with that of a commercial computational fluid dynamic (CFD) software package for thermal analysis, while their efficiencies are orders of magnitude higher than that of the classical Green function method. The rest of the chapter will be organized as follows. In Section 2.2, we formulate the temperature field computation problem and present the concept of Green function for thermal problems. In Section 2.3, we discuss in detail the three thermal simulation algorithms. Section 2.4 shows the experimental results, and the summary is provided in Section 2.5.

2.2 Problem Formulation and the Green Function for Thermal Problems

2.2.1 Problem formulation

Fig. 2.2(a) shows an IC chip with the associated packaging, and Fig. 2.2(b) shows a schematic of the structure in Fig. 2.2(a) where the packaging including the heat spreader and the heat sink has been simplified but the multilayered structure of the chip is explicitly shown. As stated in the previous section, the steady-state temperature

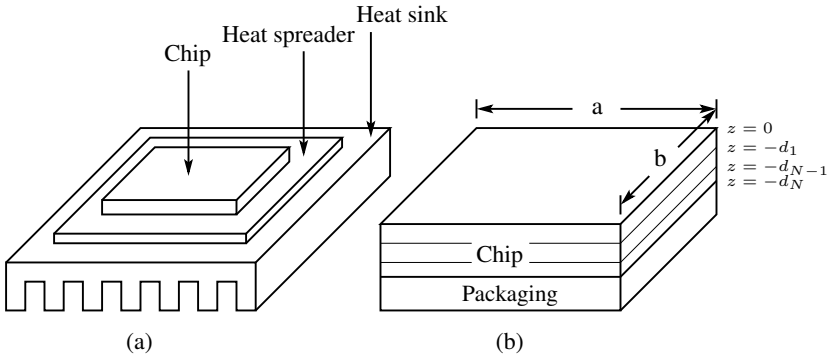


Figure 2.2: Schematic of a VLSI chip with packaging (a) IC chip and the packaging structure (b) simplified model of the chip and packaging.

distribution inside the chip is governed by Poisson's equation

$$\nabla^2 T(\mathbf{r}) = -\frac{g(\mathbf{r})}{k_{l(\mathbf{r})}} \tag{2.10}$$

where $\mathbf{r} = (x, y, z)$, $T(\mathbf{r})$ is the temperature (°C) distribution inside the chip, $g(\mathbf{r})$ is the volume power density (W/m^3), and $k_{l(\mathbf{r})}$ is the thermal conductivity (W/(m·°C)) of the layer where point $\mathbf{r}$ is located [Ozi68]. The vertical surfaces and the top surface of the chip are assumed to be adiabatic [Kok74], and the bottom surface of the chip is assumed to be convective, with an effective heat transfer coefficient h (W/(m^2·°C)) [CRT+98]. In mathematical form, these boundary conditions can be ex-

pressed as

$$\left.\frac{\partial T(\mathbf{r})}{\partial x}\right|_{x=0,a} = \left.\frac{\partial T(\mathbf{r})}{\partial y}\right|_{y=0,b} = 0 \tag{2.11}$$

$$\left.\frac{\partial T(\mathbf{r})}{\partial z}\right|_{z=0} = 0 \tag{2.12}$$

$$k_N \left.\frac{\partial T(\mathbf{r})}{\partial z}\right|_{z=-d_N} = h\left(T(\mathbf{r})|_{z=-d_N} - T_a\right) \tag{2.13}$$

where T_a is the ambient temperature, and k_N is the thermal conductivity of the bottom layer of the chip. In addition, we enforce the continuity conditions at the interface between adjacent layers within the multilayered chip, i.e.,

$$T(\mathbf{r})|_{z=-d_i+\epsilon} = T(\mathbf{r})|_{z=-d_i-\epsilon} \tag{2.14}$$

$$k_i \left.\frac{\partial T(\mathbf{r})}{\partial z}\right|_{z=-d_i+\epsilon} = k_{i+1} \left.\frac{\partial T(\mathbf{r})}{\partial z}\right|_{z=-d_i-\epsilon} \tag{2.15}$$

where ϵ is an infinitesimally small quantity and k_i is the thermal conductivity of the i^{th} material layer in the multilayered chip structure.

2.2.2 Green function for the rectangular-shaped multilayered structure

Let $G(\mathbf{r}, \mathbf{r}')$, with $\mathbf{r} = (x, y, z)$ and $\mathbf{r}' = (x', y', z')$, be the distribution of temperature above T_a in the multilayer when a unit point power source of 1W is placed at

position $\mathbf{r}'$. Then $G(\mathbf{r}, \mathbf{r}')$ satisfies the equation

$$\nabla^2 G(\mathbf{r}, \mathbf{r}') = -\frac{\delta(\mathbf{r} - \mathbf{r}')}{k_{l(\mathbf{r})}} \tag{2.16}$$

and the boundary conditions

$$\left.\frac{\partial G(\mathbf{r}, \mathbf{r}')}{\partial x}\right|_{x=0,a} = \left.\frac{\partial G(\mathbf{r}, \mathbf{r}')}{\partial y}\right|_{y=0,b} = 0 \tag{2.17}$$

$$\left.\frac{\partial G(\mathbf{r}, \mathbf{r}')}{\partial z}\right|_{z=0} = 0 \tag{2.18}$$

$$k_N \left.\frac{\partial G(\mathbf{r}, \mathbf{r}')}{\partial z}\right|_{z=-d_N} = hG(\mathbf{r}, \mathbf{r}')|_{z=-d_N} \tag{2.19}$$

$$G(\mathbf{r}, \mathbf{r}')|_{z=-d_i+\epsilon} = G(\mathbf{r}, \mathbf{r}')|_{z=-d_i-\epsilon} \tag{2.20}$$

$$k_i \left.\frac{\partial G(\mathbf{r}, \mathbf{r}')}{\partial z}\right|_{z=-d_i+\epsilon} = k_{i+1} \left.\frac{\partial G(\mathbf{r}, \mathbf{r}')}{\partial z}\right|_{z=-d_i-\epsilon} \tag{2.21}$$

where $\delta(\mathbf{r} - \mathbf{r}') = \delta(x - x')\delta(y - y')\delta(z - z')$ is the three-dimensional Dirac delta function, and $G(\mathbf{r}, \mathbf{r}')$ is the Green function. The temperature field under an arbitrary power density distribution can be obtained easily as

$$T(\mathbf{r}) = T_a + \int_0^a dx' \int_0^b dy' \int_{-d_N}^0 dz' G(\mathbf{r}, \mathbf{r}') g(\mathbf{r}') \tag{2.22}$$

As shown in [GM96] and [NGM98] for electrical problems, the Green function can be

generally written in the form

$$G(\mathbf{r},\mathbf{r}') = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)\cos\left(\frac{m\pi x'}{a}\right)\cos\left(\frac{n\pi y'}{b}\right) Z'_{mn}(z,z') \tag{2.23}$$

where $Z'_{mn}(z,z')'s$ are functions of only the z coordinates of the source and field points. The specific form of each $Z'_{mn}(z,z')$ depends on the boundary conditions, and it can be derived similarly to that shown in [GM96] and [NGM98]. For completeness, the detailed derivation of the Green function suitable for thermal problems is presented in Appendix A.

In the following analysis, we assume that both the heat sources and the field regions are located on discrete horizontal planes. Since the vertical dimensions of the devices are much smaller than that of the silicon chip, this assumption is reasonable for most practical purposes. For a particular pair of source and field planes, i.e., for a particular z and z', the Green function can be written as

$$G(x,y,x',y') = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} C_{mn}\cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)\cos\left(\frac{m\pi x'}{a}\right)\cos\left(\frac{n\pi y'}{b}\right) \tag{2.24}$$

The temperature distribution on the field plane due to the heat sources on the source

plane is given by

$$T(x,y) = T_a + \int_0^a dx' \int_0^b dy' G(x,y,x',y') P_d(x',y') \tag{2.25}$$

where $P_d(x',y')$ is the power density distribution on the source plane.

2.3 Green Function Based Thermal Simulation Algorithms

2.3.1 Algorithm I: Thermal simulation using the DCT and table look-up

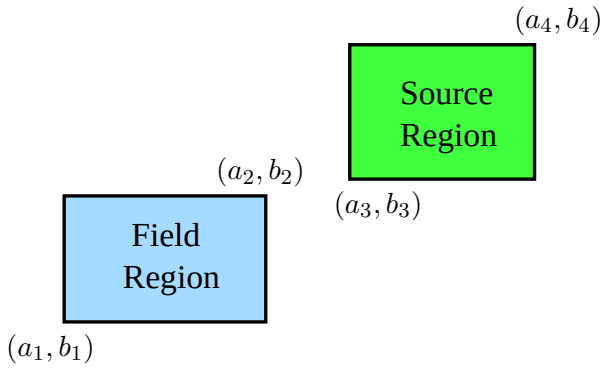


Figure 2.3: Source and field regions for computing the temperature distribution.

Since practically all of the on-chip geometries can be decomposed into combinations of rectangles, we only focus on the rectangular-shaped source and field regions

in the following analysis. Fig. 2.3 shows a schematic of a source and a field region. Note that the two regions can have different z coordinates if the field plane does not coincide with the source plane. Our objective here is to calculate the average temperature $\overline{T_f}$ of the field region efficiently given the power density P_d of the source region. To simplify the analysis, we assume that P_d is a constant within the source region. This is not a very restrictive assumption, since if the power density is not uniformly distributed in the source region, we can always divide the source region into smaller rectangular-shaped sub-regions and assume that the power density is uniform within each sub-region.

The average temperature in the field region can be computed using

$$\overline{T_f} = \frac{1}{(a_2 - a_1)(b_2 - b_1)} \int_{a_1}^{a_2} dx \int_{b_1}^{b_2} dy T(x, y) \tag{2.26}$$

Substituting (2.24) and (2.25) into (2.26), and modifying the integration limits of (2.25)

according to the location and dimensions of the source region, we obtain

$$\overline{T_f} = T_a + \frac{P_d}{(a_2-a_1)(b_2-b_1)} \times \int_{a_1}^{a_2} dx \int_{b_1}^{b_2} dy \int_{a_3}^{a_4} dx' \int_{b_3}^{b_4} dy' G(x,y,x',y')$$

$$= T_a + C_{00} P_d (a_4-a_3)(b_4-b_3) +$$

$$\left\{ \frac{P_d(b_4-b_3)}{(a_2-a_1)} \sum_{m=0}^{\infty} D_{m0} \left[\sin\left(\frac{m\pi a_2}{a}\right) - \sin\left(\frac{m\pi a_1}{a}\right) \right] \left[\sin\left(\frac{m\pi a_4}{a}\right) - \sin\left(\frac{m\pi a_3}{a}\right) \right] \right\} +$$

$$\left\{ \frac{P_d(a_4-a_3)}{(b_2-b_1)} \sum_{n=0}^{\infty} E_{0n} \left[\sin\left(\frac{n\pi b_2}{b}\right) - \sin\left(\frac{n\pi b_1}{b}\right) \right] \left[\sin\left(\frac{n\pi b_4}{b}\right) - \sin\left(\frac{n\pi b_3}{b}\right) \right] \right\} +$$

$$\left\{ \frac{P_d}{(a_2-a_1)(b_2-b_1)} \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} F_{mn} \left[\sin\left(\frac{m\pi a_2}{a}\right) - \sin\left(\frac{m\pi a_1}{a}\right) \right] \times \right.$$

$$\left[\sin\left(\frac{m\pi a_4}{a}\right) - \sin\left(\frac{m\pi a_3}{a}\right) \right] \left[\sin\left(\frac{n\pi b_2}{b}\right) - \sin\left(\frac{n\pi b_1}{b}\right) \right] \times$$

$$\left. \left[\sin\left(\frac{n\pi b_4}{b}\right) - \sin\left(\frac{n\pi b_3}{b}\right) \right] \right\} \tag{2.27}$$

where

$$D_{m0} = \begin{cases} C_{m0} \left(\frac{a}{m\pi}\right)^2 & \text{if } m \neq 0 \\ 0 & \text{if } m = 0 \end{cases} \tag{2.28}$$

$$E_{0n} = \begin{cases} C_{0n} \left(\frac{b}{n\pi}\right)^2 & \text{if } n \neq 0 \\ 0 & \text{if } n = 0 \end{cases} \tag{2.29}$$

$$F_{mn} = \begin{cases} C_{mn} \left(\frac{a}{m\pi}\right)^2 \left(\frac{b}{n\pi}\right)^2 & \text{if } m \neq 0, n \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.30}$$

Using the identity

$$\sin(\theta_1)\sin(\theta_2) = \frac{1}{2}(\cos(\theta_1 - \theta_2) - \cos(\theta_1 + \theta_2)) \tag{2.31}$$

the first summation

$$\sum_{m=0}^{\infty} D_{m0} \left[\sin\left(\frac{m\pi a_2}{a}\right) - \sin\left(\frac{m\pi a_1}{a}\right)\right] \left[\sin\left(\frac{m\pi a_4}{a}\right) - \sin\left(\frac{m\pi a_3}{a}\right)\right] \tag{2.32}$$

can be re-written as a sum of eight terms in the form

$$\pm\frac{1}{2}\sum_{m=0}^{\infty} D_{m0}\cos\left(\frac{m\pi(a_i \pm a_j)}{a}\right) \tag{2.33}$$

where $i = 1, 2$ and $j = 3, 4$.

To utilize the DCT, we first discretize the source and field planes into M equal divisions along the x direction and N equal divisions along the y direction and form the grids. Then we truncate the summation in equation (2.33) at index M. As will be discussed later, the indices M and N are determined by the considerations of both the resolution of thermal analysis and the convergence of the Green function. If we assume that all the vertices of the field and source regions are located on grid points, i.e., $\frac{a_i}{a} = \frac{k_i}{M}$, $\frac{a_j}{a} = \frac{k_j}{M}$, where k_i and k_j are integers, and $0 \leq k_i \leq M$, $0 \leq k_j \leq M$, then

equation (2.33) becomes

$$\pm\frac{1}{2}\sum_{m=0}^{M} D_{m0}\cos\left(\frac{m\pi(k_i\pm k_j)}{M}\right) \tag{2.34}$$

Let

$$k=\begin{cases} k_i\pm k_j & \text{if } 0\leq k_i\pm k_j\leq M \\ -(k_i\pm k_j) & \text{if } k_i\pm k_j<0 \\ 2M-(k_i\pm k_j) & \text{if } k_i\pm k_j>M \end{cases} \tag{2.35}$$

then $0\leq k\leq M$ and equation (2.34) can be re-written as

$$\pm\frac{1}{2}\sum_{m=0}^{M} D_{m0}\cos\left(\frac{m\pi k}{M}\right) \tag{2.36}$$

This is precisely one term in the type-I DCT of the sequence D_{m0}, and the DCT sequence can be computed efficiently using the FFT in $O(M\log(M))$ time [OSB99]. After the DCT sequence is obtained, it can be stored in a vector and used many times in future temperature calculations. As a result, the computation of summation (2.32) is reduced to eight look-ups in the DCT vector in constant time and then adding up the look-up results. Similarly, the summation involving E_{0n} in equation (2.27) can also be obtained efficiently using the DCT and table look-ups.

The double summation in equation (2.27) can be re-written as a sum of 64 terms in

the form

$$\pm\frac{1}{4}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty}F_{mn}\cos\left(\frac{m\pi(a_i\pm a_j)}{a}\right)\cos\left(\frac{n\pi(b_p\pm b_q)}{b}\right) \tag{2.37}$$

where $i = 1, 2$, $j = 3, 4$, $p = 1, 2$, and $q = 3, 4$. Using a similar approach, equation (2.37) can be cast into

$$\pm\frac{1}{4}\sum_{m=0}^{M}\sum_{n=0}^{N}F_{mn}\cos\left(\frac{m\pi k}{M}\right)\cos\left(\frac{n\pi l}{N}\right) \tag{2.38}$$

where $0{\leq}k{\leq}M$ and $0{\leq}l{\leq}N$. This is one term in the 2-D type-I DCT of the matrix F_{mn}. The 2-D DCT matrix can be computed using the FFT in $O((M{\cdot}N)\times\log(M{\cdot}N))$ time, and after the 2-D DCT table is obtained, the double summation reduces to 64 table look-ups in constant time and then adding up the look-up results.

Note that when multiple heat sources are present, their effects on the average temperature rise above T_a in the field region, i.e., the integral term in equation (2.25), can be summed up to obtain the total average temperature rise.

The selection of the discretization parameters M and N deserves some more considerations. Assume that the minimum feature size along the x and y directions that must be resolved are x_{min} and y_{min}, respectively, then M and N must satisfy

$$M{\geq}M_r = \frac{a}{x_{min}} \quad \text{and} \quad N{\geq}N_r = \frac{b}{y_{min}} \tag{2.39}$$

where M_r and N_r represent the minimum values of M and N from resolution considerations. However, since M and N are also the truncation points of the summations in equation (2.27), they must be large enough to ensure the convergence of the summations. As pointed out in [Gha], the summations converge more slowly as x_{min} and y_{min} become smaller relative to the chip dimensions a and b. Thus, the actual values of M and N cannot be determined merely based on M_r and N_r. Let M_c and N_c be the minimum values of $M \geq M_r$ and $N \geq N_r$ such that the convergence is achieved in (2.27). In our implementation, M_c and N_c are determined as follows. We consider nine representative regions on each of the source and field planes as shown in Fig. 2.4. Each region has dimensions of $x_{min} \times y_{min}$. We first set $M_c = M_r$ and $N_c = N_r$. Then we increase M_c and N_c gradually until the convergence of the summations in (2.27) is achieved for all of the possible locations of the source and field regions provided the source region coincides with one of the nine representative regions on the source plane while the field region coincides with one of the nine representative regions on the field plane. Finally, to assist the utilization of the FFT in the DCT computations, M and N are chosen to be integers that are powers of 2 and are no smaller than M_c and N_c, respectively.

Compared with the classical Green function method, the advantage of our algorithm is that it replaces the expensive double summations in the expressions involving the Green function by the inexpensive summations of a few numbers in the pre-

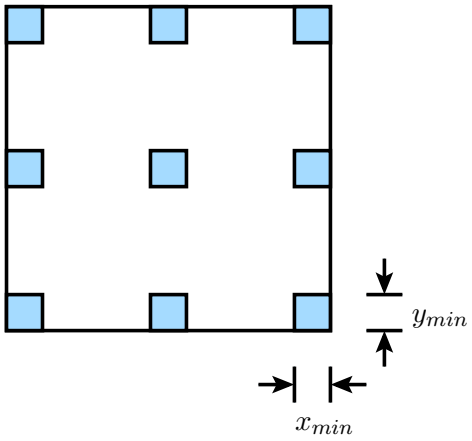


Figure 2.4: The locations of the nine representative regions on the source plane. Each region has dimensions of $x_{min} \times y_{min}$. One region is located at the center of the plane, one is at the mid-point of each edge, and one is at each corner. Similarly, we have nine representative regions on the field plane.

calculated look-up table and vectors. The look-up table and vectors only depend on the chip dimensions and the physical properties of the substrate, but are independent of the layout and power distribution. Hence, the look-up table and vectors can be calculated once and then used many times in thermal-aware physical designs, which significantly reduces the amortized cost of obtaining the table and vectors, and improves the overall efficiency of the algorithm.

2.3.2 Algorithm II: Full-chip thermal simulation using the spectral domain computations

Algorithm I gained its efficiency from the faster evaluations of the expressions involving the Green function. Asymptotically, however, it is still an expensive method

for simulations involving a large number of heat sources and field regions because the effects of the heat sources on the field regions are calculated in a pair-wise fashion. The second algorithm we present in this section targets full-chip thermal simulations with large problem sizes. It uses spectral domain analysis to reduce the asymptotic time complexity of calculating the on-chip temperature distribution. In the following analysis, we focus on the effect of one source plane on the temperature distribution in the field plane. When multiple source planes are present, their effects can be easily summed up to obtain the final solution.

Since the convolution integral in (2.25) can be considered as the governing equation of a linear system determined by the Green function $G(x, y, x', y')$, we can use spectral domain analysis to accelerate the computations corresponding to the convolution integral.

The first step of our algorithm is to obtain the spectral domain representation of the power density map in the form

$$P_d(x', y') = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{ij} \phi_{ij}(x', y') \tag{2.40}$$

where

$$\phi_{ij}(x, y) = \cos\left(\frac{i\pi x}{a}\right) \cos\left(\frac{j\pi y}{b}\right) \tag{2.41}$$

It is easy to show that $\phi_{ij}(x,y)$ satisfies the equation

$$\lambda_{ij}\phi_{ij}(x,y) = \int_0^a dx' \int_0^b dy' G(x,y,x',y')\phi_{ij}(x',y') \tag{2.42}$$

where

$$\lambda_{ij} = \begin{cases} abC_{ij} & \text{if } i=j=0 \\ \frac{1}{2}abC_{ij} & \text{if } i=0, j\neq 0 \text{ or } i\neq 0, j=0 \\ \frac{1}{4}abC_{ij} & \text{if } i\neq 0, j\neq 0 \end{cases} \tag{2.43}$$

is the response of the linear system to the spectral component $\phi_{ij}(x,y)$ [CCS99]. After the spectral domain representation of the power density distribution in the source plane is obtained, the temperature distribution in the field plane can be calculated easily by

$$T(x,y) = T_a + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \lambda_{ij} a_{ij} \phi_{ij}(x,y) \tag{2.44}$$

As will be shown next, both the spectral decomposition in (2.40) and the double-summation in (2.44) can be calculated efficiently using the DCT and IDCT through the FFT.

Now we assume that the source plane is divided into $M_s \times N_s$ rectangular grid cells of equal size as shown in Fig. 2.5, and the power density in each grid cell on the source plane is uniform, i.e., the power density distribution can be written in the piece-wise

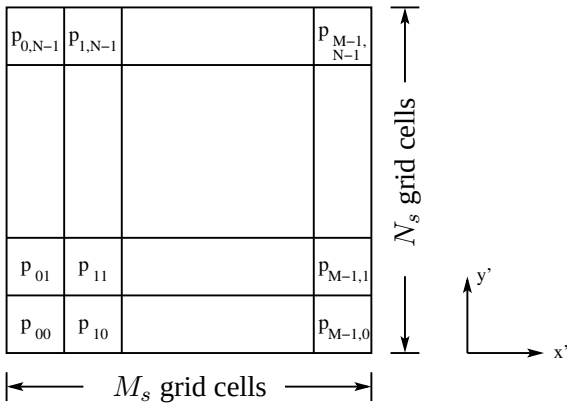


Figure 2.5: The arrangement of the $M_s \times N_s$ grid cells on the source plane.

constant form

$$P_d(x', y') = \sum_{m=0}^{M_s-1} \sum_{n=0}^{N_s-1} P_{mn} \Theta(x' - (m + \frac{1}{2})\Delta x_s, y' - (n + \frac{1}{2})\Delta y_s) \tag{2.45}$$

where

$$\Theta(x', y') = \begin{cases} 1 & \text{if } |x'| \leq \frac{1}{2}\Delta x_s \text{ and } |y'| \leq \frac{1}{2}\Delta y_s \\ 0 & \text{otherwise} \end{cases} \tag{2.46}$$

and $\Delta x_s = \frac{a}{M_s}$, $\Delta y_s = \frac{b}{N_s}$. P_{mn} is the power density of the mn^{th} grid cell.

Note that if the piece-wise constant power density map is not directly given in the form of (2.45), it can be conveniently derived from the layout geometries and the power generated by each circuit component. Assume that the layout of each component is within a rectangular-shaped region as shown in Fig. 2.6, and the region corresponding to the i^{th} component C_i is defined by $x_i^L \leq x \leq x_i^R$ and $y_i^B \leq y \leq y_i^T$. The range of

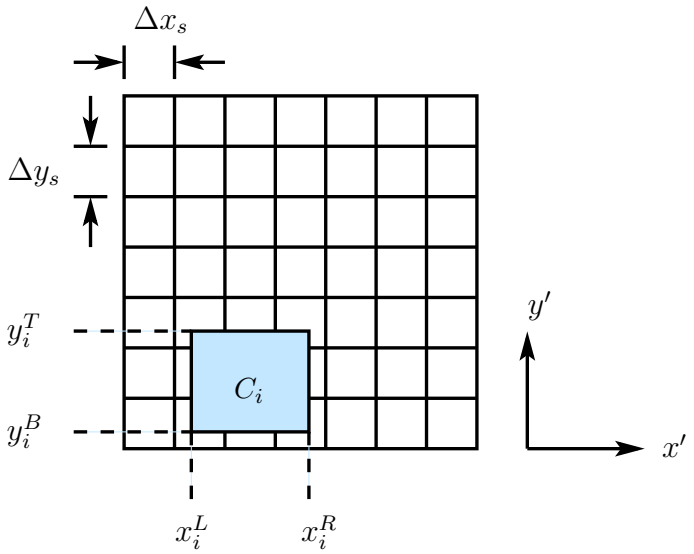


Figure 2.6: Calculating the power density map from the given layout geometries and the power generated by each circuit component.

the indices m and n of the grid cells that the i^{th} component overlaps is given by

$$\begin{aligned} \lfloor \frac{x_i^L}{\Delta x_s} \rfloor \leq m \leq \lfloor \frac{x_i^R}{\Delta x_s} \rfloor \\ \lfloor \frac{y_i^B}{\Delta y_s} \rfloor \leq n \leq \lfloor \frac{y_i^T}{\Delta y_s} \rfloor \end{aligned} \tag{2.47}$$

Assume that the total power generated by the i^{th} component is given by P_i^T, then its contribution to the power density of the mn^{th} grid cell that overlaps with it is

$$\delta P_{mn}^i = P_i^T \times \frac{S_{mn}^i}{(x_i^R - x_i^L)(y_i^T - y_i^B)} \times \frac{1}{\Delta x_s \cdot \Delta y_s} \tag{2.48}$$

where S_{mn}^i is the overlap area of the rectangle corresponding to the i^{th} component and

the mn^{th} rectangular-shaped grid cell, and it can be calculated in constant time. Therefore, obtaining the piece-wise constant power density map from the layout geometries and the power generated by each circuit component has only a linear time complexity with respect to the number of components in the circuit, and it can be usually ignored compared with the costs of other calculations involved in the thermal simulation.

Substituting (2.45) into (2.40) and using the orthogonality property of the cosine functions in the integral sense, we obtain

$$a_{ij} = A_{ij} \sum_{m=0}^{M_s-1} \sum_{n=0}^{N_s-1} P_{mn} \cos\left(\frac{i\pi(2m+1)}{2M_s}\right) \cos\left(\frac{j\pi(2n+1)}{2N_s}\right) \tag{2.49}$$

where

$$A_{ij} = \begin{cases} \frac{1}{M_s N_s} & \text{if } i = j = 0 \\ \frac{4}{iN_s\pi} \sin\left(\frac{i\pi}{2M_s}\right) & \text{if } i \neq 0, j = 0 \\ \frac{4}{M_s j\pi} \sin\left(\frac{j\pi}{2N_s}\right) & \text{if } i = 0, j \neq 0 \\ \frac{16}{ij\pi^2} \sin\left(\frac{i\pi}{2M_s}\right) \sin\left(\frac{j\pi}{2N_s}\right) & \text{if } i \neq 0, j \neq 0 \end{cases} \tag{2.50}$$

Note that to accurately represent the power density distribution $P_d(x', y')$ using (2.40), the theoretical upper limit of the double summation should be infinity. In practical implementations, however, the summation must be truncated to ensure a reasonable runtime. Since (2.40) is essentially the Fourier expansion of $P_d(x', y')$, a natural criterion for determining the truncation point is that enough "energy" contained in

$P_d(x', y')$ is covered by the truncated Fourier expansion. Mathematically, we have

$$\int_0^a dx' \int_0^b dy' P_d^2(x', y') = ab \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} s_{ij} a_{ij}^2 \tag{2.51}$$

where

$$s_{ij} = \begin{cases} 1 & \text{if } i = j = 0 \\ \frac{1}{2} & \text{if } i = 0, j \neq 0 \text{ or } i \neq 0, j = 0 \\ \frac{1}{4} & \text{if } i \neq 0, j \neq 0 \end{cases} \tag{2.52}$$

Substituting (2.45) into the left hand side of (2.51), we obtain

$$\frac{1}{M_s N_s} \sum_{m=0}^{M_s-1} \sum_{n=0}^{N_s-1} P_{mn}^2 = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} s_{ij} a_{ij}^2 \tag{2.53}$$

which can be considered as a form of the Parseval's theorem. The truncation points M' and N' are then determined by

$$\sum_{i=0}^{M'-1} \sum_{j=0}^{N'-1} s_{ij} a_{ij}^2 \geq \eta \left(\frac{1}{M_s N_s} \sum_{m=0}^{M_s-1} \sum_{n=0}^{N_s-1} P_{mn}^2 \right) \tag{2.54}$$

where η is the proportion of the "energy" of the space domain signal $P_d(x', y')$ that must be covered by the truncated Fourier expansion. In practice, we found that setting η to 90% will usually be enough to obtain very accurate results in temperature calculations.

We emphasize here that (2.54) does not imply that only a fraction, η, of the total power generated by the heat sources is included in the truncated expansion. In reality, the total power is completely contained in the DC term of expansion (2.40), and (2.54) only describes how accurately we are approximating the *exact* shape of the space domain signal, i.e., $P_d(x', y')$. A smaller η implies that more components with high spectral numbers in $P_d(x', y')$ are ignored, or equivalently, more *zero mean* noises with high spectral numbers are added to the approximating power distribution. Since the temperature distribution is calculated using (2.25) and the convolution with the Green function has a low-pass filtering effect, η does not have to be extremely close to 1 in order to calculate the temperature accurately. We also point out that although η is set to a constant number, the truncation points M' and N' are not determined *a priori* in our algorithm. Instead, they depend on $P_d(x', y')$ according to (2.54). Our strategy of determining the truncation points is to first set $M' = M_s$ and $N' = N_s$. If (2.54) is not satisfied, then we increase M' to $2M_s$ and N' to $2N_s$. The summation limits M' and N' continue to increase with steps of M_s and N_s until (2.54) is satisfied. The importance of determining the truncation points dynamically based on the input data will become more obvious as the size of the problem increases.

Note that for $0 \leq i < M_s$ and $0 \leq j < N_s$, the double summation in (2.49) can be considered as a term in the 2D type-II DCT [OSB99] of the power density matrix P. For $i \geq M_s$ or $j \geq N_s$, we can always find integers s_1 and s_2 such that $i = 2s_1M_s \pm \hat{i}$ and

$j = 2s_2N_s \pm \hat{j}$ where $0 \leq \hat{i} < M_s$ and $0 \leq \hat{j} < N_s$[1]. Hence, for any i and j, we always have

$$a_{ij} = \pm A_{ij} \widetilde{P}_{\hat{i}\hat{j}} \tag{2.55}$$

where

$$\widetilde{P}_{\hat{i}\hat{j}} = \sum_{m=0}^{M_s-1} \sum_{n=0}^{N_s-1} P_{mn} \cos\left(\frac{\hat{i}\pi(2m+1)}{2M_s}\right) \cos\left(\frac{\hat{j}\pi(2n+1)}{2N_s}\right) \tag{2.56}$$

with $0 \leq \hat{i} < M_s$ and $0 \leq \hat{j} < N_s$ is the 2D type-II DCT of the P matrix and the sign of (2.55) is determined by whether s_1 and s_2 are even or odd numbers [CCS99]. Equation (2.56) can be calculated efficiently using the 2D FFT in $O((M_s \cdot N_s) \times \log(M_s \cdot N_s))$ time. After the 2D DCT matrix $\widetilde{P}$ is obtained, the calculation of a_{ij} simply involves computing the coefficient A_{ij} and finding the corresponding term $\widetilde{P}_{\hat{i}\hat{j}}$.

From (2.41), (2.44), and (2.54), the temperature distribution $T(x, y)$ can now be written as

$$T(x, y) = T_a + \sum_{i=0}^{M'-1} \sum_{j=0}^{N'-1} \lambda_{ij} a_{ij} \cos\left(\frac{i\pi x}{a}\right) \cos\left(\frac{j\pi y}{b}\right) \tag{2.57}$$

If we assume that the temperature field plane is divided into $M_f \times N_f$ rectangular grid cells of equal size, then the average temperature of the mn^{th} grid cell can be obtained

[1] If i equals an odd multiple of M_s, we will not be able to write i as $i = 2s_1M_s \pm \hat{i}$. However, for this kind of i, it can be easily shown that $a_{ij} = 0$ because $\cos\left(\frac{i\pi(2m+1)}{2M_s}\right) = 0$. Similarly, we know that $a_{ij} = 0$ if j equals an odd multiple of N_s.

by

$$T_{mn} = \frac{1}{\Delta x_f \Delta y_f} \int_{m\Delta x_f}^{(m+1)\Delta x_f} dx \int_{n\Delta y_f}^{(n+1)\Delta y_f} dy T(x,y)$$
$$= T_a + \sum_{i=0}^{M'-1} \sum_{j=0}^{N'-1} B_{ij} \cos\left(\frac{i\pi(2m+1)}{2M_f}\right) \cos\left(\frac{j\pi(2n+1)}{2N_f}\right) \tag{2.58}$$

where $\Delta x_f = \frac{a}{M_f}$, $\Delta y_f = \frac{b}{N_f}$, and

$$B_{ij} = \begin{cases} \lambda_{ij} a_{ij} & \text{if } i = j = 0 \\ 2\lambda_{ij} a_{ij} \frac{M_f}{i\pi} \sin\left(\frac{i\pi}{2M_f}\right) & \text{if } i \neq 0, j = 0 \\ 2\lambda_{ij} a_{ij} \frac{N_f}{j\pi} \sin\left(\frac{j\pi}{2N_f}\right) & \text{if } i = 0, j \neq 0 \\ 4\lambda_{ij} a_{ij} \frac{M_f N_f}{ij\pi^2} \sin\left(\frac{i\pi}{2M_f}\right) \sin\left(\frac{j\pi}{2N_f}\right) & \text{if } i \neq 0, j \neq 0 \end{cases} \tag{2.59}$$

Similar to the analysis shown previously, any $i \geq M_f$ and $j \geq N_f$ can be written as $i = 2s_3 M_f \pm \hat{\hat{i}}$ and $j = 2s_4 N_f \pm \hat{\hat{j}}$ such that $0 \leq \hat{\hat{i}} < M_f$, $0 \leq \hat{\hat{j}} < N_f$, and s_3 and s_4 are integers. Using the periodicity of the cosine function, we can finally cast T_{mn} into the form

$$T_{mn} = T_a + \sum_{\hat{\hat{i}}=0}^{M_f-1} \sum_{\hat{\hat{j}}=0}^{N_f-1} L_{\hat{\hat{i}}\hat{\hat{j}}} \cos\left(\frac{\hat{\hat{i}}\pi(2m+1)}{2M_f}\right) \cos\left(\frac{\hat{\hat{j}}\pi(2n+1)}{2N_f}\right) \tag{2.60}$$

where

$$L_{\hat{\hat{i}}\hat{\hat{j}}} = \begin{cases} B_{00} & \text{if } \hat{\hat{i}} = \hat{\hat{j}} = 0 \\ \sum_{\substack{i < M' \\ i = 2s_3 M_f \pm \hat{\hat{i}}}} \pm B_{i0} & \text{if } \hat{\hat{i}} \neq 0, \hat{\hat{j}} = 0 \\ \sum_{\substack{j < N' \\ j = 2s_4 N_f \pm \hat{\hat{j}}}} \pm B_{0j} & \text{if } \hat{\hat{i}} = 0, \hat{\hat{j}} \neq 0 \\ \sum_{\substack{i < M' \\ i = 2s_3 M_f \pm \hat{\hat{i}}}} \sum_{\substack{j < N' \\ j = 2s_4 N_f \pm \hat{\hat{j}}}} \pm B_{ij} & \text{if } \hat{\hat{i}} \neq 0, \hat{\hat{j}} \neq 0 \end{cases} \tag{2.61}$$

and the signs of the $B's$ in (2.61) are determined by whether s_3 and s_4 are even or odd numbers. After the matrix L is obtained, the double summation in (2.60) can be calculated efficiently using the 2D IDCT.

The complete thermal simulation algorithm using the Green function method, the DCT, and the spectral domain computations is shown in Fig. 2.7. The asymptotic time complexity of the algorithm is $O(\mathcal{N}_{gs} \times \log(\mathcal{N}_{gs})) + O(\mathcal{N}_{gf} \times \log(\mathcal{N}_{gf}))$ where $\mathcal{N}_{gs} = M_s \cdot N_s$ is the total number of grid cells in the power density map, and $\mathcal{N}_{gf} = M_f \cdot N_f$ is the total number of grid cells in the resulting temperature profile. This is a significant improvement over the $O(\mathcal{N}_{gs} \cdot \mathcal{N}_{gf})$ complexity of Algorithm I for full-chip thermal simulations.

Input:

- Chip geometry and physical properties of the material layers.
- Power density map - matrix P.

Output: Temperature distribution map - matrix T.
Algorithm:

1. Calculate the Green function coefficients $C_{ij}{}'s$;
2. Calculate the spectral responses of the system $\lambda_{ij}{}'s$;
3. Calculate the type-II 2D DCT of the power density matrix $\widetilde{P} = \mathrm{2DDCT}(P)$;
4. $TSE = \frac{1}{M_s N_s} \sum_{m=0}^{M_s-1} \sum_{n=0}^{N_s-1} P_{mn}^2$;
5. $M' = M_s,\ N' = N_s$;
   $ASE = \sum_{i=0}^{M'-1} \sum_{j=0}^{N'-1} s_{ij} a_{ij}^2$;
   while ($ASE < \eta \times TSE$)
   $M' = M' + M_s$, $N' = N' + N_s$;
   Update ASE;
   end while;
6. Calculate the matrix L;
7. Calculate the temperature distribution map using the type-II 2D IDCT $T = T_a + \mathrm{2DIDCT}(L)$;

Figure 2.7: Thermal simulation algorithm using the Green function method, the DCT, and the spectral domain computations.

2.3.3 Algorithm III: Thermal simulation with local high accuracy requirements

Although Algorithm II can achieve a $O(\mathcal{N}_{gs} \times \log(\mathcal{N}_{gs})) + O(\mathcal{N}_{gf} \times \log(\mathcal{N}_{gf}))$ time complexity as opposed to a $O(\mathcal{N}_{gs} \cdot \mathcal{N}_{gf})$ complexity of Algorithm I for full-chip ther-

mal simulations, Algorithm I is still more efficient for performing the localized analysis, where the effects of a few critical circuit blocks on the temperature distribution in a few key field regions are of interest. This is because to apply Algorithm II, we must always superimpose regular grids over the entire source and field planes and calculate the complete temperature profile from the complete power density distribution. The size of each grid cell must be comparable with that of the resolution requirement of the calculation, and the total number of grid cells determines the problem size. Therefore, although Algorithm II has a smaller asymptotic time complexity than Algorithm I for full-chip thermal simulations, it may also require the formulation of a problem with much larger size than Algorithm I if only some localized temperature calculations are required by circuit designers.

We will face an even more difficult decision concerning whether Algorithm I or Algorithm II should be used when a circuit designer has different requirements on the accuracy of the thermal simulation over different parts of the same chip. For example, in mixed signal designs where analog circuits are fabricated on the same chip as digital circuits, the analog blocks often have more stringent accuracy requirements on the thermal simulation because the operations of the analog circuits are more sensitive to temperature. If the full-chip temperature profile is required, then Algorithm I will be too slow to use. However, in order to use Algorithm II, the size of each grid cell must be small enough so that the high accuracy requirements of the analog blocks are

satisfied. This may result in very dense grids and a large problem size. For these kinds of problems, a better strategy can be adopted to accelerate the runtime of the algorithm further by combining the advantages of both Algorithm I and II. The key idea is to use coarse grids to divide the source and field planes where each grid cell in the source plane can contain several logic gates or analog functional units, and the size of each grid cell in the field plane satisfies the accuracy requirements of the digital circuits. The power density of each grid cell in the source plane is calculated by summing up the power dissipations of all the logic gates and analog functional units located in it and dividing the sum by the area of the grid cell. A coarse temperature map for the field plane is then obtained from the coarse power density map using Algorithm II and is used for the digital blocks. Finally, for each analog functional unit on the field plane whose temperature is to be calculated more accurately, we use Algorithm I to compute the contributions to its temperature rise from the nearby logic gates and analog function units on the source plane, and use this result to correct the temperature obtained by Algorithm II over the coarse grid cell. To simplify the presentation, we assume in the following analysis that the source plane coincides with the field plane and both of them are divided into $M \times N$ coarse grid cells. However, this assumption is not essential to the algorithm and it can be relaxed easily to handle multiple source and field planes such as that in the emerging three-dimensional IC technologies.

Fig. 2.8 shows a chip that is divided into $M \times N$ coarse grid cells each of which

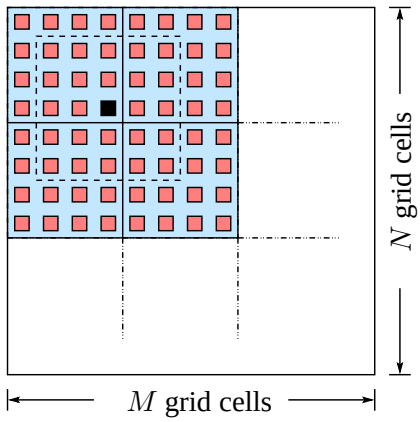


Figure 2.8: A mixed signal chip where the analog block has higher requirement on the accuracy of thermal simulations. The logic gates and analog functional units within the dashed line constitute the set $C(A)$.

contains several logic gates or analog functional units, and let the shaded area represent the analog block. An $M \times N$ temperature map is first obtained. The inaccuracies in the temperature calculations, besides that due to the truncation of the spectral domain representation of the power density map, will come from two sources which include

- Assuming that the power density in each grid cell is uniform.
- Only the average temperature of each grid cell is calculated, i.e., all of the logic gates and analog functional units inside the same grid cell obtain the same calculated temperature.

Now assume that we need to calculate the temperature of the analog functional unit A located in the ij^{th} grid cell and represented by the black rectangle more accurately. Let T_{ij} be the average temperature of the ij^{th} grid cell obtained using Algorithm II,

and let $T_{ij,S}$ be the contribution to the average temperature rise of the ij^{th} grid cell from the logic gate or analog functional unit S assuming that the power generated by S is uniformly distributed in the grid cell in which it resides. Denote the more accurate average temperature of the analog functional unit A by $T_A^{accurate}$, and let $T_{A,S}^{accurate}$ be the accurate contribution to the temperature rise of A from the logic gate or analog functional unit S. The temperature $T_A^{accurate}$ can be obtained by

$$T_A^{accurate} = T_{ij} - \sum_{S \in C(A)} T_{ij,S} + \sum_{S \in C(A)} T_{A,S}^{accurate} \tag{2.62}$$

where $C(A)$, which will be called the interaction set of A in the following analysis, is the set of logic gates and analog functional units that are physically close to A, and hence, whose contributions to the temperature rise of A must be re-calculated accurately. The size of $C(A)$ is determined by the actual accuracy requirement on the temperature of A, and a higher accuracy requirement is usually associated with a larger $C(A)$. Both $T_{ij,S}$ and $T_{A,S}^{accurate}$ can be calculated efficiently using Algorithm I, and the overall efficiency of the combined algorithm is higher than that of Algorithm II applied with a fine grid over the entire chip that satisfies the high accuracy requirements of the analog functional units.

2.3.4 Time complexity analysis

We summarize the time complexities of the three algorithms in this section. Note that the calculations involved in each of the algorithms can be divided into two parts, i.e., those only depend on the chip geometry and the physical properties of the chip materials, and those depend on the input power density distribution. The computation steps that only involve the chip geometry and material properties can be performed in the pre-characterization phase of the design, and their results can be stored for further uses. Therefore, the amortized costs of these steps are usually rather low in the overall physical design process, where the optimization routine executes the thermal simulation many times. The steps that involve the input power density distribution, however, must be executed within the optimization routine in physical designs. Hence, they usually dominate the overall runtime of the thermal-aware physical design algorithms such as the thermal-aware floorplanning and placement. The establishment of the look-up table and vectors in Algorithm I and the calculation of the spectral responses of the linear system in Algorithm II can both be performed in the pre-characterization phase, and in the following analysis, we will ignore the costs of these steps and only focus on the time complexity of the calculations that depend on the input power density distribution.

For the input-power-dependent steps in thermal simulations, Algorithm I has a time complexity of $O(\mathcal{N}_s \times \mathcal{N}_f)$, where $\mathcal{N}_s$ and $\mathcal{N}_f$ are the number of heat sources and field

regions, respectively. Algorithm II always works with full-chip power density distribution and generates the complete on-chip temperature profile. It has a time complexity of $O(\mathcal{N}_{gs}\times\log(\mathcal{N}_{gs})) + O(\mathcal{N}_{gf}\times\log(\mathcal{N}_{gf}))$ where $\mathcal{N}_{gs} = M_s \cdot N_s$ is the total number of grid cells in the input power density map, and $\mathcal{N}_{gf} = M_f \cdot N_f$ is the total number of grid cells in the obtained temperature profile. Here, M_s and N_s are the number of grid divisions along the x and y directions on the source plane, and M_f and N_f are the number of grid divisions along the x and y directions on the field plane. It is obvious that Algorithm II is better than Algorithm I for full-chip temperature profiling, because the latter has a time complexity of $O(\mathcal{N}_{gs} \cdot \mathcal{N}_{gf})$. For the localized analysis where only a few source and field regions are involved, however, Algorithm I can often perform better because $\mathcal{N}_{gs}$ and $\mathcal{N}_{gf}$ are determined by the highest resolution requirement of the analysis, and $\mathcal{N}_s$ and $\mathcal{N}_f$ are usually much smaller than $\mathcal{N}_{gs}$ and $\mathcal{N}_{gf}$ for this type of problems.

To compare Algorithm II and Algorithm III, we assume that there are $\mathcal{N}_{total}$ logic gates and analog functional units in the design. Using Algorithm II directly with a grid size comparable to the smallest size of the gates and functional units will result in a time complexity of $O(\mathcal{N}_{total}\times\log(\mathcal{N}_{total}))$. For Algorithm III, a coarse grid is first used in the calculation. If we assume that each coarse grid cell contains K gates and functional units, then the time it takes to obtain the coarse temperature profile is $O(\frac{\mathcal{N}_{total}}{K}\times\log(\frac{\mathcal{N}_{total}}{K}))$. Now, if the accurate temperature correction is to be performed

over all of the gates and functional units, then an additional cost of $O(\mathcal{N}_{total} \cdot K')$ is required where K' is the size of the interaction set of each gate or functional unit, and the total cost becomes $O(\mathcal{N}_{total} \times (\frac{1}{K}\log(\frac{\mathcal{N}_{total}}{K}) + K'))$. Note that the $O(\mathcal{N}_{total} \cdot K')$ term in the complexity analysis involves a relatively large pre-factor due to the 80 look-ups needed to calculate the correction corresponding to a pair of gates or functional units. Hence, the actual runtime of Algorithm III is often longer than that of Algorithm II when the accurate temperature correction is to be performed over all of the gates and functional units. However, as pointed out previously, it frequently happens in real design environments that the temperature correction is only required for a small portion of the circuit. Therefore, the total cost of Algorithm III becomes $O(\frac{\mathcal{N}_{total}}{K} \times \log(\frac{\mathcal{N}_{total}}{K}) + \mathcal{N}_c \cdot K')$, where $\mathcal{N}_c$ is the number of gates and functional units that require temperature corrections, and Algorithm III becomes more efficient than Algorithm II under this situation.

2.4 Experimental Results

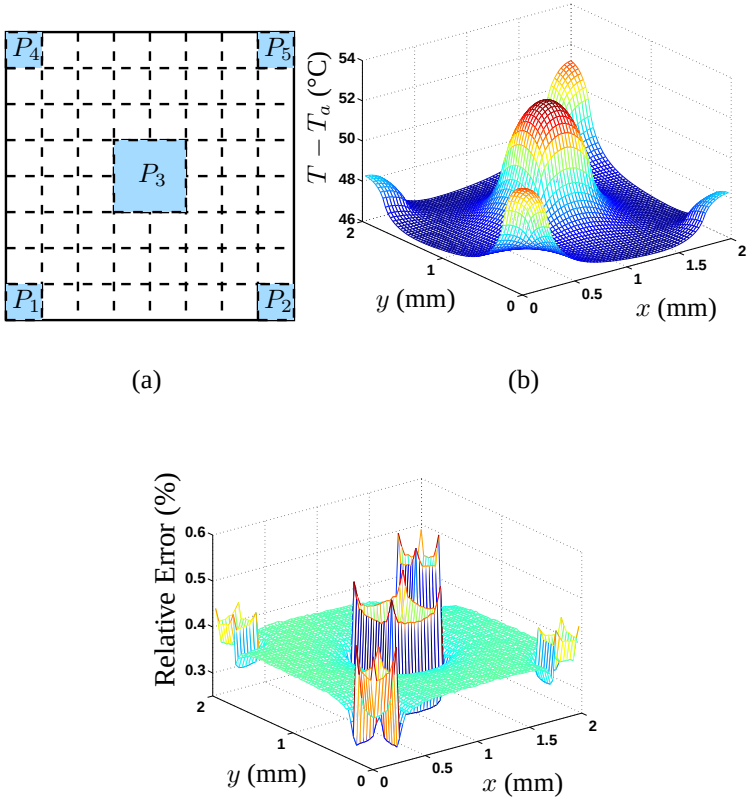
In this section, we present in detail the performance of the three algorithms, which are implemented in C++ and compiled using the level 3 optimization of g++. The experiments are performed on a desktop with a 3.2GHz Intel Pentium-4 CPU running the Red Hat Linux 8.0 operating system. We first compare the results obtained from

Algorithm I with that from a commercial computational fluid dynamic (CFD) software package and that from the direct application of the Green function method in terms of accuracy and efficiency. Then we use Algorithm I as our base method to characterize the performance of the other two algorithms.

The commercial CFD software package uses a finite volume approach which meshes the entire substrate. Because of the discretized nature of the method, meshing errors are unavoidable. In order to control the meshing errors while still complete the computation within a reasonable amount of time, we start with a relatively rough mesh and continue refining it and re-running the simulation until the maximum error converges to around 1%. By doing this, we ensure that the result produced by the CFD software itself is accurate, and therefore it can be used as a valid criterion to evaluate the accuracy of our algorithms.

2.4.1 Accuracy and efficiency of Algorithm I

Fig. 2.9(a) shows the top surface of a silicon chip with dimensions of 2mm$\times 2$mm $\times 0.5$mm. The area is divided into 8×8 equal square sections, and five power sources are placed in the corresponding sections as shown in the figure. The thermal conductivity k of silicon is 148W/(m·°C), and the effective heat transfer coefficient h of the bottom surface of the chip is chosen to be 8700W/(m^2·°C), which is consistent with the value used in [CRT+98]. The strength of the five power sources are $(P_1, P_2, P_3, P_4, P_5)$



(a) (b)

(c)

Figure 2.9: Accuracy of Algorithm I (a) power source locations (b) computed temperature distribution above T_a using the proposed algorithm (c) relative error of the proposed algorithm compared with the result from a commercial CFD software package.

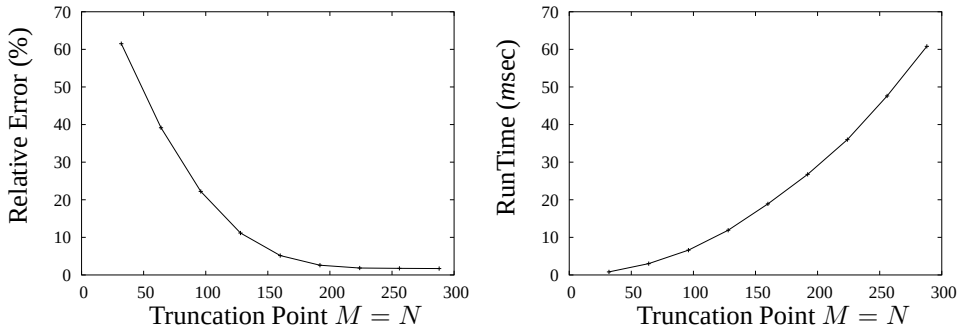
= (0.2W, 0.1W, 1W, 0.1W, 0.2W).

Fig. 2.9(b) shows the top surface temperature map obtained using Algorithm I, where $T - T_a$ is the temperature rise above the ambient. In obtaining the temperature map, the top surface of the chip was divided into 64×64 small square regions with

equal size and the average temperature in each small square region was computed. The parameters M and N were both set to 64, the minimum required values from resolution considerations, because the convergence of the Green function has already been achieved with $M = N = 64$. Fig. 2.9(c) shows the relative error in the temperature map compared with the computation result obtained from a commercial CFD software package for thermal analysis. We can see clearly that the error is below 1%, which demonstrates the accuracy of our method.

We next compare the efficiency of Algorithm I with that of the direct application of the Green function method to compute the temperature distribution. We still use the same chip dimensions and physical properties as in the previous example. However, only one power source is used this time to make the presentation clearer. The power source occupies a square region with dimensions of $\frac{2}{128}\mathrm{mm} \times \frac{2}{128}\mathrm{mm}$ at the exact center of the chip. The strength of the power source is $P_s = 50\mathrm{mW}$. The average temperature above T_a of the source region itself is computed. The parameters M and N are both chosen to be 512 in our algorithm from convergence considerations for the Green function, i.e., we require the truncation error to be within 1%. The infinite summations in the Green function are more difficult to converge in this example because the sizes of the source and field regions relative to the chip dimensions are smaller than those in the previous example.

Using Algorithm I, the average temperature of the source region itself above T_a



(a) (b)

Figure 2.10: Accuracy and computation time of the direct application of the Green function method (a) relative error in $T - T_a$ versus truncation point (b) runtime versus truncation point.

is found to be 11.537°C. The total computation time using the pre-calculated look-up table and vectors is only 5.5×10^{-4}*m*sec. As a comparison, we also computed the average temperature above T_a of the source region using equation (2.27) directly, which corresponds to the direct application of the Green function method. In the direct method, it is unnecessary to consider the resolution issue because equation (2.27) does not require the vertices (a_i, b_i) of the source and field regions to coincide with some grid points. So the parameters M and N are completely determined by the convergence consideration. Since the chip is square, we set $M = N$ in our analysis.

Fig. 2.10 shows the relative error and the corresponding runtime of the direct method. We can observe from the figure that even for a 5% relative error in $T - T_a$, the truncation point must be higher than 160. The runtime at this truncation point is

19*m*sec, which is four orders of magnitude slower than our algorithm, and the accuracy of our algorithm is much higher.

2.4.2 Comparison between Algorithm I and Algorithm II

Now, we compare the efficiency and accuracy of Algorithm I and II using a real chip example. Fig 2.11(a) shows a floorplan from [LHL], which is similar to that of the DEC Alpha 21264 processor but is scaled from the 350nm to the 65nm technology. The scaled chip dimensions are $3.3\text{mm}\times3.3\text{mm}\times0.506\text{mm}$, and we assume that the chip has the same physical properties as those used in the previous examples except that a layer representing the interconnects is inserted between the insulating top surface and the substrate as shown in Fig 2.11(b). The added layer is assumed to have a thickness of 6μm and an effective thermal conductivity of 101W/(m·°C), which corresponds to a mixing of 25% copper, which has a thermal conductivity of 401W/(m·°C), and 75% oxide, which has a thermal conductivity of 1W/(m·°C). In real designs, the effective thermal conductivity of the interconnect layer can be estimated by taking the weighted average of the thermal conductivities of interconnect metal and oxide based on the designers' experiences on the interconnect densities of previous designs[2]. We further assume that the power is generated by the modules located at the interface between the

[2]For early stages of physical design where the detailed information about routing is usually unavailable, it is reasonable to use a uniform effective thermal conductivity to characterize the thermal property of each interconnect layer.

interconnect layer and the substrate, and the temperature profile of this interface where the modules are located is calculated. Fig. 2.11(c) shows the power density distribution of the modules in W/cm^2. We divided the module layer into 64×64 small square regions with equal size and computed the temperature maps using Algorithm I and II, which are shown in Fig. 2.11(d) and (e). Fig. 2.11(f) shows the difference between the temperature maps obtained using the two algorithms, and we can see that the results match each other very well. From the figures, we can also observe that the temperature maps are much smoother than the power density map. This can be explained by the relatively high thermal conductivity of the silicon substrate and the horizontal heat transfer [SSHV03]. For the CPU times required to obtain the temperature maps, Algorithm I uses 30*m*sec after the look-up table and vectors have been pre-calculated, while Algorithm II only uses 10*m*sec after the spectral responses of the linear system determined by the underlying Green function have been pre-calculated. Note that the runtime of Algorithm I is linear with respect to the number of heat sources and there are only 14 heat sources in the example shown here. For cell level full-chip simulations where the number of heat sources is significantly larger, the advantages of Algorithm II will become even more obvious. Therefore, we conclude that Algorithm II is more suitable for full-chip temperature profiling, where a large number of heat sources and field regions are involved.

To further demonstrate the efficiency of Algorithm II in full-chip thermal simu-

lations, we tested a chip with dimensions of 1cm$\times$1cm$\times$0.5mm and has the same physical properties as the chips used in Section 2.4.1. There are 1024$\times$1024 square grid cells of equal size located on the top surface of the chip and a 1024$\times$1024 temperature distribution map of the cell layer is calculated. Fig. 2.12 shows the input power density map and the resulting temperature map. The time it takes to obtain this temperature map containing 1.05M grid cells is only 3.7sec, excluding the time for the pre-calculations, while the runtime of Algorithm I becomes intractable.

2.4.3 Effectiveness of Algorithm III

Finally, we show an example of thermal simulation with local high accuracy requirement. We consider a chip that contains 8$\times$8 coarse grid cells each of which has dimensions of 3.3mm$\times$3.3mm, as shown in Fig. 2.13(a). The chip has the same material properties as the ones used in Section 2.4.1. We embed the layout and power density distribution shown in Fig. 2.11(a) and (c) in the coarse grid cell located at the lower left corner of the chip, which we denote by $CGC(0,0)$ in the following analysis, and the power density of each of the other 63 coarse grid cells is randomly generated between 0 and 100W/cm^2. Suppose that we want to obtain a 8$\times$8 coarse temperature map over the 64 coarse grid cells and a 64$\times$64 fine temperature map within $CGC(0,0)$. We compare two simulation schemes. In the first scheme, Algorithm II alone is used. In order to achieve the accuracy requirement of the fine temperature

map within $CGC(0,0)$, we have to divide each of the 64 coarse grid cells into 64×64 fine cells, which results in a total of 512×512 fine cells over the entire chip. The time it takes to complete this simulation is 850*m*sec. In the second simulation scheme, we first obtain a 8×8 coarse temperature map from the 8×8 coarse power density map assuming that the power density within each coarse grid cell is uniform. The average temperature of $CGC(0,0)$ is found to be 79.4°C, while we know from the first simulation scheme that the actual temperature within $CGC(0,0)$ can vary from 71.2°C to 84.9°C. Next, we use a correction step as described in Section 2.3.3 to obtain the fine temperature map within $CGC(0,0)$. The overall runtime of the second simulation scheme for obtaining both the coarse and the fine temperature maps is only 70*m*sec, which is an order of magnitude faster than the direct application of Algorithm II with a fine grid over the entire chip. In Fig. 2.13(b) and (c), we show the fine temperature map within $CGC(0,0)$ achieved after the correction step and the relative error compared with the result obtained using the first simulation scheme. We can see that the maximum relative error is only about 1.3%. This demonstrates that using Algorithm III with a coarse grid and the correction scheme can indeed achieve a local accuracy comparable to that obtained by Algorithm II with a fine grid over the entire chip, while the overall runtime is significantly reduced.

2.5 Summary

In this chapter, we presented three highly accurate thermal simulation algorithms based on the Green function method and analyzed in detail the relative advantages of each of the algorithms. Algorithm I combines the DCT and the table look-up technique to significantly reduce the time required for each evaluation of the Green function, and it is suitable for efficiently performing the localized analysis, where the effects of a few critical circuit blocks on the temperature distributions in a few field regions are sought. Algorithm II is based on the spectral domain analysis, and it takes advantage of the high efficiency of the FFT algorithm in transforming signals between the space and spectral domains. For full-chip thermal simulations, it can achieve an $O(\mathcal{N}_{gs} \times \log(\mathcal{N}_{gs})) + O(\mathcal{N}_{gf} \times \log(\mathcal{N}_{gf}))$ asymptotic time complexity as opposed to the $O(\mathcal{N}_{gs} \cdot \mathcal{N}_{gf})$ complexity of Algorithm I, where $\mathcal{N}_{gs}$ and $\mathcal{N}_{gf}$ are the total number of grid cells in the source and field planes, respectively. Algorithm III is a combination of both Algorithm I and Algorithm II, and it reflects the idea of the pre-corrected FFT. Its key application area is the full-chip thermal simulation with different accuracy requirements over the same chip, such as in the mixed signal design environments, where the analog blocks often have more stringent requirements on the accuracy of thermal simulations over the digital blocks. Experimental results show that all three algorithms can achieve around 1% errors compared with that of a commercial computational fluid dynamics software package for thermal analysis, while at the same time gaining orders

of magnitude speedups over the classical Green function methods.

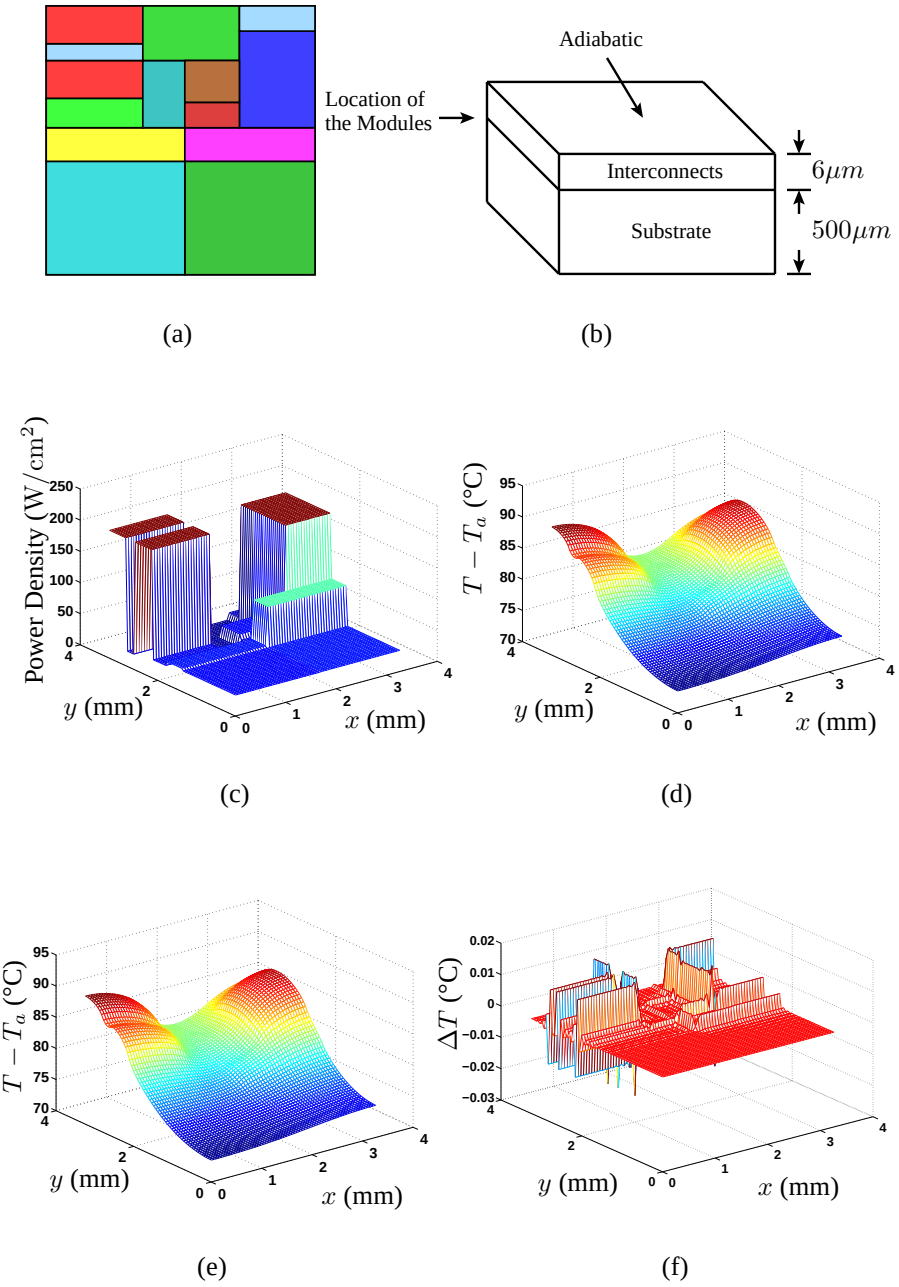
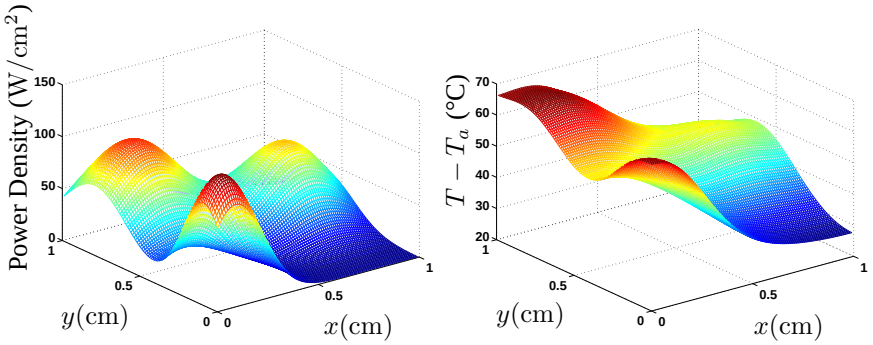


Figure 2.11: Power and temperature distribution of a realistic chip (a) floorplan (b) schematic of the substrate and interconnect layers (c) power distribution (d) temperature distribution obtained using Algorithm I (e) temperature distribution obtained using Algorithm II (f) difference in the temperature distribution map obtained using the two algorithms.



(a) (b)

Figure 2.12: Cell level power density and temperature distribution of a 1cm$\times$1cm chip (a) power density distribution (b) temperature distribution.

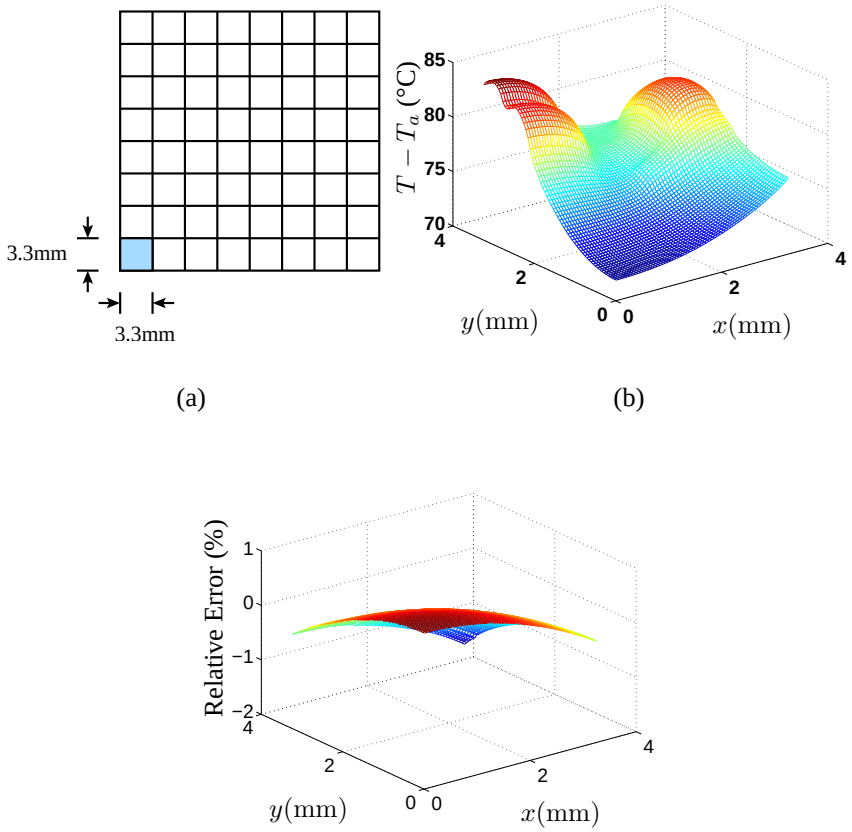


Figure 2.13: Effectiveness of Algorithm III (a) location of the coarse grid cell $CGC(0,0)$ that has higher requirement on thermal simulation (b) temperature map within $CGC(0,0)$ calculated using Algorithm III (c) relative error of Algorithm III compared with Algorithm II applied with a fine grid over the entire chip.

Chapter 3

Efficient Module Assignment for Pin-Limited Designs under the Stacked-Vdd Paradigm

3.1 Design Considerations in Stacked-Vdd Circuits

As stated in the introduction, power pins constitute the bulk of all I/O pins, and they introduce deliberate redundancy in delivering the same signal values so as to lower the IR and $L\frac{di}{dt}$ noise in the power grids. If, by some means, we could reduce the density of the currents flowing through the power grids, the number of power pins required to supply currents to the circuit would also be reduced accordingly, and the pin limitation

bottleneck could see significant relief. In [RSHK05], a high-tension power delivery scheme was proposed to reduce power grid noise and the effect of electromigration. In this new circuit paradigm, logic blocks are stacked several levels high and power is delivered to the circuit as multiples of the regular supply voltage V_{dd}. Next, the delivered high-tension supply voltage is divided into several Vdd domains each of which has a range of V_{dd}, and circuit blocks are distributed to different Vdd domains. Voltage regulators are used to control the voltage levels of internal supply rails.

An example of a two level stacked-Vdd circuit is shown in Fig. 3.1. The advantage of this new circuit structure is that when logic blocks are stacked n levels high and the current requirements between logic blocks operating in different Vdd domains are balanced, the current flowing through each external power grid would be reduced to $\frac{1}{n}$ of the original value, where the words external power grid refer to a power grid that is connected to power pins, i.e., nV_{dd} and GND rails in an n level stacked-Vdd circuit. Therefore, the noise and electromigration issues would be significantly alleviated.

Clearly, this circuit structure can be used to reduce the number of power pins required by a chip because of the reduced current flows in the external power grids. An important consideration in the design of a stacked-Vdd circuit is the current balance between logic blocks operating in different Vdd domains. If the currents are not balanced, the difference will flow through voltage regulators. This will not only lead to

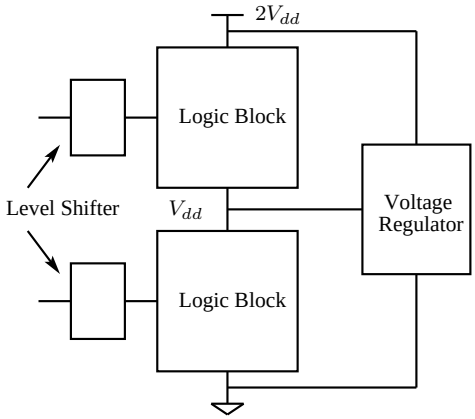


Figure 3.1: A schematic of a 2-level stacked-Vdd circuit structure.

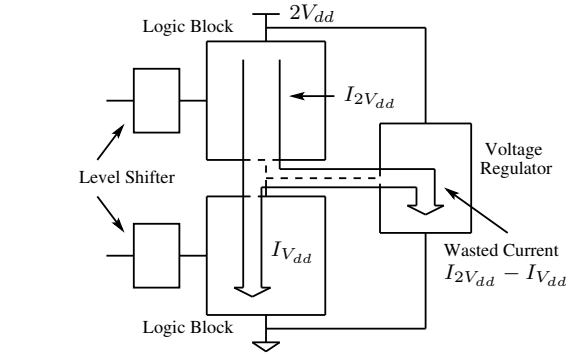
unnecessary power waste[1], but also increase the currents flowing through the external power grids, and therefore require a larger number of power pins in order to maintain the same level of signal integrity. In Fig. 3.2(a), we show an example of unbalanced current flow between modules operating in the two different Vdd domains. It can be seen that a current $|I_{2V_{dd}} - I_{V_{dd}}|$ will be wasted in the voltage regulator, where $I_{2V_{dd}}$ and $I_{V_{dd}}$ are the currents flowing through the two circuit blocks, respectively. The current balance issue has been addressed at the circuit level in [GK05] for architectures that contain parallel structures, where a data control circuitry is designed to distribute the work load at run time so as to reduce the current imbalance.

A more subtle issue associated with assigning circuit blocks to different Vdd domains is that the current balance must be maintained locally. The importance of this

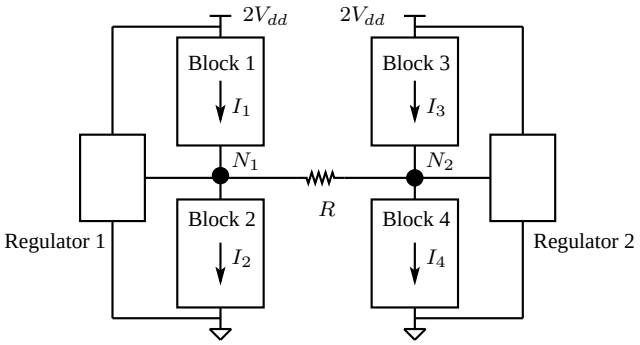
[1]Because the power wasted in a voltage regulator is proportional to the current flowing through it, we will use the words wasted power and wasted current interchangeably in this chapter.

point can be clearly seen in Fig. 3.2(b). We assume that Block 1 and Block 2 are physically close to each other in the layout, and the same is true for Block 3 and Block 4. However, Block 1 and 2 are far away from Block 3 and 4. As a result, the resistance R associated with the internal power rail between the two parts of the circuit can not be ignored. Since the nodes marked N_1 and N_2 are both maintained at voltage level V_{dd} by regulators, there will be no current flowing through the resistance R. If $I_1 = I_2$ and $I_3 = I_4$, local current balance is achieved and no power is wasted in the voltage regulators since no current flows through them. If, on the other hand, $I_1 > I_2$, $I_3 < I_4$, but $I_1 + I_3 = I_2 + I_4$, then although the currents are still balanced globally, there will be a current $I_1 - I_2$ flowing through Regulator 1 and a current $I_4 - I_3$ flowing through Regulator 2 because there is no current flowing through the resistance R. Therefore, some amount of power will be wasted in the regulators under this situation.

Another important issue that has to be considered in the design of a stacked-Vdd circuit is at which level should the circuit be partitioned into different Vdd domains. Note that a level shifter is required at the output of a logic block if it is used to drive another logic block operating in a different Vdd domain. If each logic block corresponds to a cell at placement level, too many level shifters will have to be used which not only leads to a significant overhead in terms of silicon area, but also impairs the performance of the circuit because of the extra delays caused by level shifters. In this paper, we address the module assignment problem at the floorplanning level where



(a)



(b)

Figure 3.2: Power wasted in voltage regulators (a) if the currents consumed by the logic blocks operating in the two different Vdd domains are not balanced, the difference will flow through voltage regulators and present itself as wasted power (b) current balance must be maintained locally in order to maximally reduce the power waste.

the number of modules is usually not very large. Therefore, the performance degradation and the area waste caused by level shifters can be largely ignored. In addition, unlike [GK05], we do not impose the restriction that the circuit contains parallel processing units. Instead, we utilize the observation that the operations of many modules on a VLSI chip are correlated, e.g., the modules on a pipelined data path tend to be on

at the same time.

3.2 Problem Formulation and Module Assignment Algorithm

Because the two level stacked-Vdd circuit provides a good tradeoff between chip performance and engineering complexity, it will be the focus of this work, and the primary steps of our module assignment algorithm include

- Obtaining a floorplan that contains both regular modules and voltage regulators so that the regulators are distributed relatively uniformly across the chip.
- Using the power simulation results over a set of benchmark programs to characterize the correlation between modules, which is represented in the form of a graph.
- Using an iterative approach to perform a max-cut partitioning of the graph, which corresponds to the assignment of modules to the two different Vdd domains.

In what follows, we will provide the details of the flow shown above, and special emphasis is placed on the partition-based module assignment step.

3.2.1 Problem formulation

In this section, we assume that a floorplan including both regular modules and voltage regulators is given, and the primary objective is to assign modules to different Vdd domains so as to achieve the maximal current balance. The approach that is used to obtain a floorplan in which the voltage regulators are distributed relatively uniformly across the die will be shown in the next section. The module assignment problem for a two level stacked-Vdd circuit is formulated as follows:

Given a floorplan including the location and size of each module and voltage regulator, the structure of the power grids, a set of current consumption traces of modules obtained through simulations over a set of benchmark programs, find the assignment of modules to the two different Vdd domains so that the total power wasted by voltage regulators is minimized.

Since the floorplan and the structures of the power grids are given as the input to the problem, we could run detailed simulations to obtain the trace of current flowing through each of the voltage regulators, and therefore calculate the wasted power. However, one simulation is required for each candidate assignment of modules, and the overall runtime of this scheme becomes exponential with respect to the number of modules in the layout, which makes it impractical for real design problems. In what

follows, we will show how to estimate the current flowing through each voltage regulator and how to obtain the assignment of modules by solving a single graph partitioning problem.

3.2.2 Estimation of the current flowing through a regulator and the formulation of the graph partitioning problem

Note that the tapping points of voltage regulators to the internal power grid, i.e., the connecting points between voltage regulators and the V_{dd} rail, have properties that are similar to those of power pins. For well designed regulators, they provide rather stable voltage levels at V_{dd}. In [Chi04], it was demonstrated that each module primarily draws currents from nearby power pins, and the same observation can be applied to the tapping points of voltage regulators. Assume we have K voltage regulators distributed across the chip. Each regulator is represented by the point it taps into the V_{dd} grid. As shown in Fig. 3.3, we can divide the chip into K regions accordingly such that there is one regulator in each region and the i^{th} region contains all the points on chip that primarily draw(sink) currents from(to) the i^{th} regulator. The division of the chip into non-overlapping regions can be achieved through meshing the entire die area using a fine grid and calculating the value of certain metric associated with each grid cell to determine which region it belongs to. This metric could be distance based, i.e., each cell belongs to the region that is controlled by the nearest voltage regulator, or it could

be based on other criterion determined by our understanding of the power grids and the accuracy requirements. In our implementation, we choose to use the Euclidean distance as the metric, and therefore, the resulting division of the chip corresponds to the Voronoi diagram of the region enclosed by the chip boundary. However, we emphasize that our algorithm is not tied to the Voronoi diagram because the metric calculating part is an independent function and it can be easily modified to use a different metric.

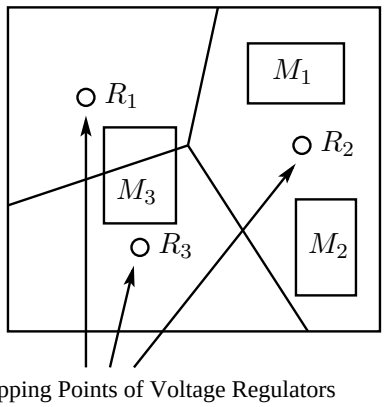


Figure 3.3: Partitioning of the chip into disjoint regions each of which is controlled by a voltage regulator.

After the chip is partitioned into disjoint regions, we can assume that the imbalance current caused by the modules located in a particular region only goes through the regulator in the same region. For example, if modules M_1 and M_2 are the only modules located in the region corresponding to tapping point R_2, M_1 works between the $2V_{dd}$ and V_{dd} rails and draws a current I_1, M_2 works between the V_{dd} and GND rails and draws a current I_2, and $I_1 > I_2$, then a current $I_1 - I_2$ will flow through the voltage

regulator tapped at point R_2. If a module is located at the boundary between multiple regions, e.g., M_3 in Fig. 3.3, it will be decomposed into several sub-modules with one sub-module in each region it overlaps and with the constraint that all sub-modules must be assigned to the same Vdd domain.

Let us focus on a particular region corresponding to a particular voltage regulator. Assume the modules located in this region are M_1, M_2, ..., M_n, where the current flowing through module M_i as a function of time t is given by $I_i(t)$. Because voltage regulators can only respond to the low to mid frequency components of the imbalance currents while the high frequency components are usually handled by on-chip decaps, we pre-process the input current traces obtained through cycle-accurate power simulations to smooth out the high frequency components in the current signals. The smoothing process is performed by first dividing the entire time sequence of the simulated program into consecutive segments of clock cycles as shown in Fig. 3.4, and then for each segment, taking the average current consumption of each module. Therefore, $I_i(t)$ should be understood as containing only the low to mid frequency components of the current flowing through module M_i.

If we associate a 0/1 integer variable x_i with module M_i defined as

$$x_i = \begin{cases} 0 & \text{if } M_i \text{ operates between the } 2V_{dd} \text{ and } V_{dd} \text{ rails} \\ 1 & \text{if } M_i \text{ operates between the } V_{dd} \text{ and } GND \text{ rails} \end{cases} \tag{3.1}$$

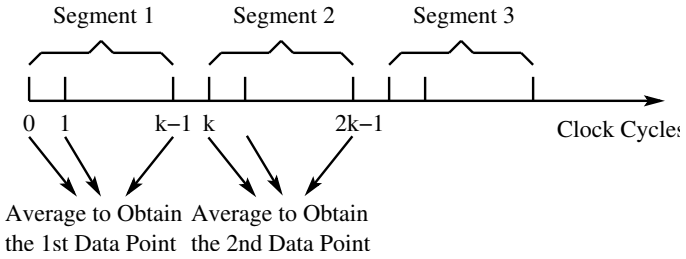


Figure 3.4: Smoothing the current trace to remove the high frequency components.

the total current flowing through the voltage regulator at time t, which is proportional to the instantaneous power wasted in the same regulator, will be approximated by

$$I_R(t) = \left| \sum_{i=1}^{n} I_i(t) \cdot (1 - x_i) - \sum_{i=1}^{n} I_i(t) \cdot x_i \right| = \left| \sum_{i=1}^{n} I_i(t) \cdot (1 - 2x_i) \right| \tag{3.2}$$

The objective is to minimize the average wasted current $\overline{I_R(t)}$. It is easy to see that this is a NP-complete 0/1 integer linear programming (ILP) problem that can only be solved using heuristics. In our work, instead of minimizing $\overline{I_R(t)}$, we minimize $\overline{I_R(t)^2}$, which can be written as

$$\overline{I_R(t)^2} = \overline{\left(\sum_{i=1}^{n} I_i(t) \cdot (1 - 2x_i) \right)^2} = \overline{\left(\sum_{i=1}^{n} I_i(t) \right)^2} - 4 \sum_{i<j} \overline{I_i(t) I_j(t)} \, (x_i + x_j - 2x_i x_j) \tag{3.3}$$

When going from the first to the second step in equation (3.3), we used the equality $x_i^2 = x_i$ when x_i is a 0/1 integer variable. It is easy to see from (3.3) that to minimize

$\overline{I_R(t)^2}$ is equivalent to maximize

$$S = \sum_{i<j} \overline{I_i(t)I_j(t)}(x_i + x_j - 2x_i x_j) \tag{3.4}$$

and

$$x_i + x_j - 2x_i x_j = \begin{cases} 0 & \text{if } x_i = x_j \\ 1 & \text{if } x_i \neq x_j \end{cases} \tag{3.5}$$

The intuition behind (3.4) and (3.5) is that if modules M_i and M_j are in different Vdd domains, a positive term $\overline{I_i(t)I_j(t)}$ will appear in summation (3.4), but not otherwise. Based on this observation, the problem of maximizing S in (3.4) can be cast into the following equivalent graph partitioning problem.

Given a weighted graph $G = (V, E, W)$ where $V = \{V_1, V_2, \ldots, V_n\}$, $E = \{(V_i, V_j) | V_i, V_j \in V\}$, and the weight set $W = \{w(V_i, V_j) = \overline{I_i(t)I_j(t)} | V_i, V_j \in V\}$, find a two way partitioning of the graph so that the total cut of the edges crossing the partition is maximized.

Up to this point, we have been considering one of the K disjoint regions over the chip and the modules that are completely located within it. A graph partitioning problem is formulated to assign modules to the two different Vdd domains so as to minimize the power wasted in the voltage regulator controlling the region. For the

entire chip, a similar graph partitioning problem can be formulated where node V_i in the graph corresponds to module M_i in the layout. The only difference from the problem formulation shown above is in calculating the weight of each edge in the graph. Let S_i represent the area of the i^{th} module, and denote the overlap area between the i^{th} modules and the k^{th} region over the chip by S_{ik}. The weight of edge (V_i, V_j) is calculated by

$$w(V_i, V_j) = \left(\sum_{k=1}^{K} \frac{S_{ik} S_{jk}}{S_i S_j} \right) \overline{I_i(t) I_j(t)} \tag{3.6}$$

The intuition behind (3.6) is that for any pair of modules, only the portions that are located in the same region over the chip count toward the calculation of the correlation between them. A further implication of (3.6) is that if modules M_i and M_j are completely separated into two disjoint regions, the weight $w(V_i, V_j)$ will be zero, and therefore, the corresponding edge can be removed from the graph. In Fig. 3.5, we show the resulting graph corresponding to a chip that contains five modules and is divided into two regions. The circles marked R_1 and R_2 represent the tapping points of voltage regulators to the V_{dd} rail. Note that there is no edge connecting nodes V_2 and V_5 because modules M_2 and M_5 are completely separated into two disjoint regions controlled by two individual voltage regulators. Similarly, there is no edge between nodes V_3 and V_5. For the modules that overlap with the boundary between the two regions, i.e., M_1 and M_4, we assume that α portion of M_1 and β portion of M_2 are located in the region controlled by voltage regulator R_1, and correspondingly, $1 - \alpha$ portion of

module M_1 and $1-\beta$ portion of module M_2 are located in the region controlled by voltage regulator R_2. According to (3.6), the weights of edges (V_1, V_2) and (V_1, V_4) are calculated by

$$w(V_1, V_2) = \alpha\overline{I_1(t)I_2(t)} \tag{3.7}$$

and

$$w(V_1, V_4) = [\alpha\beta + (1-\alpha)(1-\beta)]\overline{I_1(t)I_2(t)} \tag{3.8}$$

respectively.

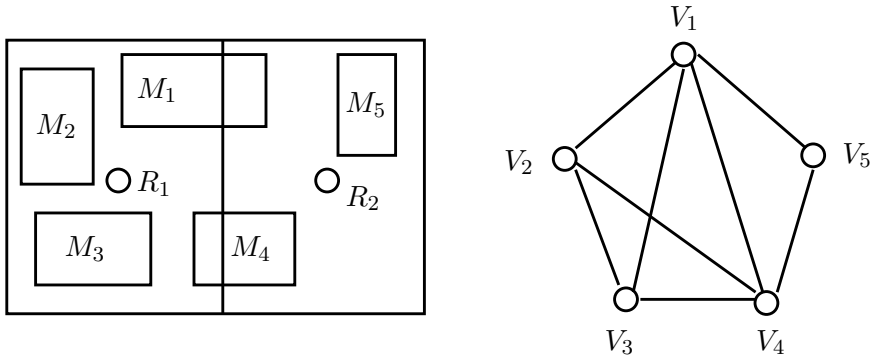


Figure 3.5: An example of graph construction. Module M_i in the layout corresponds to node V_i in the graph.

3.2.3 Graph partitioning heuristic

A two step heuristic is used to partition the node set of the graph into two sub-sets so that the total cut is maximized. In the first step, we greedily assign nodes to the sub-sets so as to obtain a reasonably good initial partition. The primary operations

involved in this step include sorting the weighted edges in decreasing weight order and examining them consecutively. For each edge under examination, if none of the two nodes associated with it has been assigned to a partition, we assign them to two different partitions. If one of the nodes has been assigned but not the other, we assign the other node to the opposite partition. Finally, if both nodes have been assigned, we skip the current edge and proceed to the next edge in the sorted edge list.

After the initial node assignment, we use a F-M like algorithm to iteratively improve the partition and increase the cut size. Since the F-M algorithm is a rather mature method, we will not go into the details of every step of our implementation. Instead, we will highlight the differences between our algorithm and the conventional F-M algorithm, and then list our implementation in the form of a pseudo-code. Readers who are interested in the conventional F-M algorithm are referred to [SY95]. The first difference between our algorithm and the conventional F-M algorithm is that the latter requires the node weights to be balanced between the two partitions, while in our case, nodes carry no weight and maximizing the total cut size is our only concern. The second difference between the two algorithms is that while the F-M algorithm tries to minimize the cut size, our algorithm tries to maximize it. Therefore the calculation of the gain of each move should be modified.

The initial gain of moving a node V_i from its current partition to the opposite par-

tition is given by

$$g(V_i) = \sum_{V_j \in FP(V_i)} w(V_i, V_j) - \sum_{V_j \in TP(V_i)} w(V_i, V_j) \tag{3.9}$$

where $FP(V_i)$ contains all the nodes that are in the same partition as node V_i and are connected to V_i, and $TP(V_i)$ contains all the nodes that are in the opposite partition to node V_i and are connected to V_i. For example, in the partition shown in Fig. 3.6, the initial gain of moving node V_2 from its current partition to the opposite partition is given by $g(V_2) = w(V_1, V_2) - w(V_2, V_4) - w(V_2, V_5)$.

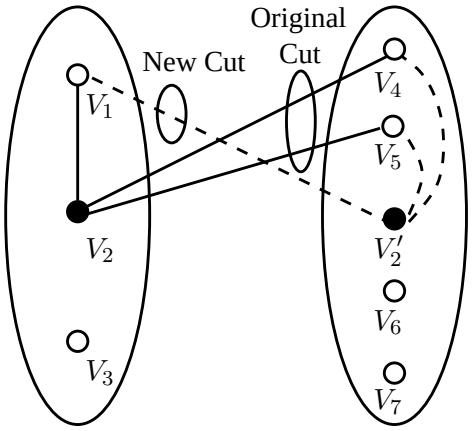


Figure 3.6: An example for gain calculation in the F-M like algorithm showing the cut set before and after the node V_2 is moved to the opposite partition.

When a node V_i is moved from one partition to the other, the gains associated with the nodes connected to V_i must be updated. Assume node V_j is connected to V_i, the gain associated with V_j should be updated as follows:

- If V_j and V_i were in the same partition before the movement of V_i

$$g(V_j)^{new} = g(V_j)^{old} - 2w(V_i, V_j) \tag{3.10}$$

- if V_j and V_i were in different partitions before the movement of V_i

$$g(V_j)^{new} = g(V_j)^{old} + 2w(V_i, V_j) \tag{3.11}$$

The complete F-M like algorithm that is used to iteratively improve the cut size of the partition is listed in Fig. 3.7.

3.3 Floorplanning Involving Voltage Regulators and the Complete Algorithm Flow

In this section, we first briefly describe the technique that is used to obtain the floorplan that contains both regular modules and voltage regulators. Then we present the overall flow of the algorithm. We have used Parquet [AM03] with a modified cost function to perform the floorplanning. Besides the conventional optimization objectives such as wirelength, an additional objective that is unique to our problem is that the voltage regulators, which are also considered as modules, should be distributed

1. *do*
2. node_moved ← *false*;
3. Calculate the initial gain of each node;
4. *for* i ← 1 *to* number_of_nodes
5. Select the free node that has the maximum gain and call it $V_{s(i)}$;
6. Lock node $V_{s(i)}$;
7. Update the gains of the nodes connected to $V_{s(i)}$;
8. *end for;*
9. Find the number K such that $G = \sum_{i=1}^{K} g(V_{s(i)})$ is maximized;
10. *if* $G > 0$
11. Make the K moves permanent;
12. node_moved ← *true*;
13. Free all locked nodes;
14. *end if*;
15. *while*(node_moved == *true*);

Figure 3.7: Iterative improvement of the partition through a F-M like algorithm.

relatively uniformly across the die. This will help reducing the power grid noise in the V_{dd} rail. We have achieved this new objective through modifying the cost function in Parquet. Specifically, a penalty term in the form

$$penalty = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \frac{1}{d(R_i, R_j)} \tag{3.12}$$

is added to the cost function used in simulated annealing, where R_i with $i = 1, 2, \ldots, K$ represent voltage regulators, and $d(R_i, R_j)$ is the distance between the points where regulators R_i and R_j tap into the V_{dd} grid. The intuition behind (3.12) is that when regulators R_i and R_j come closer to each other, the penalty term will become larger, and therefore, the corresponding floorplan will have a higher probability of being rejected.

The overall flow of the complete algorithm is given in Fig. 3.8.

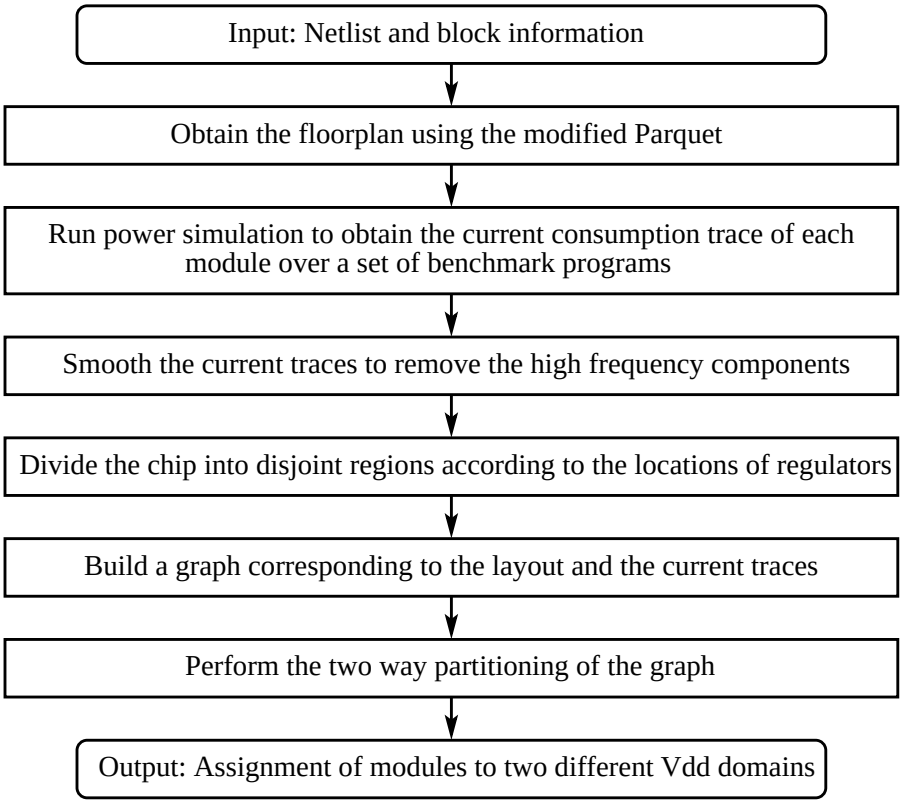
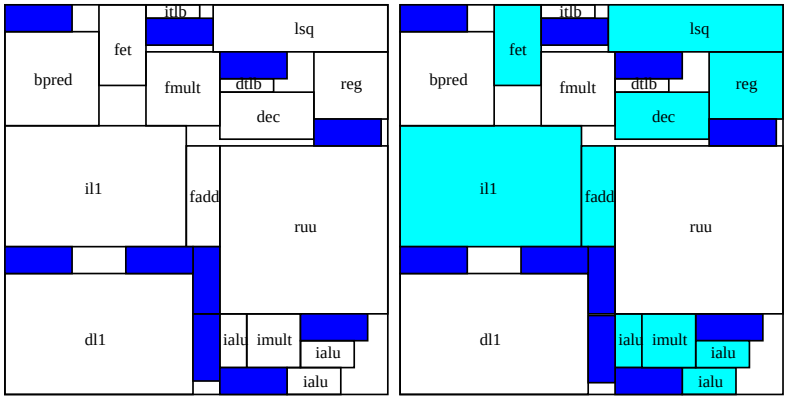


Figure 3.8: Complete algorithm for assigning modules to two different Vdd domains.

3.4 Experimental Results

3.4.1 Floorplanning using modified Parquet

Fig. 3.9(a) shows the floorplan of a microarchitecture used in [NLS06] with ten voltage regulators inserted. The microarchitecture is based on the DLX architecture [PH96] and the dark regions represent voltage regulators. All modules are assumed to be hard and the floorplanning is performed using Parquet with the modified cost function as described in the previous section. We can see that with the simple modifications we have made to Parquet, the voltage regulators can be reasonably uniformly distributed across the die.



(a) (b)

Figure 3.9: Floorplan with voltage regulators (a) floorplan (b) assignment of modules to the two different Vdd domains.

3.4.2 Calculation of the edge weight in the graph and module assignment using the partition-based algorithm

After the floorplan is obtained, we simulate the architecture using the eight benchmark programs contained in the SPEC 2000 suite that cover both integer and floating-point operations. The eight programs with their respective instruction counts in billions are listed in Table 3.1. To speed up the simulations, we utilize SMARTS [WWFH03], a periodic sampling technique to obtain the current consumption trace of each module for each of the benchmark programs. Specifically, we start simulating a program at clock cycle 0 and continue the simulation for s consecutive clock cycles. The average current consumption of each module is calculated during this period of time. Next, we stride forward and start the simulation again at the l^{th} clock cycle with $l \gg s$. The average current consumption of each module is calculated again for the s clock cycles that follow. This process continues until the entire program is completed. We can see clearly that by using this strategy, we obtain a sampled sequence of the average current consumption trace of each module.

Note that the time averaged total current consumption of the chip may vary significantly while running different programs, and the objective of our algorithm is to obtain a partition of modules that is deemed good across the entire benchmark suite, i.e., we want to ensure that each benchmark program imposes a similar weight in affecting the partition of modules. To achieve this objective, we normalize the current

Benchmark	Type	Instruction (B)
vpr	Integer	11
gcc	Integer	35
gzip	Integer	63
bzip2	Integer	94
parser	Integer	301
art	Floating-point	54
equake	Floating-point	175
mesa	Floating-point	305

Table 3.1: Benchmarks from the SPEC 2000 suite, along with the reference instruction counts in billions.

consumption traces associated with each program so that the normalized average total current consumption of the chip while running that program becomes 1. In Table 3.2, we show an example that contains only two modules and for which the simulation result is collected at only two sample points. The total current consumption of the design at the two sample points are 30mA and 50mA, respectively. Therefore, the average total current consumption of the design is (30mA+50mA)/2 = 40mA. We use 40mA to normalize the current consumptions of the two modules at the two sample points, which generates the unit-less numbers shown on the right of the table for the normalized current consumption traces. Next, we concatenate the normalized current consumption traces for all of the programs such that a combined current trace is obtained for each module. The combined current traces are used to build the graph as described in Section 3.2.2, and the graph is partitioned using the algorithm presented in Section 3.2.3.

	I_{M_1}	I_{M_2}	I_{total}	$I^N_{M_1}$	$I^N_{M_2}$
Sample 1	10mA	20mA	30mA	0.25	0.5
Sample 2	20mA	30mA	50mA	0.5	0.75

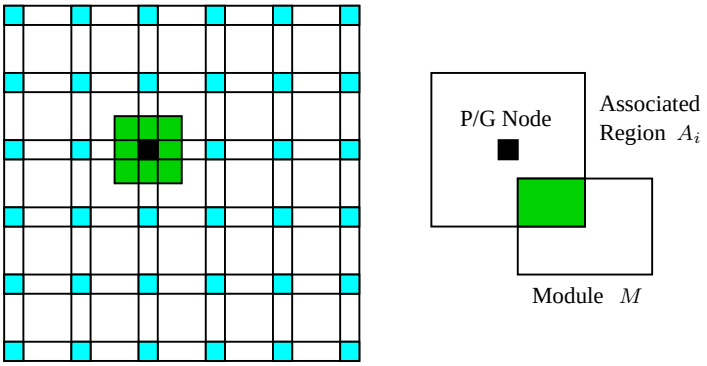
Table 3.2: An example of normalizing the current consumption traces.

The result of the partition is shown in Fig. 3.9(b), where the lightly darkened regions represent the modules operating between the $2V_{dd}$ and V_{dd} rails while the white regions represent the modules operating between the V_{dd} and GND rails.

3.4.3 Experimental setup for the Validation of the module assignment

To validate that the module assignment obtained by our algorithm indeed reduces the power wasted in voltage regulators, we first use a regular grid structure similar to that shown in Fig. 3.10(a) to represent the V_{dd} rail, and we assume that current sources are attached to the nodes in the power grid, which model the currents consumed by modules. Next, we associate each node in the power grid with a region on the chip and assume that all the modules located in that region only source(sink) currents from(to) that particular node, e.g., the dark region surrounding the black colored node in Fig. 3.10(a) is associated with that node. For node i with the associated region A_i, if only part of a module M overlaps with A_i, then only the corresponding portion of the current consumed by M is attached to node i. For example, in Fig. 3.10(b), only

25% of module M overlaps with the area associated with the power grid node. As a result, only 25% of the current consumed by module M is attached to that particular node.



(a) (b)

Figure 3.10: Testing the actual wasted power (a) the structure of the V_{dd} grid and the region that source(sink) current from(to) a particular node (b) calculating the current source attached to a power grid node when the module only partially overlaps the region associated with the node.

After the value of the current source attached to each power grid node is calculated, a modified nodal analysis (MNA) equation in the form

$$\begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} \begin{pmatrix} V \\ I_{reg} \end{pmatrix} = \begin{pmatrix} I_s \\ V_{reg} \end{pmatrix} \tag{3.13}$$

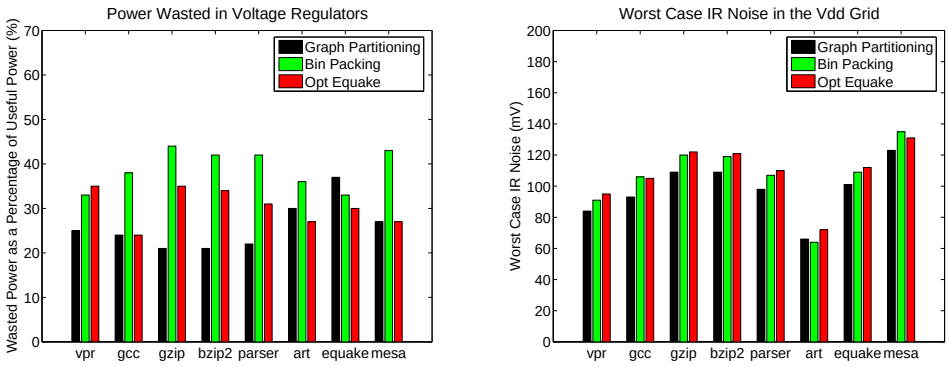
can be established to calculate the current flowing through each of the voltage regulators, and therefore the wasted power. Here G_{ij}'s are submatrices of the coefficient matrix, V is the vector of nodal voltages, I_{reg} is the vector containing the currents

flowing through voltage regulators, I_s is the vector of known current sources attached to the nodes in the V_{dd} grid, and V_{reg} is a vector whose components are all V_{dd}'s, the voltage level maintained by the regulators.

We compare three module assignment schemes, one using the algorithm presented in this chapter, a second using a bin-packing technique, and a third using the assignment optimized for one particular benchmark program, i.e., *equake*. For the bin-packing technique, we take the combined current traces as described in the previous subsection and calculate the average current consumption of each module. Then the module assignment is obtained such that the difference between the currents consumed by the modules in the two different Vdd domains is minimized. Since the floorplan contains only 16 regular modules, we can afford to enumerate all possible module assignments and obtain the best bin-packing result. For the last approach, the module assignment is generated completely based on the current consumption traces of the *equake* program, and the purpose of this experiment is to see how the assignment performs across the benchmark suite when it is optimized with respect to only one particular program.

3.4.4 Result of comparison between different module assignment schemes

We emphasize that our partition-based approach is highly efficient and the runtime of obtaining the assignment of modules for the DLX architecture is only 0.01sec on a desktop with a 3.2GHz Pentium-4 CPU. In what follows, we show the result of validation using the procedure described in the previous subsection.



(a) (b)

Figure 3.11: Comparison between the module assignments using the partition-based approach, the bin-packing technique, and the one that is optimized with respect to *equake* in terms of (a) the total power wasted in voltage regulators, and (b) the worst case IR noise in the V_{dd} grid.

Fig. 3.11(a) shows the comparison of the power wasted in voltage regulators between the three module assignment schemes. In obtaining the figure, we divide the total power wasted in voltage regulators by the total power consumed by regular mod-

ules, where the latter is termed "useful power" in the figure. We can see clearly that the design obtained using our approach has about 32% less power waste on average compared with the design where the module assignment is performed using the bin-packing technique. Therefore, our approach is more suitable for chip designs where the power constraint is stringent. In addition, it is also clear that although the module assignment optimized for *equake* wastes less power in executing that particular program and another benchmark program *art*, it is not as good as our design in general and wastes about 23% more power on average.

It is important to notice that a design optimized for power also tends to achieve low IR noise in the V_{dd} grid. This is because to reduce the power waste, good current balance must be maintained locally as described in Section 3.1, which is beneficial to reducing the IR noise since the current consumed by a module in one Vdd domain will immediately be recycled by some nearby modules in the other Vdd domain without flowing through a long resistive path in the power grid. In Fig. 3.11(b), we compare the maximum IR noise encountered in the V_{dd} grid when the module assignment is performed using the three different schemes presented above, respectively. It can be seen that our design technique achieves very reasonable IR noise, which demonstrates that while our partition-based module assignment scheme can significantly reduce the power waste, it does not sacrifice the noise performance of the design.

Note that for a floorplan with n modules, the entire design space contains 2^n dif-

Benchmark	Avg P_{waste} (%)	Min P_{waste} (%)	Our P_{waste} (%)	Avg Noise (mV)	Min Noise (mV)	Our Noise (mV)
vpr	51	23	**25**	102	83	**84**
gcc	49	29	**24**	113	93	**93**
gzip	52	25	**21**	128	108	**109**
bzip2	52	26	**21**	129	109	**109**
parser	51	25	**22**	116	96	**98**
art	48	26	**30**	75	60	**66**
equake	47	25	**37**	118	98	**101**
mesa	49	28	**27**	140	121	**123**

Table 3.3: Comparison between the random module assignment and the assignment obtained using the partition-based approach in terms of the wasted power and the worst case IR noise in the V_{dd} grid. The wasted power is given as a percentage of the useful power.

ferent module assignments. It is not practical to test all possible solutions since each test requires a full simulation of the input current traces for each of the benchmark programs, which may take hours to complete, and for 2^n different candidate designs, the overall runtime of the enumeration becomes intractable. However, it is interesting to sample the design space and see how the performance of the module assignment obtained using our partition-based approach compares with other sample designs. To achieve this objective, we have generated 50 different module assignments randomly and calculated the total power wasted in voltage regulators and the worst case IR noise in the V_{dd} grid for each of the benchmark programs.

The result of comparison is shown in Table 3.3. We can see that the total power wasted in voltage regulators in our design is rather close to the minimum wasted power obtained through random experiments. We also emphasize that the minimum wasted

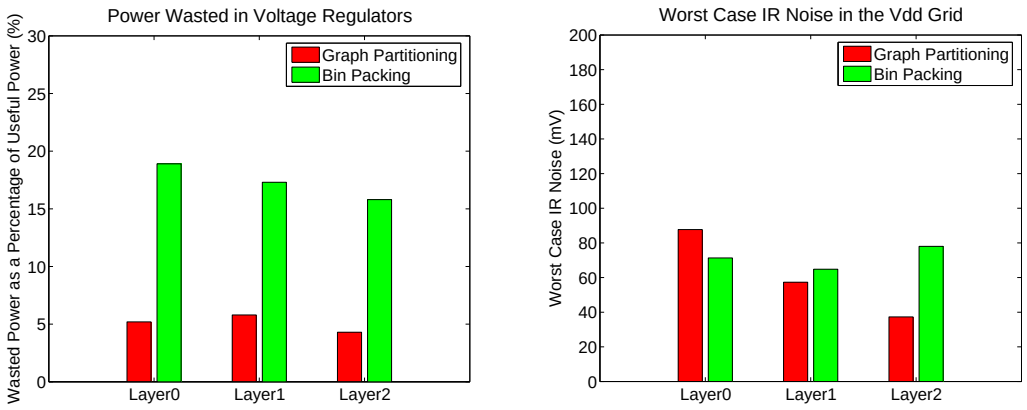
power and IR noise data shown in Table 3.3 are not achieved by a unique module assignment among the 50 randomly generated designs, e.g., the design that has the minimum power waste for *vpr* is different from the design that has the minimum power waste for *art*. However, the design generated by our module assignment technique achieves consistently good result across the benchmark programs. This demonstrates the effectiveness of our approach, since in reality, a chip can only implement one design no matter how many different programs it will run in the future.

3.4.5 An example of a 3D circuit

We have also studied the effectiveness of using the stacked-Vdd paradigm in 3D circuit design. A three-layer 3D circuit structure is used and the floorplan of the n300 benchmark from the GSRC suite is generated. The three active layers contain 118, 92, and 90 modules, respectively, and the final footprint area of the die is scaled to $1\mathrm{cm}^2$. Because of the lack of available current traces for GSRC benchmarks, we have assumed that the mean current consumption of each module is a random number between 100mA and 1000mA, and the instantaneous current consumption of module M_i is assumed to be varying randomly around its mean I_i^{mean}.[2] The assignment of

[2]The instantaneous current consumption of module M_i is given by $I_i(t) = I_i^{mean} \times (1 + \delta_g(t)) \times (1 + \delta_i(t))$, where $\delta_g(t)$ is a random number that remains the same for all of the modules, and $\delta_i(t)$ is a random number that varies from module to module. It is not difficult to see that $\delta_g(t)$ can be used to model the change of operations that affect the entire chip while $\delta_i(t)$ can be used to model the local variation in current consumption as a function of time t. In our experiment, $\delta_g(t)$ is randomly selected within the range $[-0.3, 0.3]$, and $\delta_i(t)$ is randomly selected within the range $[-0.2, 0.2]$.

modules to different Vdd domains is performed individually for each active layer, and the runtimes of the assignments are 1.98sec, 0.76sec, and 0.70sec, respectively. As is done previously, our module assignment strategy is compared with the bin-packing technique. For the bin-packing technique, it is impractical to enumerate all possible module assignments and find the best one in this example because of the large number of modules involved. As an alternative, we use hMetis [KAKS99] to perform a balanced two way partitioning of modules with the current consumption of each module as its weight, and the modules belonging to the same partition are assigned to the same Vdd domain.



(a) (b)

Figure 3.12: Comparison between the module assignment using the partition-based approach and the bin-packing technique in terms of (a) the total power wasted in voltage regulators, and (b) the worst case IR noise in the V_{dd} grid, for the 3D circuit example.

In Fig. 3.12, we compare the two module assignment schemes in terms of the total

power wasted in voltage regulators and the worst case IR noise in the V_{dd} grid for the three active layers in the 3D circuit. It can be seen that our module assignment approach results in a circuit that wastes far less power than the one generated by the bin-packing technique, and the noise performance of our design is also rather good.

As for the floorplan of the DLX architecture, we have also obtained 50 random module assignments for each active layer of the 3D circuit, and some of the critical statistics of the random experiments are listed in Table 3.4. Again, it is clearly seen that the power wasted in our design is very low compared with other sample designs from the random experiments.

Benchmark	Avg P_{waste} (%)	Min P_{waste} (%)	Our P_{waste} (%)	Avg Noise (mV)	Min Noise (mV)	Our Noise (mV)
Layer0	17.8	8.8	**5.2**	108	61	**88**
Layer1	18.5	9.1	**5.8**	78	48	**57**
Layer2	18.6	9.8	**4.3**	70	41	**37**

Table 3.4: Comparison between the random module assignment and the assignment obtained using the partition-based approach for a 3D circuit in terms of the wasted power and the worst case IR noise in the V_{dd} grid. The wasted power is given as a percentage of the useful power.

3.5 Summary

In this chapter, we presented a partition-based algorithm for efficiently assigning modules to different Vdd domains in a two level stacked-Vdd circuit. The objective is to minimize the power wasted in voltage regulators. The primary steps of the algo-

rithm include building a graph that represents the correlations between modules and performing a max-cut partitioning of the graph using a F-M like algorithm. Experimental results on a DLX architecture show that compared with assigning the modules to different Vdd domains using a bin-packing technique, the design generated by our algorithm wastes about 32% less power in voltage regulators, and therefore is more suitable for applications where the power constraint is stringent. Next, we tested 50 different cases where the modules are assigned to the two different Vdd domains randomly, and it was found that the total power wasted in voltage regulators in our design is rather close to the minimum wasted power obtained through random experiments. Finally, experiments on a 3D IC example show that our module assignment approach is equally effective in reducing the power waste in 3D ICs.

Chapter 4

Optimization of Integrated Spiral Inductors Using Sequential Quadratic Programming

4.1 Introduction

On-chip spiral inductors are key components in many RF integrated circuits (RFIC's) running at GHz frequency range. During the past few years, the design and optimization of integrated spiral inductors has attracted much interest in both the IC design and electronic design automation communities. The objective of inductor optimization may vary depending on the application. It could be high quality factor Q, small

area occupied by the device, or small parasitic effects, etc. In this work, we focus on the optimization of high-Q spiral inductors. Roughly speaking, there are three major loss mechanisms that degrade the quality factor of an on-chip inductor: the energy loss due to the series resistance of the spiral itself, the electric coupling between the spiral and the substrate, and the magnetically induced eddy current flowing in the substrate. In [YW98], the substrate loss due to eddy current is significantly reduced by inserting a patterned ground shield between the spiral and the substrate as shown in Fig. 4.1, and in [CAG93], both the electric and magnetic couplings to the substrate are practically eliminated by etching away the substrate beneath the spiral. The energy loss due to the series resistance of the spiral, however, can only be reduced by optimizing the geometrical parameters of the inductor such as the number of turns, the outer length, the width of each metal trace constituting the spiral, and the space between adjacent metal traces.

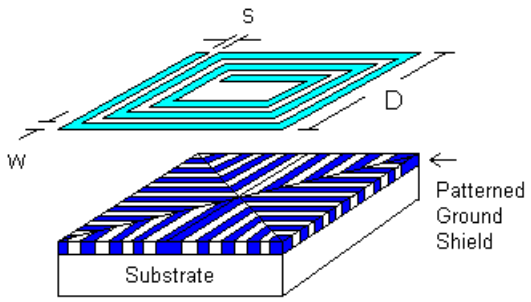


Figure 4.1: A three turn square spiral inductor with a patterned ground shield.

Several previous works have targeted the optimization of geometrical parameters of an integrated spiral inductor to increase its quality factor. Enumeration is used in [Nik] where the geometrical parameters are first discretized, then each combination of the resulting parameter values is simulated, and finally, the parameters that result in the highest Q are used in the design. This method, although intuitive and simple, can be highly inefficient, especially when the number of adjustable parameters becomes large because the time complexity of the enumeration method is exponential with respect to the number of optimization variables. In [HMBL99], geometric programming is used to solve the spiral inductor optimization problem. Geometric programming is a powerful mathematical programming method based on the assumption that both the objective function and the constraints are posynomial functions. To satisfy this requirement, the inductance of the spiral must be extracted using a approximate formula derived from curve-fitting a large number of pre-fabricated or pre-simulated designs. In addition, the entire device must be represented by a single π model with all the lumped parasitic components expressed in posynomial functions. The limitation of this method is that several high frequency effects which are significant at GHz frequency range cannot be handled by the simple closed form formulas, e.g., in expressing the inductance and the parasitic resistance of a spiral as posynomial functions, it is impossible to model the proximity effect which may become significant at frequencies as low as a few hundred MHz.

In this work, we propose using sequential quadratic programming (SQP) to optimize the quality factor of an integrated spiral inductor. SQP is an iterative mathematical programming technique based on the observation that almost any smooth continuous function can be locally approximated by a quadratic function and it has the desired property that the local convergence rate is superlinear if the starting point of the iterations is close enough to the optimal solution. Compared with enumeration, the SQP algorithm achieves at least an order of magnitude speedup which can significantly reduce the turn-around time of the design of spiral inductors, and compared with the geometric programming approach, SQP can be used with any physical model to optimize the device operating at any frequency, which makes it suitable to a broader range of applications. The SQP optimizer is built upon a spiral inductor extraction engine similar to that used in [NM98]. The quality factor and the effective inductance of the device are extracted from the two port Y parameters, and the well-known proximity effect and skin effect are handled automatically by meshing the metal traces in the longitudinal direction. We assume that a patterned ground shield exists beneath the inductor and thus the eddy current in the substrate is not modeled. In addition, to make the implementation simple, we have only worked on square spiral inductors. The rest of the chapter will be organized as follows. Section 4.2 formulates the inductance optimization problem. Section 4.3 presents the computation of the quality factor, effective inductance, and sensitivity of a spiral inductor with respect to design

parameters. Section 4.4 introduces the SQP algorithm. In Section 4.5, we show the experimental results, and the summary is provided in Section 4.6.

4.2 Problem Formulation

Fig. 4.1 shows the schematic of a square spiral inductor. Let n, D, w, and s be the number of turns, outer length, width of a metal trace, and the space between metal traces of the spiral, respectively. Let $Q(n, D, w, s)$ and $L(n, D, w, s)$ be the quality factor and the inductance of the spiral, and let L_{exp} and δ be the targeting inductance value and the tolerance allowed for the inductance to deviate from the targeting value. Then the inductance optimization problem is formulated as

$$\begin{aligned} maximize \quad & Q(n, D, w, s) \\ subject\ to \quad & (1-\delta)L_{exp} \leq L(n, D, w, s) \leq (1+\delta)L_{exp} \\ & 2n(w+s) \leq D \\ & D_L \leq D \leq D_U \\ & w_L \leq w \leq w_U \\ & s_L \leq s \leq s_U \end{aligned} \tag{4.1}$$

Here, D_L, D_U, w_L, w_U, s_L, and s_U are the lower and upper bounds of the corre-

sponding optimization variables, respectively. The number of turns n is treated as a parameter rather than a variable because it can only take discrete values. D, w, and s are treated as continuous variables for optimization purposes and may be rounded to the nearest grid point during the layout generation process. Since Q and L are both nonlinear functions of the optimization variables, our formulation produces a nonlinear optimization problem, which will be solved using the SQP algorithm. To successfully apply the SQP method, we must be able to accurately compute Q, L and their sensitivities with respect to D, w, and s. We will first discuss how to extract Q and L, and perform the sensitivity analysis in the next section. The detailed description of the SQP algorithm will be left to Section 4.4.

4.3 Extraction Engine

4.3.1 Inductance and quality factor extraction

For the simplicity of implementation, we have only considered the square spiral inductors in this work. The extension to other geometries such as octagonal spirals is straightforward although computationally more complicated. To extract the inductance and quality factor, the spiral is first divided into a series connection of metal segments. The length of each segment should not exceed a small fraction of the wavelength of the EM wave at the operating frequency of the device such that it is meaningful to

model the segments by lumped circuit elements. Fig. 4.2 shows a schematic of the inductor model that is used in the extraction where each segment is represented by an equivalent π model. Each series branch includes the self inductance and the parasitic resistance of the metal segment itself, and each parallel branch includes the coupling capacitance to the substrate and the conductance of the substrate. In addition, there are mutual inductances between parallel segments, and the coupling capacitance between each pair of parallel segments in adjacent turns are also modeled.

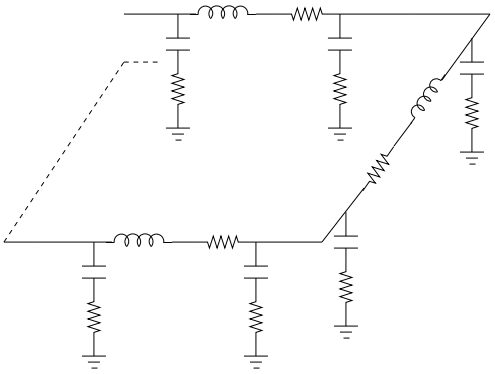


Figure 4.2: A distributed π model for square spiral inductors.

From [NM98], we know that using modified nodal analysis (MNA), the governing equation of the circuit can be expressed as

$$\begin{pmatrix} Y^C & D^T & -B^T \\ D & -Z^L & 0 \\ B & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{V} \\ \mathbf{I} \\ \mathbf{I_V} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ V_1 \\ V_2 \end{pmatrix} \tag{4.2}$$

where Y^C is the AC conductance matrix of the parallel branches, Z^L is the AC impedance matrix of the series branches, D is an upper bidiagonal matrix with 1's on the diagonal entries and -1's on the superdiagonal entries and $B = \begin{pmatrix} 1 & 0 & \dots & 0 \\ & & & \\ 0 & \dots & 0 & 1 \end{pmatrix}$. $\mathbf{V}$ and $\mathbf{I}$ are the nodal voltage vector and branch current vector, respectively. $\mathbf{I_V}$ is the vector of the current flowing through the two voltage sources V_1 and V_2.

If the entire spiral is modeled by an equivalent π circuit as shown in Fig. 4.3, the inductance and the quality factor are given by

$$L = -\frac{1}{2\pi f} Im(\frac{1}{Y_{12}}) \tag{4.3}$$

and

$$Q = -\frac{Im(Y_{11})}{Re(Y_{11})} \tag{4.4}$$

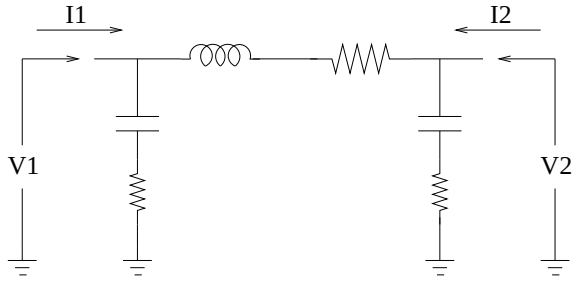


Figure 4.3: Equivalent π model of the spiral.

The two port Y parameters can be extracted from (4.2). Setting $V_1 = 1$ and $V_2 = 0$, Y_{11} is given by $Y_{11} = (\frac{I_{V_1}}{V_1})_{V_2=0} = (I_{V_1})_{V_1=1,V_2=0}$, while setting $V_1 = 0$ and $V_2 = 1$, Y_{12}

is given by $Y_{12} = (\frac{I_{V_1}}{V_2})_{V_1=0} = (I_{V_1})_{V_1=0,V_2=1}$. The total computational cost is one LU factorization of the coefficient matrix in (4.2) and two forward/backward substitutions. The method that is used to build the Y^C and Z^L matrices is similar to that in [NM98] and we briefly recapitulate it here. The matrix elements of Y^C are given by

$$Y_{ij}^C = \begin{cases} -\frac{1}{Z_{ij}^C} & if \;\; i \neq j \\ \sum_{k=0}^{n+1} \frac{1}{Z_{ik}^C} & if \;\; i = j \end{cases} \tag{4.5}$$

where Z_{ij}^C is the total impedance between node i and node j excluding that of the series branches, and $n+1$ is the total number of nodes excluding the ground node. All the distributed parasitic capacitances are calculated using the simple parallel plate capacitor equation. More accurate expressions can be found in [WLM00] or use a capacitance extraction package such as FastCap [NW91]. To construct the Z^L matrix, we divide the relative position of metal segments i and j into three categories.

Category 1: Segments i and j are perpendicular to each other.

$$Z_{ij}^L = 0 \tag{4.6}$$

because there is no mutual inductance between perpendicular segments.

Category 2: Segments i and j are parallel to each other but are on the opposite sides of the spiral.

$$Z_{ij}^{L} = -j\omega M_{ij} \tag{4.7}$$

where M_{ij} is the mutual inductance between segments i and j. For the two wire segments shown in Fig. 4.4, the mutual inductance is given by

$$M_{ij} = M(\frac{l_1 + l_2}{2}, d) - M(\frac{l_2 - l_1}{2}, d) \tag{4.8}$$

where

$$\begin{aligned} M(l,d) = \frac{\mu l}{2\pi}[ln(\frac{\sqrt{l^2+d^2}+l}{d}) - \frac{\sqrt{l^2+d^2}}{l} + \frac{d}{l}] \\ + \frac{\mu l}{2\pi}[\frac{w^2 l}{12d^2\sqrt{l^2+d^2}} - \frac{t^2 l}{12d^2(d+\sqrt{l^2+d^2})}] \end{aligned} \tag{4.9}$$

with w and t representing the width and thickness of each metal segment, respectively [Moh].

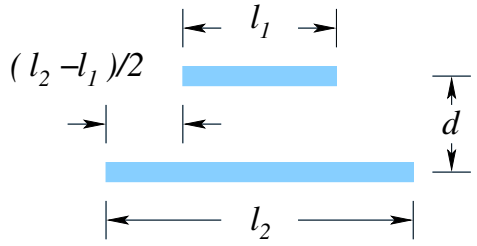


Figure 4.4: Configuration of the two metal segments for mutual inductance computation.

Category 3: Segments i and j are parallel to each other and are on the same side of the spiral.

Generally, there is no simple closed form formula available in this case for either the mutual inductance between segments i and j or the self inductance of a segment because of the presence of skin effect and proximity effect. These two high frequency effects result in a nonuniform current distribution across the cross section of a metal segment which invalidates all the formulas based on the assumption that the current distribution is uniform. As pointed out in [KI01], the proximity effect may become significant at frequencies as low as a few hundred MHz for integrated spiral inductors, which means that it should be considered in all simulations of radio frequency devices.

To handle the nonuniform current distributions in metal segments due to skin and proximity effects, each metal segment is first divided into filaments along the longitudinal direction and it is assumed that the current distribution in each filament is uniform [NM98] [WWMS79]. Assume we have n parallel metal segments on each side of the spiral, the i^{th} segment is divided into N_i filaments, and let(i, j) represents the j^{th} filament of segment i, we have

$$V_{ij} = \sum_{k=1}^{n} \sum_{m=1}^{N_k} (r_{ij}\delta_{ki}\delta_{jm} + j\omega M_{ij,km}) I_{km} = \sum_{k=1}^{n} \sum_{m=1}^{N_k} Z_{ij,km} I_{km} \tag{4.10}$$

where V_{ij} is the voltage across filament (i, j), r_{ij} is the DC resistance of filament (i, j), and $M_{ij,km}$ is the mutual inductance between filament (i, j) and (k, m), which can be computed using (4.8) and (4.9). If $(i, j) = (k, m)$ however, $M_{ij,km}$ becomes the self inductance of filament (i, j) and should be computed using

$$L = \frac{\mu l}{2\pi}[ln(\frac{2l}{w+t}) + 0.5 + (\frac{\sqrt{w^2+t^2+0.46tw}}{3l}) - \frac{w^2+t^2}{24l^2}] \tag{4.11}$$

where l, w, and t are the length, width, and thickness of the filament, respectively [Moh]. Equation (4.10) can be inverted to obtain

$$I_{ij} = \sum_{k=1}^{n}\sum_{m=1}^{N_k} Y_{ij,km}V_{km} \tag{4.12}$$

Since V_{km} remains the same for the filaments in the same segment, the total current in the i^{th} segment can be computed using

$$I_i = \sum_{j=1}^{N_i} I_{ij} = \sum_{j=1}^{N_i}\sum_{k=1}^{n} V_k \sum_{m=1}^{N_k} Y_{ij,km} = \sum_{k=1}^{n}(\sum_{j=1}^{N_i}\sum_{m=1}^{N_k} Y_{ij,km})V_k = \sum_{k=1}^{n} Y_{ik}V_k \tag{4.13}$$

Equation (4.13) can be inverted again to obtain

$$V_i = \sum_{k=1}^{n} Z_{ik}I_k \tag{4.14}$$

The Z_{ik}'s should be used to construct the portions of the Z^L matrix corresponding to the parallel segments on the same side of the spiral.

4.3.2 Sensitivity computation

To apply the SQP method, we must be able to compute the sensitivity of the inductance value L and quality factor Q with respect to each optimization variable. Let p represent an optimization variable, then

$$\frac{\partial L}{\partial p} = \frac{1}{2\pi f} Im(\frac{1}{Y_{12}^2}\frac{\partial Y_{12}}{\partial p}) \tag{4.15}$$

and

$$\frac{\partial Q}{\partial p} = -\frac{Im(\frac{\partial Y_{11}}{\partial p})Re(Y_{11}) - Im(Y_{11})Re(\frac{\partial Y_{11}}{\partial p})}{[Re(Y_{11})]^2} \tag{4.16}$$

From (4.15) and (4.16), we can see that to obtain the sensitivity, we must compute the partial derivative of the Y parameters with respect to the optimization variables. One way to accomplish this is to use the finite difference approximation which requires one extra simulation for the sensitivity with respect to each optimization variable. Since simulation is an expensive process, we chose to use the adjoint method to compute the sensitivity which does not require any extra simulation at all.

Since the Y parameters can be obtained from $\mathbf{I_V}$ in (4.2) by setting V_1 and V_2 to appropriate values, the question comes down to the computation of the sensitivity of

$\mathbf{I_V}$ with respect to the optimization variables. Note that the value of each optimization variable only affects the coefficient matrix in (4.2) but not the right hand side, we consider the following generalized system of linear equations.

$$A\mathbf{x} = \mathbf{b} \tag{4.17}$$

where A is the coefficient matrix with p as the parameter and $\mathbf{b}$ is a vector independent of p, then we have

$$\frac{\partial x_i}{\partial p} = \sum_{ij} \frac{\partial x_i}{\partial A_{ij}} \frac{\partial A_{ij}}{\partial p} \tag{4.18}$$

$\frac{\partial A_{ij}}{\partial p}$ can be obtained directly from the physical model used to construct the coefficient matrix, and $\frac{\partial x_i}{\partial A_{ij}}$ can be computed using the adjoint method [PRV95], which gives

$$\frac{\partial x_i}{\partial A_{kl}} = -\xi_{ik} x_l \tag{4.19}$$

where ξ_{ik} is the k^{th} element of vector ξ_i, and ξ_i is the solution to the equation

$$A^T \xi_{\mathbf{i}} = \mathbf{e_i} \tag{4.20}$$

Here $\mathbf{e_i}$ is the i^{th} column of the identity matrix. Note that (4.20) is extremely inexpensive to solve because after solving (4.17) using LU factorization, the LU factors of A^T

can be obtained automatically, i.e., $A^T = U^T L^T$ if $A = LU$, and the cost of solving (4.20) is only one forward/backward substitution.

4.4 SQP Algorithm

Sequential quadratic programming is a versatile method for solving general non-linear constrained optimization problems of the form

$$\begin{aligned} &minimize && f(\mathbf{x}) \\ &subject\ to && \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ & && \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \end{aligned} \tag{4.21}$$

where $f : R^n \rightarrow R, \mathbf{h} : R^n \rightarrow R^m, and\ \mathbf{g} : R^n \rightarrow R^p$. Our formulation of the inductor optimization problem fits perfectly into the framework of (4.21). The Lagrangian of the problem is defined as

$$L(\mathbf{x}, \mathbf{u}, \mathbf{v}) = f(\mathbf{x}) + \mathbf{u^t h}(\mathbf{x}) + \mathbf{v^t g}(\mathbf{x}) \tag{4.22}$$

In each iteration of the algorithm, a quadratic subproblem

$$\begin{aligned} minimize \quad & \nabla f(\mathbf{x^k})^t \mathbf{d_x} + \tfrac{1}{2}\mathbf{d_x^t} B_k \mathbf{d_x} \\ subject\ to \quad & \nabla \mathbf{h}(\mathbf{x^k})^t \mathbf{d_x} + \mathbf{h}(\mathbf{x^k}) = \mathbf{0} \\ & \nabla \mathbf{g}(\mathbf{x^k})^t \mathbf{d_x} + \mathbf{g}(\mathbf{x^k}) \leq \mathbf{0} \end{aligned} \tag{4.23}$$

is formed where B_k is the approximation to the Hessian matrix H_L of the Lagrangian with respect to **x**. The gradients ∇f, $\nabla \mathbf{h}$, and $\nabla \mathbf{g}$ are computed using the sensitivity analysis discussed in section 4.3.2. The quadratic subproblem can be solved efficiently using any well known method such as the active set method. If $\mathbf{d_x^k}$ is the solution to (4.23) in iteration **k**, the solution to the original problem (4.21) can be updated using

$$\mathbf{x^{k+1}} = \mathbf{x^k} + \alpha \mathbf{d_x^k} \tag{4.24}$$

where α is the step length parameter. As pointed out in [BT95], there are different ways that the Hessian matrix of the Lagrangian can be approximated, one of the most popular ones is due to Broyden, Fletcher, Goldfarb, and Shanno, where B_k is initially set to the identity matrix and updated using the formula

$$B_{k+1} = B_k + \frac{\mathbf{y_k} \mathbf{y_k^t}}{\mathbf{s_k^t} \mathbf{y_k}} - \frac{B_k \mathbf{s_k} \mathbf{s_k^t} B_k}{\mathbf{s_k^t} B_k \mathbf{s_k}} \tag{4.25}$$

where

$$\mathbf{s_k} = \mathbf{x^{k+1}} - \mathbf{x^k} \tag{4.26}$$

$$\mathbf{y_k} = \nabla_{\mathbf{x}} L(\mathbf{x^{k+1}}, \mathbf{u^{k+1}}, \mathbf{v^{k+1}}) - \nabla_{\mathbf{x}} L(\mathbf{x^k}, \mathbf{u^k}, \mathbf{v^k}) \tag{4.27}$$

To ensure the positive definiteness of B_k, the $\mathbf{y_k}$ vector is reset to

$$\mathbf{y_k} \leftarrow \theta_k \mathbf{y_k} + (1 - \theta_k) B_k \mathbf{s_k} \tag{4.28}$$

if $\mathbf{y_k^t s_k}$ is not sufficiently positive. Here $\theta_k \in [1, 0)$ is the number closest to 1 such that $\mathbf{y_k^t s_k} \geq \sigma \mathbf{s_k^t} B_k \mathbf{s_k}$ for some $\sigma \in (0, 1)$ [SQP].

The iteration will continue until the Karush-Kuhn-Tucker condition is satisfied and the $\mathbf{d_x}$ vector becomes zero. We have only provided the most basic background of the SQP algorithm here and will not elaborate on this topic any further. Interested readers are referred to [BT95] and [NW99].

4.5 Experimental Results

We used the CFSQP package [LZT] as the SQP optimization engine to optimize the quality factor of square spiral inductors. The inductors are assumed to be fabricated using $1\mu m$ thick metal with sheet resistance of $20m\Omega/\Box$ and rest $5\mu m$ above the

	Case 1		*Case 2*		*Case 3*		*Case 4*	
Lexp	$4.5nH$		$6nH$		$12nH$		$15nH$	
Freq	$2GHz$		$2GHz$		$2GHz$		$1GHz$	
D bound	$150\mu m/250\mu m$		$200\mu m/400\mu m$		$600\mu m/800\mu m$		$600\mu m/800\mu m$	
w bound	$2\mu m/10\mu m$		$2\mu m/20\mu m$		$2\mu m/20\mu m$		$2\mu m/20\mu m$	
s bound	$2\mu m/10\mu m$		$2\mu m/10\mu m$		$2\mu m/20\mu m$		$2\mu m/20\mu m$	
Num. turns	3		3		3		3	
Opt. Result	Enum	SQP	Enum	SQP	Enum	SQP	Enum	SQP
D	$250\mu m$	$250\mu m$	$400\mu m$	$400\mu m$	$600\mu m$	$600\mu m$	$780\mu m$	$778.42\mu m$
w	$8\mu m$	$8.18\mu m$	$16\mu m$	$17.55\mu m$	$10\mu m$	$10.92\mu m$	$20\mu m$	$20\mu m$
s	$2\mu m$	$2\mu m$	$4\mu m$	$2\mu m$	$4\mu m$	$3.53\mu m$	$2\mu m$	$2\mu m$
L	$4.32nH$	$4.28nH$	$5.77nH$	$5.70nH$	$11.73nH$	$11.40nH$	$14.28nH$	$14.25nH$
Q	7.37	7.46	10.53	10.78	6.92	6.96	7.37	7.38
Runtime	$170s$	$11s$	$2286s$	$41s$	$4579s$	$28s$	$2179s$	$22s$

Table 4.1: Comparison of the optimization results using SQP and enumeration.

substrate. The lower and upper bound of the optimization variables are provided to the optimizer as the input and the tolerance of the allowed inductance deviation is set to 5%. Table 4.1 compares the optimization result of four sample inductors using SQP and enumeration. L_{exp} and $Freq$ are the expected inductance and operating frequency of the device. $D\ bound$, $w\ bound$, and $s\ bound$ specify the lower and upper bounds of the outer length, trace width, and the space between metal traces, respectively. For the SQP method, the initial solution provided to the optimizer is chosen in such a way that each variable takes the average value of its lower and upper bounds. If this solution is infeasible, the SQP algorithm will first find a feasible solution automatically and then start optimization [LZT]. For the enumeration method, D, w, and s are incremented by $10\mu m$, $2\mu m$, and $2\mu m$ each time, respectively.

From the result of comparison, we can see that the quality of each optimized design obtained from SQP is as good as that from enumeration while the runtime of the former

is at least one order of magnitude smaller than that of the latter. We point out that the grid we used for the enumeration method is rather coarse, and if a finer grid is used, the advantage of SQP over enumeration will become even more significant. In addition, we emphasize that the objective of this work was to demonstrate the speedup that can be obtained by using the SQP instead of the enumeration algorithm. The code for extracting the inductance and quality factor is not optimized for speed and accuracy. However, even if a different program is used to implement the extraction engine, we can still expect SQP to have similar speedup ratio compared with enumeration.

A subtle point concerning the SQP algorithm versus enumeration is that the optimization problem we are solving has not been proven to be a convex program. As a result, no mathematical programming algorithm can guarantee the finding of the global optimum, i.e., there is a possibility that the algorithm will stop at a local optimum. However, extensive experiments have shown that most of the time, the global optimum can be found in a single run of the SQP algorithm, and in case the algorithm does stop at a local optimum in one run, we can still obtain the global optimum by running the algorithm at most a few times starting from different initial points. This will preserve the validity of our claim that the SQP algorithm is superior to enumeration in terms of runtime considering the orders of magnitude speedup each single run of SQP can achieve.

4.6 Summary

In this work, we used sequential quadratic programming to optimize the quality factor of integrated spiral inductors. Experimental results have demonstrated that SQP can achieve at least an order of magnitude speedup compared with enumeration while maintaining the same quality of the optimized design. Besides its high local convergence rate, another advantage of the SQP method is that it makes no assumption about the formality of either the objective or the constraint functions, which makes it quite compatible with physical models derived from first principles.

Chapter 5

Conclusion

The continuing progress in semiconductor manufacturing technologies has significantly improved the performance of today's VLSI circuits. However, it has also brought about many challenges to CAD tool developers, who play a vital role in improving the quality and reducing the time of circuit design. Two of the most prominent challenges include properly handling the effects that have become important in circuits with feature size in the nanometer range and developing high efficiency CAD algorithms to assist the design of ICs containing hundreds of millions of transistors.

In this thesis, we focused on the development of several high efficiency analysis and optimization algorithms for the computer aided design of electronic circuits. In the first part of the thesis, several Green function-based thermal simulation algorithms were presented, and they can be used, respectively, to perform full chip temperature

profiling, localized thermal analysis, and thermal simulations where the accuracy requirement differs from place to place over the same chip. The accuracy and efficiency of the algorithms were demonstrated through comparisons with the results from a commercial computational fluid dynamic software for thermal analysis.

The second part of the thesis is concerned with reducing the power waste in the design of stacked-Vdd circuits, which may hold the key to resolving the I/O pin limitation problem in VLSI circuits with high performance and high integration density. A graph partition-based algorithm was developed for efficiently assigning modules to different Vdd domains so that the maximum current balance is achieved and the minimum amount of power is wasted by the circuit. The effectiveness of this approach was verified through comparisons with the module assignment obtained using a bin-packing technique and the results from a set of random experiments.

As stated previously, the main focus of the second part of the thesis is on the assignment of modules to two different Vdd domains when the floorplan is given. Although we have also made some modifications to the floorplanner Parquet so as to effectively distribute the voltage regulators during the floorplanning process, the resulting regulator distribution may not be optimal. As a future work, it would be interesting to develop a strategy that can incrementally adjust the location of each regulator after the initial floorplan is obtained so that the performance of the final design can be further improved.

Finally, in the last part of the thesis, we used the SQP technique to optimize the quality factor of on-chip spiral inductors, which are important components in integrated RF circuits. Experimental results show that high quality inductors can be obtained using this approach within a very reasonable amount of time.

Appendix A

Derivation of the Green Function

In this appendix, we present the derivation of the Green function for the rectangular-shaped multilayered chip structure. From the definition of the Green function in Section 2.2.2, we know that, using the separation of variables, the Green function $G(\mathbf{r}, \mathbf{r}')$ can be written as

$$G(\mathbf{r}, \mathbf{r}') = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \cos\left(\frac{m\pi x}{a}\right) \cos\left(\frac{n\pi y}{b}\right) Z_{mn}(z) \tag{A.1}$$

for a particular $\mathbf{r}'$. The x and y dependencies in (A.1) ensure that the boundary condition (2.17) is satisfied. The Poisson's equation (2.16) now becomes

$$\sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \left(\frac{d^2 Z_{mn}(z)}{dz^2} - \gamma_{mn}^2 Z_{mn}(z)\right) \cos\left(\frac{m\pi x}{a}\right) \cos\left(\frac{n\pi y}{b}\right) = -\frac{\delta(\mathbf{r} - \mathbf{r}')}{k_{l(\mathbf{r})}} \tag{A.2}$$

where $\gamma_{mn} = \sqrt{\left(\frac{m\pi}{a}\right)^2 + \left(\frac{n\pi}{b}\right)^2}$. Multiplying both sides of equation (A.2) by $\cos\left(\frac{m\pi x}{a}\right)$ $\cos\left(\frac{n\pi y}{b}\right)$ and integrating over x and y, we obtain

$$\frac{ab}{s}\left(\frac{d^2 Z_{mn}(z)}{dz^2} - \gamma_{mn}^2 Z_{mn}(z)\right) = -\frac{\delta(z-z')}{k_{l(\mathbf{r})}}\cos\left(\frac{m\pi x'}{a}\right)\cos\left(\frac{n\pi y'}{b}\right) \tag{A.3}$$

where

$$s = \begin{cases} 1 & \text{if } m = n = 0 \\ 2 & \text{if } m = 0, n \neq 0 \text{ or } m \neq 0, n = 0 \\ 4 & \text{if } m \neq 0, n \neq 0 \end{cases} \tag{A.4}$$

Let $Z_{mn}(z) = Z'_{mn}(z)\cos\left(\frac{m\pi x'}{a}\right)\cos\left(\frac{n\pi y'}{b}\right)$. Then $Z'_{mn}(z)$ satisfies the equation

$$\frac{d^2 Z'_{mn}(z)}{dz^2} - \gamma_{mn}^2 Z'_{mn}(z) = -\frac{s\delta(z-z')}{abk_{l(\mathbf{r})}} \tag{A.5}$$

Let l_s be the layer in which the source point $\mathbf{r}' = (x', y', z')$ resides. Then for layer i other than l_s, the right hand side of equation (A.5) vanishes. Hence

$$Z'^{\,i}_{mn}(z) = \begin{cases} \alpha^i_{mn} z + \beta^i_{mn} & \text{if } m = n = 0 \\ \alpha^i_{mn} e^{\gamma_{mn} z} + \beta^i_{mn} e^{-\gamma_{mn} z} & \text{otherwise} \end{cases} \tag{A.6}$$

where $Z'^{\,i}_{mn}(z)$ represents $Z'_{mn}(z)$ in layer i and $i \neq l_s$. For layer l_s, we need to consider the field point $\mathbf{r}$ above and below the source point $\mathbf{r}'$ separately. Let $Z'^{\,l_s^U}_{mn}(z)$ and

${Z'}_{mn}^{l_s^L}(z)$ represent $Z'_{mn}(z)$ above and below the source point in layer l_s, respectively. Then ${Z'}_{mn}^{l_s^U}(z)$ can be written as

$$ {Z'}_{mn}^{l_s^U}(z) = \begin{cases} \alpha_{mn}^{l_s^U} z + \beta_{mn}^{l_s^U} & \text{if } m = n = 0 \\ \alpha_{mn}^{l_s^U} e^{\gamma_{mn} z} + \beta_{mn}^{l_s^U} e^{-\gamma_{mn} z} & \text{otherwise} \end{cases} \tag{A.7} $$

A similar expression can be written for ${Z'}_{mn}^{l_s^L}(z)$ by replacing all instances of U in (A.7) by L. We require that $Z'_{mn}(z)$ be continuous at $z = z'$, therefore

$$ {Z'}_{mn}^{l_s^U}(z)|_{z=z'} = {Z'}_{mn}^{l_s^L}(z)|_{z=z'} \tag{A.8} $$

Integrating equation (A.5) from $z' - \epsilon$ to $z' + \epsilon$ where ϵ is an infinitesimally small quantity, we obtain

$$ \left.\frac{d{Z'}_{mn}^{l_s^U}(z)}{dz}\right|_{z=z'} - \left.\frac{d{Z'}_{mn}^{l_s^L}(z)}{dz}\right|_{z=z'} = -\frac{s}{abk_{l_s}} \tag{A.9} $$

where k_{l_s} is the thermal conductivity of layer l_s. In addition, we also require that

$Z'_{mn}(z)$ satisfies the following set of boundary conditions

$$\left.\frac{\partial Z'^{\,1}_{mn}(z)}{\partial z}\right|_{z=0} = 0 \tag{A.10}$$

$$k_N \left.\frac{\partial Z'^{\,N}_{mn}(z)}{\partial z}\right|_{z=-d_N} = h Z'^{\,N}_{mn}(z)|_{z=-d_N} \tag{A.11}$$

$$Z'^{\,i}_{mn}(z)|_{z=-d_i} = Z'^{\,i+1}_{mn}(z)|_{z=-d_i} \tag{A.12}$$

$$k_i \left.\frac{\partial Z'^{\,i}_{mn}(z)}{\partial z}\right|_{z=-d_i} = k_{i+1} \left.\frac{\partial Z'^{\,i+1}_{mn}(z)}{\partial z}\right|_{z=-d_i} \tag{A.13}$$

such that $G(\mathbf{r}, \mathbf{r}')$ satisfies the boundary conditions (2.18)-(2.21). Equations (A.8)-(A.13) are sufficient to determine the sets of coefficients α_{mn} and β_{mn} for all the layers. We consider two different cases.

1. $m = n = 0$

   Equations (A.8)-(A.13) now become

$$\alpha^{l_s^U}_{00} z' + \beta^{l_s^U}_{00} = \alpha^{l_s^L}_{00} z' + \beta^{l_s^L}_{00} \tag{A.14}$$

$$\alpha^{l_s^U}_{00} - \alpha^{l_s^L}_{00} = -\frac{1}{abk_{l_s}} \tag{A.15}$$

$$\alpha^1_{00} = 0 \tag{A.16}$$

$$k_N \alpha^N_{00} = h(-\alpha^N_{00} d_N + \beta^N_{00}) \tag{A.17}$$

$$-\alpha^i_{00} d_i + \beta^i_{00} = -\alpha^{i+1}_{00} d_i + \beta^{i+1}_{00} \tag{A.18}$$

$$k_i \alpha^i_{00} = k_{i+1} \alpha^{i+1}_{00} \tag{A.19}$$

where $i = 1, 2, \ldots, N-1$. Equations (A.14)-(A.19) can be solved to obtain

$$\alpha_{00}^{1} = \alpha_{00}^{2} = \ldots = \alpha_{00}^{l_s - 1} = \alpha_{00}^{l_s^U} = 0 \tag{A.20}$$

$$\alpha_{00}^{l_s^L} = \frac{1}{abk_{l_s}} \quad \text{and} \quad \alpha_{00}^{i} = \frac{1}{abk_i} \text{ where } i > l_s \tag{A.21}$$

$$\beta_{00}^{N} = \frac{1}{abk_N}\left(\frac{k_N}{h} + d_N\right) \tag{A.22}$$

$$\beta_{00}^{i} = \beta_{00}^{i+1} + \left(\frac{1}{abk_i} - \frac{1}{abk_{i+1}}\right) d_i$$

$$\text{where } i = l_s^L, l_s + 1, \ldots, N-1 \tag{A.23}$$

$$\beta_{00}^{1} = \beta_{00}^{2} = \ldots = \beta_{00}^{l_s - 1} = \beta_{00}^{l_s^U} = \beta_{00}^{l_s^L} + \frac{z'}{abk_{l_s}} \tag{A.24}$$

2. $m \neq 0$ or $n \neq 0$

Equations (A.8)-(A.13) now become

$$\alpha_{mn}^{l_s^U} e^{\gamma_{mn} z'} + \beta_{mn}^{l_s^U} e^{-\gamma_{mn} z'} = \alpha_{mn}^{l_s^L} e^{\gamma_{mn} z'} + \beta_{mn}^{l_s^L} e^{-\gamma_{mn} z'} \tag{A.25}$$

$$(\alpha_{mn}^{l_s^U} - \alpha_{mn}^{l_s^L}) e^{\gamma_{mn} z'} - (\beta_{mn}^{l_s^U} - \beta_{mn}^{l_s^L}) e^{-\gamma_{mn} z'} = -\frac{s}{abk_{l_s}\gamma_{mn}} \tag{A.26}$$

$$\alpha_{mn}^{1} = \beta_{mn}^{1} \tag{A.27}$$

$$k_N \gamma_{mn} (\alpha_{mn}^{N} e^{-\gamma_{mn} d_N} - \beta_{mn}^{N} e^{\gamma_{mn} d_N}) = h(\alpha_{mn}^{N} e^{-\gamma_{mn} d_N} + \beta_{mn}^{N} e^{\gamma_{mn} d_N}) \tag{A.28}$$

$$\alpha_{mn}^{i} e^{-\gamma_{mn} d_i} + \beta_{mn}^{i} e^{\gamma_{mn} d_i} = \alpha_{mn}^{i+1} e^{-\gamma_{mn} d_i} + \beta_{mn}^{i+1} e^{\gamma_{mn} d_i} \tag{A.29}$$

$$k_i (\alpha_{mn}^{i} e^{-\gamma_{mn} d_i} - \beta_{mn}^{i} e^{\gamma_{mn} d_i}) = k_{i+1} (\alpha_{mn}^{i+1} e^{-\gamma_{mn} d_i} - \beta_{mn}^{i+1} e^{\gamma_{mn} d_i}) \tag{A.30}$$

From equations (A.29) and (A.30), we obtain

$$\begin{pmatrix} \alpha_{mn}^{i+1} \\ \beta_{mn}^{i+1} \end{pmatrix} = \frac{1}{2k_{i+1}} \begin{pmatrix} k_{i+1}+k_i & (k_{i+1}-k_i)e^{2\gamma_{mn}d_i} \\ (k_{i+1}-k_i)e^{-2\gamma_{mn}d_i} & k_{i+1}+k_i \end{pmatrix} \begin{pmatrix} \alpha_{mn}^{i} \\ \beta_{mn}^{i} \end{pmatrix} \tag{A.31}$$

and

$$\begin{pmatrix} \alpha_{mn}^{i} \\ \beta_{mn}^{i} \end{pmatrix} = \frac{1}{2k_{i}} \begin{pmatrix} k_{i}+k_{i+1} & (k_{i}-k_{i+1})e^{2\gamma_{mn}d_i} \\ (k_{i}-k_{i+1})e^{-2\gamma_{mn}d_i} & k_{i}+k_{i+1} \end{pmatrix} \begin{pmatrix} \alpha_{mn}^{i+1} \\ \beta_{mn}^{i+1} \end{pmatrix} \tag{A.32}$$

Equations (A.31) and (A.32) can be used to obtain α_{mn}^{i+1} and β_{mn}^{i+1} from α_{mn}^{i} and β_{mn}^{i}, or obtain α_{mn}^{i} and β_{mn}^{i} from α_{mn}^{i+1} and β_{mn}^{i+1}. From equation (A.27) and (A.31), we know that all the α_{mn}^{i} and β_{mn}^{i} for $i = 1, 2, \ldots, l_s - 1, l_s^U$ are proportional to $\alpha_{mn}^{1} = \beta_{mn}^{1}$. Let

$$\alpha_{mn}^{i} = \alpha_{mn}^{i*}\alpha_{mn}^{1} \tag{A.33}$$

$$\beta_{mn}^{i} = \beta_{mn}^{i*}\alpha_{mn}^{1} \tag{A.34}$$

where $i = 1, 2, \ldots, l_s - 1, l_s^U$. Then apparently $\alpha_{mn}^{1*} = \beta_{mn}^{1*} = 1$, and from equation (A.31), we can obtain the numerical values of α_{mn}^{i*} and β_{mn}^{i*} for $i =$

$1, 2, \ldots, l_s - 1, l_s^U$. From equation (A.28), we obtain

$$\beta_{mn}^N = \frac{k_N\gamma_{mn} - h}{k_N\gamma_{mn} + h} e^{-2\gamma_{mn}d_N}\alpha_{mn}^N \tag{A.35}$$

and from equation (A.32), we know that all the α_{mn}^i and β_{mn}^i for $i = N, N - 1, \ldots, l_s + 1, l_s^L$ are proportional to α_{mn}^N. Let

$$\alpha_{mn}^i = \alpha_{mn}^{i*}\alpha_{mn}^N \tag{A.36}$$

$$\beta_{mn}^i = \beta_{mn}^{i*}\alpha_{mn}^N \tag{A.37}$$

where $N, N - 1, \ldots, l_s + 1, l_s^L$. Then apparently $\alpha_{mn}^{N*} = 1$, and from equations (A.35) and (A.32), we can obtain the numerical values of α_{mn}^{i*} and β_{mn}^{i*} for $i = N, N - 1, \ldots, l_s + 1, l_s^L$. Equations (A.25) and (A.26) now become

$$(\alpha_{mn}^{l_s^U *}e^{\gamma_{mn}z'} + \beta_{mn}^{l_s^U *}e^{-\gamma_{mn}z'})\alpha_{mn}^1 = (\alpha_{mn}^{l_s^L *}e^{\gamma_{mn}z'} + \beta_{mn}^{l_s^L *}e^{-\gamma_{mn}z'})\alpha_{mn}^N \tag{A.38}$$

$$(\alpha_{mn}^{l_s^U *}e^{\gamma_{mn}z'} - \beta_{mn}^{l_s^U *}e^{-\gamma_{mn}z'})\alpha_{mn}^1 - (\alpha_{mn}^{l_s^L *}e^{\gamma_{mn}z'} - \beta_{mn}^{l_s^L *}e^{-\gamma_{mn}z'})\alpha_{mn}^N = -\frac{s}{abk_{l_s}\gamma_{mn}} \tag{A.39}$$

The numerical values of α_{mn}^1 and α_{mn}^N can be obtained by solving equations (A.38) and (A.39). Then from equations (A.33), (A.34), (A.36), and (A.37), we can obtain the numerical values of all α_{mn}^i and β_{mn}^i.

Hence, we have obtained the analytical solution of the Green function $G(\mathbf{r}, \mathbf{r}')$ in the form

$$G(\mathbf{r}, \mathbf{r}') = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \cos\left(\frac{m\pi x}{a}\right) \cos\left(\frac{n\pi y}{b}\right) \cos\left(\frac{m\pi x'}{a}\right) \cos\left(\frac{n\pi y'}{b}\right) Z'_{mn}(z, z') \tag{A.40}$$

where we have written the $Z'_{mn}(z)$ as $Z'_{mn}(z, z')$ in the above expression since the coefficients α^i_{mn} and β^i_{mn} both depend on z'.

Bibliography

[AM03] S. N. Adya and I. L. Markov. Fixed-outline floorplanning: Enabling hierarchical design. *IEEE Transactions on VLSI Systems*, 11(6):1120–1135, Dec. 2003.

[BT95] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, pages 1–51, 1995.

[CAG93] J. Y.-C. Chang, A. A. Abidi, and M. Gaitan. Large suspended inductors on silicon and their use in a 2-μm CMOS RF amplifier. *IEEE Electron Device Letters*, 14(5):246–248, May 1993.

[CCS98] J. P. Costa, M. Chou, and L. M. Silveira. Precorrected-DCT techniques for modeling and simulation of substrate coupling in mixed-signal IC's. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 358–362, Jun. 1998.

[CCS99] J. P. Costa, M. Chou, and L. M. Silveira. Efficient techniques for accurate modeling and simulation of substrate coupling in mixed-signal IC's. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(5):597–607, May 1999.

[Chi04] E. Chiprout. Fast flip-chip power grid analysis via locality and grid-shells. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 485–488, Nov. 2004.

[CK99] Y. K. Cheng and S. M. Kang. An efficient method for hot-spot identification in ULSI circuits. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 124–127, Nov. 1999.

[CLRK00] D. Chen, E. Li, E. Rosenbaum, and S. M. Kang. Interconnect thermal modeling for accurate simulation of circuit timing and reliability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(2):197–205, Feb. 2000.

[CRT+98] Y. K. Cheng, P. Raha, C. C. Teng, E. Rosenbaum, and S. M. Kang. ILLIADS-T: An electrothermal timing simulator for temperature-sensitive reliability diagnosis of CMOS VLSI chips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(8):668–681, Aug. 1998.

[Gha] R. Gharpurey. Modeling and analysis of substrate coupling in integrated circuits. Ph. D. Thesis, University of California - Berkeley, Berkeley, CA, 1995.

[GK05] J. Gu and C. H. Kim. Multi-story power delivery for supply noise reduction and low voltage operation. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 192–197, Aug. 2005.

[GM96] R. Gharpurey and R. G. Meyer. Modeling and analysis of substrate coupling in integrated circuits. *IEEE Journal of Solid-State Circuits*, 31(3):344–353, Mar. 1996.

[GS03] B. Goplen and S. S. Sapatnekar. Efficient thermal placement of standard cells in 3D ICs using a force directed approach. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 86–89, Nov. 2003.

[HBZ+03] H. Hu, D. T. Blaauw, V. Zolotov, K. Gala, M. Zhao, R. Panda, and S. S. Sapatnekar. Fast on-chip inductance simulation using a precorrected-FFT method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(1):49–61, Jan. 2003.

[HMBL99] M. D. M. Hershenson, S. S. Mohan, S. P. Boyd, and T. H. Lee. Optimization of inductor circuits via geometric programming. In *Proceedings of the IEEE/ACM Design Automation Conference*, pages 994–998, Jun. 1999.

[HS90] A. Haji-Sheikh. Peak temperature in high-power chips. *IEEE Transactions on Electron Devices*, 37(4):902–907, Apr. 1990.

[KAKS99] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Transactions on VLSI Systems*, 7(1):69–79, Mar. 1999.

[KI01] W. B. Kuhn and N. M. Ibrahim. Analysis of current crowding effects in multiturn spiral inductors. *IEEE Transactions on Microwave Theory and Techniques*, 49(1):31–37, Jan. 2001.

[Kok74] A. G. Kokkas. Thermal analysis of multi-layer structures. *IEEE Transactions on Electron Devices*, 21(11):674–681, Nov. 1974.

[LHL] W. Liao, L. He, and K. Lepak. Temperature-aware performance and power modeling. Technical Report UCLA Engr. 04-250, University of California - Los Angeles, Los Angeles, CA, 2004.

[LPAC04] P. Li, L. T. Pileggi, M. Asheghi, and R. Chandra. Efficient full-chip thermal modeling and analysis. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 319–326, Nov. 2004.

[LZT] C. Lawrence, J. L. Zhou, and A. L. Tits. User's guide for CFSQP version 2.5. http://64.238.116.66/aemdesign/download-cfsqp/cfsqp-manual.pdf.

[Moh] S. S. Mohan. The design, modeling, and optimization of on-chip inductor and transformer circuits. Ph. D. Thesis, Stanford University, Stanford, CA, 1999.

[NGM98] A. M. Niknejad, R. Gharpurey, and R. G. Meyer. Numerically stable Green function for modeling and analysis of substrate coupling in integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(4):305–315, Apr. 1998.

[Nik] A. M. Niknejad. Analysis, simulation, and applications of passive devices on conductive substrate. Ph. D. Thesis, University of California - Berkeley, Berkeley, CA, 2000.

[NLS06] V. Nookala, D. Lilja, and S. S. Sapatnekar. Temperature-aware floorplanning of microarchitecture blocks with IPC-power dependence modeling

and transient analysis. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 298–303, Oct. 2006.

[NM98] A. M. Niknejad and R. G. Meyer. Analysis, design, and optimization of spiral inductors and transformers for Si RF IC's. *IEEE Journal of Solid-State Circuits*, 33(10):1470–1481, Oct. 1998.

[NW91] K. Nabors and J. White. FastCap: A multipole accelerated 3-D capacitance extraction program. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11):1447–1459, Nov. 1991.

[NW99] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, 1999.

[OSB99] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-Time Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999.

[Ozi68] M. N. Ozisik. *Boundary Value Problems of Heat Conduction*. Oxford University Press, Oxford, UK, 1968.

[PH96] D. A. Patterson and J. L. Hennessy. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, San Francisco, CA, 2nd edition, 1996.

[PRV95] L. T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic Circuit and System Simulation Methods*. McGraw-Hill, New York, NY, 1995.

[PW97] J. R. Phillips and J. K. White. A precorrected-FFT method for electrostatic analysis of complicated 3-D structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(10):1059–1072, Oct. 1997.

[RBMH98] S. Rzepka, K. Banerjee, E. Meusel, and C. Hu. Characterization of self-heating in advanced VLSI interconnect lines based on thermal finite element simulation. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part A*, 21(3):406–411, Sept. 1998.

[RSHK05] S. Rajapandian, K. Shepard, P. Hazucha, and T. Karnik. High-tension power delivery: Operating 0.18μm CMOS digital logic at 5.4V. In *Digest of Technical Papers, IEEE International Solid-State Circuits Conference*, pages 298–299, Feb. 2005.

[Sem] Semiconductor Industry Association. International technology roadmap for semiconductors, 2006. http://public.itrs.net/Links/2006Update/2006UpdateFinal.htm.

[SQP] Sequential quadratic programming. http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/continuous/constrained/nonlinearcon/section2_1_1.html.

[SSHV03] K. Skadron, M. R. Stan, W. Huang, and S. Velusamy. Temperature-aware microarchitecture. In *Proceedings of the Annual International Symposium on Computer Architecture*, pages 2–13, Jun. 2003.

[SY95] S. M. Sait and H. Youssef. *VLSI Physical Design Automation: Theory and Practice*. IEEE Press, Piscataway, NJ, 1995.

[TK00] 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*, 19(2):253–266, Feb. 2000.

[WC02] T. Y. Wang and C. P. Chen. 3-D thermal-ADI: A linear-time chip level transient thermal simulator. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(12):1434–1445, Dec. 2002.

[WLM00] S. C. Wong, G. Y. Lee, and D. J. Ma. Modeling of interconnect capacitance, delay, and crosstalk in VLSI. *IEEE Transactions on Semiconductor Manufacturing*, 13(1):108–111, Feb. 2000.

[WM04] B. Wang and P. Mazumder. Fast thermal analysis for VLSI circuits via semi-analytical Green's function in multi-layer materials. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 409–412, May 2004.

[WWFH03] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In *Proceedings of the Annual International Symposium on Computer Architecture*, pages 84–97, Jun. 2003.

[WWMS79] W. T. Weeks, L. L. Wu, M. F. McAllister, and A. Singh. Resistive and inductive skin effect in rectangular conductors. *IBM Journal of Research and Development*, 23:652–660, Nov. 1979.

[YW98] C. P. Yue and S. S. Wong. On-chip spiral inductors with patterned ground shields for Si-based RF IC's. *IEEE Journal of Solid-State Circuits*, 33(5):743–752, May 1998.